

A Connectivity Analysis Approach to Increasing Precision in Retrieval from Hyperlinked Documents

Cathal Gurrin & Alan F. Smeaton

School of Computer Applications
Dublin City University
Ireland
cgurrin@compapp.dcu.ie

Abstract

Within the last few years very little has been made of the usefulness of Connectivity Analysis to Information Retrieval on the WWW. This document discusses hyperlinks on the WWW and our experiments on the exploitation of the immediate neighbourhood of a web page in an effort to improve search results. In order to test the hypothesis that Connectivity Analysis can increase precision in the top ranked documents from a set of relevant documents, we developed a software application to generate and re-rank a query relevant subset of the WT2g Dataset. We discuss our software in depth and the problems that we encountered during development. Our experiments are based on implementing a number of re-ranking formulae, each of which tests the usefulness of different approaches to re-ranking a set of relevant pages, ranging from basic context analysis (inLink ranking) to combined content and context analysis approaches.

1. Introduction

Within the last few years very little has been made of the usefulness of Connectivity Analysis to Information Retrieval on the WWW. Connectivity Analysis as an IR technique is based upon the identification and exploitation of latent linkage information inherent in the structure of the WWW. In fact, as the WWW grows it becomes increasingly difficult for standard IR approaches to operate effectively. However hypertext links, the building blocks of Connectivity Analysis are constantly increasing in number and becoming more important.

Our approach outlined in this paper is to utilise connectivity information to improve the ranking of results from web search. In effect we are attempting to improve the precision of the top documents returned from a search as it has been shown that up to 85% of users only look at the first page of search results. We do this by developing a number of approaches each of which is based around re-ranking a set of 'relevant' documents by one of a number of popularity ranking formula. In so doing we distinguish between different types of links to be found on the WWW of today, even though only one syntactic type of link is supported by HTML.

2. Background and Hypothesis to Test

An author writing a WWW document will create different types of hyperlinks (or edges) between documents, even though HTML supports only one syntactic type of hyperlink (represented in HTML by the <a> tag). In fact web page authors will most probably not be aware of the significance of the different link types that they are creating. In [4] Spertus discusses hyperlinks and varieties of hyperlink information, based on information mined from identifying the target of each link. She states that much human thought has gone into creating each hyperlink and labelling it with anchor text, which forms the basis of the approaches taken in [1,2,3].

2.1 Analysis of Link Structure

The WWW in its present form is said to consist of approximately 1 Billion web pages and up to 10 Billion associated hyperlinks. This ratio of approximately 1:10 can easily be viewed in the context of the 2GB WT2g DataSet we use in the TREC-8 web track. Here we have 247,491 individual HTML documents with a total of 2,254,515 links (ratio is 1 : 9.1) from these documents to other documents. Henzinger & Bharat in [5] discuss a Connectivity Server, developed in the AltaVista Labs, that stored 1 Billion edges related to 100 million nodes, generated from AltaVista crawls. This Connectivity Server would have represented the adjacency matrix $A(G)$ for the AltaVista WWW index. In the course of our research we had to develop a Connectivity Server to handle queries for our own search software. Details will follow below.

Generally on the WWW we can separate links into one of two types based on their intended function when being created:

- **Structural** (upward, downward or crosswise) links that link separate nodes within a particular domain. They exist to aid the user in navigating within a domain, or web site.
- **Functional** (content, or outward) links on the other hand link nodes in different domains (across web site boundaries). They can be seen to mostly link from a source node to a target node that contains similar and, in the author's opinion, useful information. The author of a particular node is likely to have found the target node useful and related to the concept explored in the source node, and for that reason created the link.

In the course of this research we are more interested in the latter type of link as opposed to the former. We view all nodes within a domain as having being written/developed by the one author and representing the ideas of a single individual or organisation. Consequently the former can not be seen as a source of useful information for Connectivity Analysis because we can not allow individuals to directly influence the popularity of their own web pages (spamdexing). See [4] for a more detailed discussion of Structural links and their uses. Utilising Functional links can give us a means to judge the popularity of a particular page to WWW page authors and consequently we can generate a popularity rank for the page. When extracting information from hyperlinks on the WWW, we assume two properties inherent in hyperlinks from [6], these are:

- A link between two documents implies that the documents contain related content.
- If the documents were authored by different people, then the first author found the second document valuable.

Additionally, one can assume that a page with a large number of Functional *outLinks* will act as some type of index page and be a source of links to content that the author of the index (source) page found relevant and useful. Similarly a page with a large number of Functional *inLinks* could be seen as a popular page precisely because a lot of people have found it useful, and consequently linked to it. One could take this a step further and conclude that a document on a topic such as *Formula 1 Motor Racing* which has *inLinks* from a number of other documents relating to a similar topic, would be more useful to a user looking for information on *Formula 1* than a document which may have a number of *inLinks* from more diverse sources. It is this belief that has influenced our TREC approach and led Kleinberg [7], Brin & Page [8], and others to their conclusions as to the usefulness of Connectivity Analysis as an alternative/complementary approach to general IR on the WWW.

Kleinberg [7] describes the WWW as consisting of multiple communities of documents based around themes. He puts forward the concept of *Hubs* and *Authorities*. A *Hub* page is a page that acts as a source of links to related content. A *Hub* page will typically contain many more *outLinks* than *inLinks* and may be seen as a form of index page. An *Authority* page on the other hand, is regarded as a source of information (an authority) on a subject and is pointed at by *Hub* pages. *Hubs* and *Authorities* exhibit a mutually reinforcing relationship, that is, the higher the quality of the nodes that point to a page, the higher the quality of that page, and consequently this in turn increases the quality of any pages that link into that page. His experiments show that broad queries on broad topics can benefit greatly from Connectivity Analysis where his software (HITS) can select the most popular (best-connected) document from within a linkage-based community on the WWW. In addition, we can categorise results from a query into linkage based clusters by implementing a form of eigenvalue decomposition on a connectivity matrix generated from the immediate neighbourhood of the Result Set which has been generated in response to a query.

2.2 Representation of the Link Structure of the WWW

A natural method of representing the WWW is as a directed graph with (a finite, non-empty set of) nodes representing documents and edges representing the links between these documents. For a detailed explanation of this

concept see [9] where Adamic describes the WWW in terms of a small world graph. An edge represents a directed (source to target) link between two nodes because the HTML hyperlink is uni-directional. Within this graph we can select a sub-graph (G), consisting of a subset of the graph's nodes and a subset of the graph's edges. Before we can work with a graph such as G we need a method of representing it. The adjacency matrix of the graph is a suitable method of doing this. In the adjacency matrix $A(G)$ for G , the entry in row i and column j is 1 if the nodes i and j are joined by an edge and 0 otherwise. Of course an adjacency matrix for the entire WWW at a given point in time would be enormous, sparse and very difficult to model. Our representation of connectivity information as an adjacency matrix opens up whole new mathematical approaches to IR on the WWW, see [7]. Assuming the adjacency matrix $A(G)$ can represent the *outLinks* of the set of documents, the vast majority of *inLinks* into these documents can be represented by $A^T(G)$ which is the transpose of the matrix $A(G)$. A Connectivity Server, similar to the one described in [5], would operate using an up-to-date adjacency matrix for all the documents indexed by it.

A further implementation of graph theory would allow for the generation of a weighted graph representing the WWW (or a subset thereof) in which each edge would have an associated weight. A number of formulae that we have implemented in our research require the use of weighted edges in a sub-graph of relevant documents. This sub-graph having been generated by implementing a content analysis search on the dataset to select query relevant documents. By using a weighted graph we can regulate the proportional relevance of link types as we find links to be of varying importance, depending on the context in which they were created.

The hypothesis we set out to test was whether, given a set of 'relevant' documents, re-ranking them by a popularity measure would improve the precision of the top results, and consequently populate the higher rank positions in a result set with a larger number of higher quality/popular documents. Our approach is based on the assumption that, by implementing a conventional text-based search on the dataset, a small subset of nodes that are relevant to the topic in question will be generated. The execution of various formulae on this small subset will then result in applying higher weights to the most popular nodes and therefore increase the ranking of the related documents in the result set.

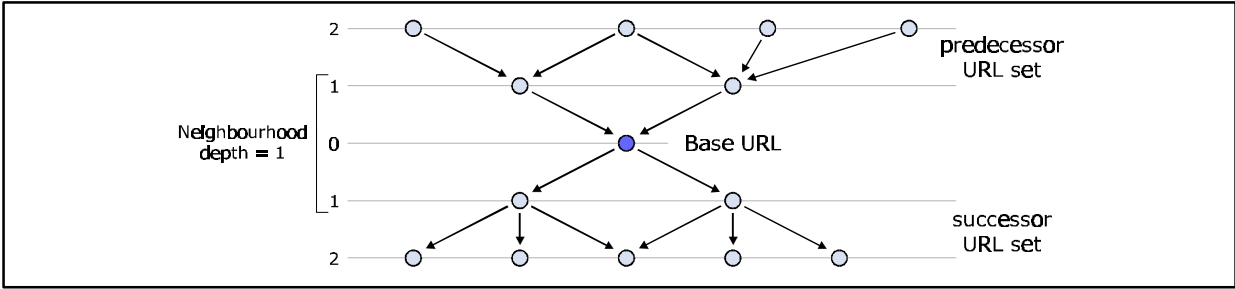


Figure 1: Link Types

Our approach was developed in order to provide higher quality results to a search by focusing on the immediate neighbourhood of the document in the WWW graph, to a depth of 1, see Figure 1. In [1] Li describes the Hyperlink Vector Voting method which ranks a document on the basis of the number of hyperlinks pointing into it (in its immediate neighbourhood) and uses the hyperlinks anchor text as an indication of the semantic content of the target document. He compares the approach with 'voting' for results on the WWW, i.e. a document's popularity & content is dependent on other authors, and not directly on the author him/herself. For additional approaches to suggesting the content of a document, by looking at the anchor text from *inLinks* into the document, see [2,3].

3. System Overview

As previously stated, the reason for having a Web Track in TREC-8 was to investigate whether Connectivity Analysis is a useful aid to web search. In order to test whether the associated hypothesis is true, we developed a software application which would produce results based on conventional Content Analysis (the baseline result) and then re-rank those results based on a number of related Connectivity Analysis approaches. Our approach was based on using the WT2g dataset, consisting of 247,491 HTML documents at 2GB storage requirements. Although we felt that the 100GB collection would be more useful as a research tool, we didn't have the storage capacity available to handle such a large dataset at that time.

3.1 DataSet Preparation

The TREC dataset and the connectivity data required pre-processing before they were of use to our software. The 247,491 HTML pages were contained in 1081 separate text files. Each HTML page had to be extracted and stored on disk before our Content Analysis software would index it. Consequently we developed an application to extract all 247,491 HTML pages from the source. These pages were given a name based on the TREC document ID (e.g. "WT04-B21-15.html") and saved to disk. As these files were being generated, we extracted a small amount of information from each document, which we stored in a database. The information we extracted consisted of:

- Document ID (internal unique representation for each document)
- Original document URL
- Document title (where available)

This information would later be used to generate intuitive results for each query, see Figure 4 for an example. This information was stored in a SQL Server Database and accessed at query execution time. It was found after testing that the Database Server we used, although it was on an under-powered PC, was capable of handling over 100 queries per second, which proved sufficient for our needs.

Once all pre-processing of the source files had been completed, files written to disk and our Database indexes generated, our Content Analysis software indexed the files. To avoid having to develop our own Content Analysis software, we used AltaVista Discovery. Discovery is a desktop term-based search application provided by AltaVista and freely downloadable from their web site. Discovery usually indexes Word, Excel, Email files etc... but also is capable of indexing HTML files stored locally on disk. Consequently it suited our needs for a term-based search engine that was capable of accepting queries and returning sets of highly-scored documents for additional processing. It took Discovery in the region of 160 hours to complete the processing of the dataset. Much of this processing was done in stages at night and at weekends as the PC was being used during the day to develop the search software. Once completed the index files generated by Discovery amounted to 1GB in size, approximately half the size of the dataset itself. We found a number of limitations with using Discovery for this task:

- Discovery will only accept requests from the local PC on IP address 127.0.0.1. Consequently all searches were run on the computer that contains Discovery, the Discovery index and the Dataset. Since our official runs, we have developed a Server to handle queries from a computer other than the one Discovery is installed on and pass back the result. Discovery is currently installed on a number of computers, each of which indexes a different version of the dataset.
- Discovery would only list the top 200 documents in response to a query. We found no way around this limitation and consequently our results only ever contained a maximum of 200 documents, even when more were deemed relevant. We avoided expanding the set of 'relevant' documents using the neighbourhood of these documents as in [7] due to the fact that our Result-Set already contained too many irrelevant documents and that by expanding this set would incorporate even more irrelevant documents and led to severe 'topic-drift' problems.
- Discovery was slow at providing responses to queries. Most queries took in the order of 15 seconds to produce content-only results for a query. Using recently acquired hardware we have reduced this time to below 2 seconds per query.

Due to latency problems encountered when using the on-line Connectivity Server we decided to develop a local Connectivity Server using the downloadable connectivity data provided by the Web Track organisers. Once it was made available we developed a simple application to process the connectivity files and insert the linkage data into another SQL Server Database which acted as a local Connectivity Server. From our point of view it provided us with a version of the

A(G) Adjacency Matrix for the entire WT2g Dataset, both for *inLinks* and *outLinks*. The weight of each edge between nodes was set to a default value of 1. The connectivity server worked by accepting a document id *i* and returning a list of all document id's that pointed to *i* or are pointed at by *i*. The internal structure of the Connectivity Server is very simple, consisting of simply source node id and target node id pairs to represent each link between two documents.

The software was implemented in JAVA under Windows NT, and the following two computers were available for our use at that time:

- PII 350, 64MB RAM, 8GB HDD, Windows NT 4
This computer provided all the processing power for the search session, accepted queries, ran Discovery, processed the result set and generated results for the user.
- P 120, 32MB Ram, 1GB HDD, Windows NT Server 4 & SQL Server 6.5
This computer provided permanent storage for document indexes and acted as a Connectivity Server.

These machines have since been replaced by a suite of networked PCs, which will be dedicated to our further work in this area.

3.2 Query Generation Technique

We took two approaches to developing queries from the topics 401-450. The first approach (submitted to TREC as official runs) was based on the title of the topic alone. However we felt this would adversely influence the quality of results produced in that a more advanced automatic, or manual technique would produce higher quality queries and consequently Result-Sets containing a higher proportion of relevant documents, as well as being more representative of a normal user's query on the WWW. Accordingly we developed a set of queries which were manually generated from the Topics and ran tests using these queries, the results of which are in this document. We found that the manually generated queries produced results that increased the precision of the baseline (content-only results) for the top 30 documents, which is what we are interested in when researching web search techniques. Figure 2 provides exact values for content-only precision at 5 to 1000 documents for both modified and unmodified queries. Use of the modified queries, as opposed to unmodified queries increases the percentage of relevant documents in the 'root-set' from 8.81% relevant to 22.29% relevant, but decreases the overall number of documents returned from 7964 to 1857, thus giving smaller, but more focused content-only result sets.

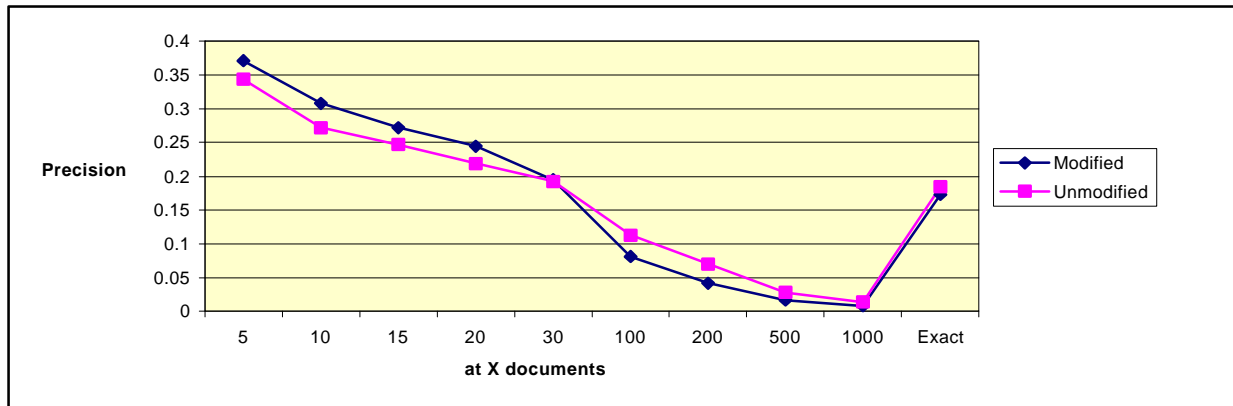


Figure 2: Baseline relevance depending on Query

4. Experiments

4.1 Re-ranking

In order to test the usefulness of various Connectivity approaches to web search our experimental software generates a content-only result set for a query. This is done by sending the query to our Discovery server, reading back and parsing the results. This baseline set of documents, the 'Result-Set', can be expected to consist of documents relevant to the query, or at least highly scored documents, although this was not as successful as we had hoped. We developed simple re-ranking schemes to re-rank the this set of relevant documents which would allow for easy visualisation of the differences

between a content only approach and a context analysis approach to Web Search. Consequently the process of executing a search on our software consists of 2 stages, a content analysis stage and a context analysis stage.

4.2 Stage 1 - Content Analysis

Each search firstly consists of a content analysis stage performed by Discovery on the test collection. The user enters a query, which is passed to the Discovery Server (an application running on a computer that is running Discovery), which in turn, sends the query (unmodified) as an HTTP request to Discovery (on the local machine), and passes the generated result back to the client computer. Discovery queries it's indexes and returns the top 200 results (max) to the query. The result returned from Discovery is then parsed to produce a set of up-to 200 documents (the Result-Set), or baseline result. The benefit of this is that a set of documents is generated which is assumed to be relevant to the query. Overall we found that, at best, only 22.29% of the Result-Set returned from Discovery was deemed relevant to the queries in topics 401-450. This Result-Set is then available for further processing. This completes Stage 1 and the results are written to disk. In this way we provide content-only results to the queries, which were submitted to TREC as an official content-only run (*dcu99c01*).

4.3 Stage 2 - Context Analysis

Stage 2 consists of a context analysis phase that re-ranks the Result-Set generated in Stage 1, based on a number of popularity ranking formulae. Each formula was developed as a separate algorithm (each of which we refer to as a numbered variation), which could be executed on the Result-Set to generate results. In all we developed 7 variations of the re-ranking algorithm (and associated formulae), each of which could be executed independently, i.e. without affecting the execution of, or results produced by any other variation. In all of the variations, if the P_n (popularity of document_n) scores of a number of documents are equal, the original document order given by Discovery is used to decide on the final ranking given to these documents.

In Connectivity Analysis we generally assume that the more popular a document is, the more *inLinks* that document will have on the WWW, hence:

Variation 1

$$P_n = \sum \text{inLinks}_n$$

In this case the P_n score is based purely on the number of *inLinks* to document_n. This we developed as the first variation that produced results for further evaluation.

However this approach is rather too simplistic. What would happen if a site such as *Yahoo* or *Microsoft* were to be contained in the Result-Set. Regardless of the topic in question these sites would be ranked highest due to the number of *inLinks* associated with them. Although none of these sites are represented in the WT2g dataset, we had to take this possibility into account. Consequently we would recommend limiting and in our research did, limit the number of *inLinks* to a maximum of 50, resulting in:

Variation 2

$$P_n (0..50) = \sum \text{inLinks}_n$$

In addition to ranking by *inLinks*, we wanted to recognise the potential of a document to act as a source of links to further information (a *hub* document). Accordingly we developed a third variation that took the number of *outLinks* into account as well as the number of *inLinks*. Once again we limited the number of *outLinks* considered for any one particular document, however this time we set the limit to 20. This was in order to avoid some large index type documents swamping the results. In addition we felt that *inLinks* would be far more relevant in generating a popularity score for documents than *outLinks* so we set the number of *inLinks* to be weighted 4 times the weight of the *outLinks*, giving the following formula:

Variation 3

$$P_n = (4 * \sum \text{inLinks}_n (0..50)) + (1 * \sum \text{outLinks}_n (0..20))$$

Increasing the weighting of *inLinks* would mean that our sub-graph would have constituent edges weighted at 4 if their targets were the node in question, but any edges with their source being the node in question would retain the default weighting of 1. This formula that we called *Rerank* was entered as one of our linkage-based runs to TREC (*dcu99l01*). This approach (assuming a Result-Set set of 200 documents) required 400 SQL Queries (this could have been reduced to 200 queries, if necessary) and consequently required 4 seconds or so to complete. The processing time required for *Rerank*, aside from the SQL queries is negligible, just the calculation of P_n 200 times.

Building on the third variation, we developed the fourth that combined both context and content analysis in Stage 2. Instead of simply re-ranking each document based on a popularity formula we allowed the ranking given in Stage 1 by AltaVista Discovery to influence P_n up to the same extent as *inLinks*. We felt that this should help the ranking process to keep the most syntactically relevant documents (as chosen by Discovery using its term matching) near to the top of the results. The formula we used can be seen below:

Variation 4

$$P_n = (2 * \sum \text{inLinks}_n (0..50)) + (1 * \sum \text{outLinks}_n (0..20)) + (\text{discWt}_{(200..0)} / 2)$$

This formula was called *RerankAdv* and required one additional modification before it was finalised. The type of links used in calculation of P_n for each document was also taken into account.

As we discussed earlier, it is intuitive that we only want to count Functional links and ignore Structural links altogether from the re-ranking equation. Consequently we had to check each *inLink* and *outLink* to determine its type. This was done using the domain strings of the node in question. We selected the lowest level substring of the domain (e.g. www.microsoft.com) for the base URL and the target URL and compared them. If the two were the same URL then we recorded the match and ignored the link as it was considered to be Structural. We felt that this approach was more valuable than ignoring all links to `***.microsoft.com` as it was felt that a URL such as research.microsoft.com was not to be considered authored by the same author as www.microsoft.com. The one author per domain assumption would not apply in this case. Had we taken this strict Functional-only approach we would have been left with a subgraph of the Connectivity Data, but with only Functional (across domain) links remaining.

Desirable as the approach to only include Functional links may seem, we found that the 2GB dataset prohibited us from fully implementing this approach. This was due to a lack of Functional links between documents in the dataset. To overcome this problem and to produce acceptable results we had to allow Structural links to exert some influence on the P_n score of each document. In a real implementation of this approach on the WWW this would not be the case as re-ranking by Structural links serves to increase the ranking of documents from large, well inter-connected (within their domain), web sites. Consequently we weighted Functional *inLinks* at 4 times the weight of Structural *inLinks* and limited the total link score (inLink_n) for each document to a maximum of 50. This replaced our subgraph with a weighted subgraph with a select few edges (Functional) having a weight of 4, with the rest having the default weight of 1. The following formula was used to calculate the *inLink* score of a node:

$$\text{inLink}_n(0..50) = ((4 * \sum \text{funct}_n) + \sum \text{struct}_n)$$

This fourth variation, *RerankAdv*, was submitted as one of our preliminary runs to TREC (*dcu99l02*). In addition to these variations another two were developed in order to clarify the benefits (if any) of re-ranking by both Functional and Structural *inlinks*. The fifth was to re-rank the Result-Set based on the number of Functional *inLinks* alone and the sixth was to re-rank based on the number of Structural *inLinks* alone:

Variation 5

$$P_n = (\sum \text{funct}_n)$$

Variation 6

$$P_n = (\sum \text{struct}_n)$$

The seventh and final variation we developed was an attempt to overcome the lack of Functional links in the dataset. Our approach was to utilise the connectivity information for each document that would be contained in the WWW as a whole. The problems of using live WWW connectivity data on the 2GB Dataset were threefold:

- many of the documents were no longer in existence as the documents from which the Dataset had been generated had been spidered in early 1997.
- many may not have been indexed by the Connectivity Source we were querying.
- a number would have had their content modified since spidering and thus the current WWW connectivity information would be considered invalid.

In order to get around this problem we replaced the actual URL of the document with the URL of its domain. This shortened URL was then used to query the AltaVista search engine using the "link:URL" query that returns the number of and actual *inLinks* into the document in question. We then ranked by the popularity of the main page (index.html) within each domain.

Variation 7

$$P_n = \sum \text{inLinks}_m \text{ (m = lowest level of domain string)}$$

5. Results

We entered three of our experimental approaches as preliminary runs into TREC in August 1999, two of which were linkage-based and the third a content-only based on the results of Discovery. The linkage-based runs (*Rerank – variation 3* and *RerankAdv - variation 4*) were detailed earlier in this document. Two of our runs were added to the pool for relevance judgement purposes (*dcu99c01* & *dcu99l02*). Additional experiments that we ran after submitting our official runs found little or no improvement in precision when incorporating Connectivity Analysis in the ranking process in the majority of cases. Our experiments find that using Functional links from within the supplied connectivity data provides the best results of all the Connectivity Analysis approaches outlined in this document. However the number of Functional links within the dataset are extremely limiting at less than 1% of the total links and therefore only a limited number of documents would be re-ranked from any one query. Consequently we are currently unable to come to any definite conclusions as to the benefit of re-ranking by Functional *inlinks*. We hope that the 10GB dataset next year will contain a higher percentage of Functional links. Figure 3 below shows the precision at 5 - 1000 documents returned from running the modified queries on WT2g.

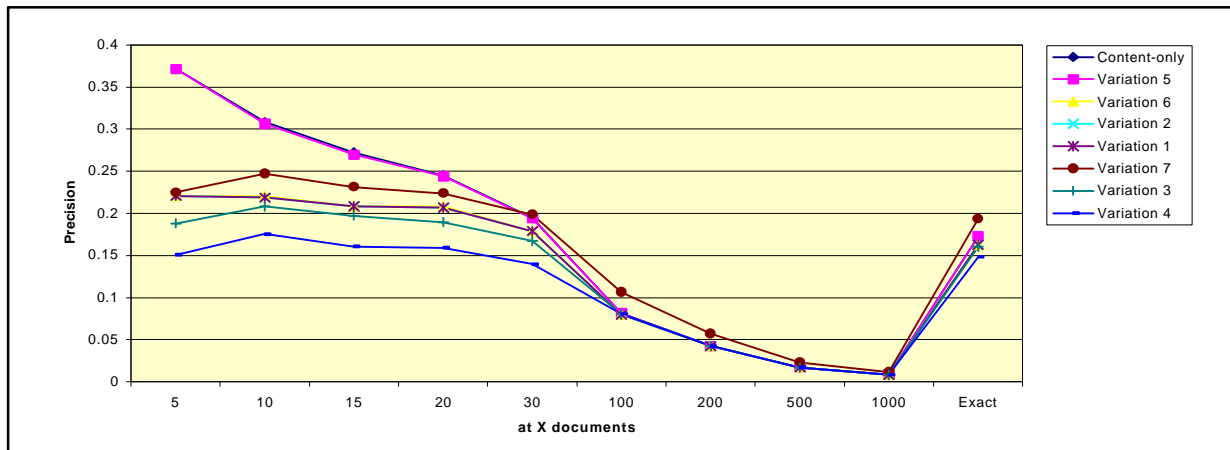


Figure 3: Precision at x Documents

We found that re-ranking by Structural *inLinks* is not viable either as this would only serve to rank highest documents from within large, well inter-connected domains. Due to the lack of Functional links in WT2g, we found that the scores for Variation 1, Variation 2 and Variation 6 are almost identical. Variation 1 and Variation 2 do not distinguish between Functional and Structural link types, yet the P_n score associated with them is almost identical to the P_n score of Variation 6 which ranks by Structural *inLinks* only. None of these Structural re-ranking approaches improve precision, in

fact all decrease precision at 10 documents by at least 28.5% from .3082 to (at best) .2204. The results from Variation 7 (which used live WWW connectivity data) were noteworthy due to the fact that this was the only approach that increased precision over content-only results, but only from 30 to 1000 documents.

In addition to topics 401-450, we have executed a number of manual queries on the software. Some of these queries have produced quite impressive results using the WT2g dataset and associated connectivity data. See Figure 4 for an example of the results generated by a query "Vegetable Soup Recipes". The variation used to generate these results was a modification of the *RerankAdv* variation. Further research is needed to determine if these results are due to particular properties of the dataset that favours certain queries.

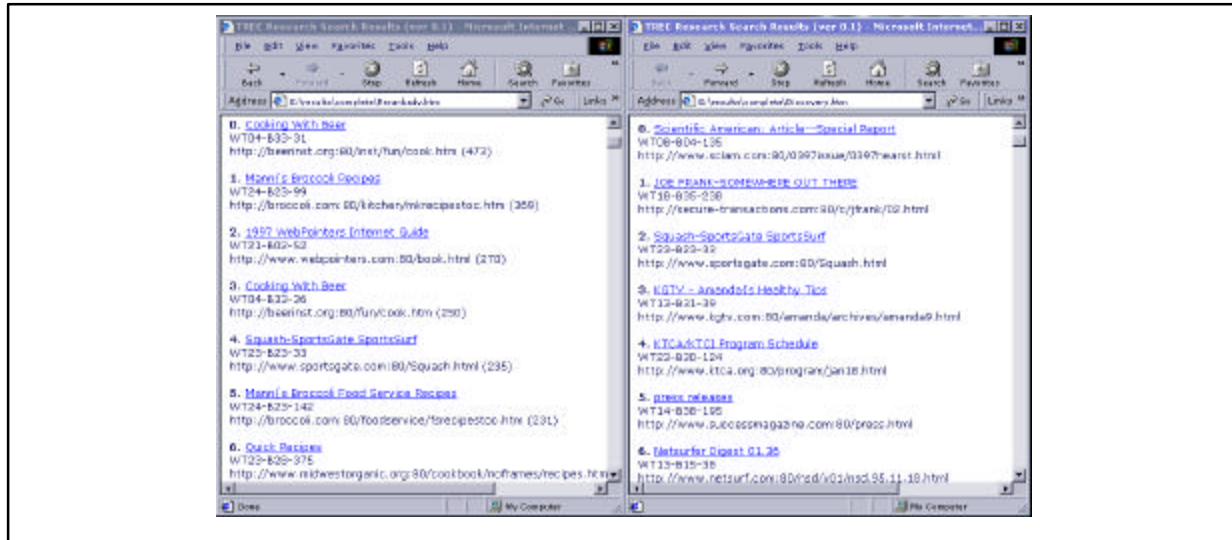


Figure 4: Example Result (left = context analysis, right = unmodified content analysis)

7. Conclusions and Future Research

We found that not enough Functional links existed in the dataset to allow many of our variations to function correctly. There was no way around this Functional link problem, except to turn to the wider WWW as a possible source of the connectivity data. See our seventh and last variation for details of how we implemented this. We are not yet in a position to draw solid conclusions from our experiments, but we plan to continue our research into this area. Of course, one of the drawbacks of these (pure) Connectivity Analysis approaches is that documents which have a large in-degree (number of *inLinks*) in the Result-Set will be ranked highly, even above other documents with a higher relevance to the query but with a smaller in-degree. This is especially true in our case, since the Result-Sets that our Content Analysis stage generated contained at most only 22.29% relevant documents. It is entirely feasible that many of our top ranked documents can not be considered relevant to the query at all, yet were ranked highly because of their popularity. We intend to implement our approaches in the future using a Confidence Factor that will indicate when and to what extent Connectivity Analysis should influence the search process. This problem of returning results that may be not entirely related to the topic represented by the query is discussed in [6,7].

Many more advanced Connectivity Analysis approaches have been developed such as those detailed in [3,6,7,10]. We didn't implement any of these approaches, as we felt that more was to be gained from developing our own ideas, with the knowledge that the other methods existed. It is our understanding that any implementation of these approaches would not succeed in improving precision (to any usable extent, if at all) when the experiments were based on the WT2g dataset, due to the lack of Functional links. We do suggest caution being taken when reviewing the Small Web Task to take the results in the context of the WT2g dataset, lest one conclude that Connectivity Analysis does not improve precision in any case. From the work of Kleinberg in [7], it is generally accepted that for queries on broad topics, Connectivity Analysis

will allow for the selection of the most popular (densely linked) documents from within a WWW community in response to a query, in addition to automatic result clustering.

Next year, we plan to partake in the Web Track using the 10GB subset and if available the 100GB collection. We hope that the 10GB dataset planned for TREC-9 will be more representative of the link structure of the WWW. If we find this not to be the case we will look into spidering our own set of HTML documents utilising a software spider that we would develop. The spider would gather documents by following the link structure of the WWW but limiting the number of documents from any particular domain (with exceptions) and implementing a priority queuing algorithm to select the most useful documents to download. It is our belief that a dataset that is gathered with the purpose of aiding experimentation into Connectivity Analysis will be vital to our future research. Our more ambitious plans for next year are supported by an increase in the amount of hardware that we have available thanks to a recent philanthropic donation to aid our connectivity research project. The hardware, based around four PIII workstations and a PIII server will be connected together on a dedicated 100Mbit/s switched network. Total storage capacity on the network is almost 200 GB and each PC is capable of expanding its number of processors as requirements dictate.

References

- [1] Li, Yanhong - **"Toward a Qualitative Search Engine"**
IEEE Internet Computing, page 24-29 (July-Aug 1998)
- [2] Amitay, Einat - **"Using Common Hypertext Links to Identify the best Phrasal Description of Target Web Documents"**
<http://www.mri.mq.edu.au/~einat/sigir/>
- [3] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson & J. Kleinberg - **"Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text"**
<http://decweb.ethz.ch/WWW7/1898/com1898.htm>
- [4] Spertus, Ellen - **"Parasite: Mining Structural Information on the Web"**
<http://atlanta.cs.nchu.edu.tw/www/PAPER206.html-admin.htm>
- [5] K. Bharat, A. Broder, M. Henzinger & P. Kumar - **"The Connectivity Server: Fast Access to Linkage Information on the Web"**
http://www.research.digital.com/SRC/personal/Andrei_Broder/cserv/386.html
- [6] K. Bharat & M. Henzinger - **"Improved Algorithms for Topic Distillation in a Hyperlinked Environment"**
ACM SIGIR'98
- [7] Kleinberg, Jon - **"Authoritative Sources in a Hyperlinked Environment"**
<http://www.cs.cornell.edu/Info/People/kleinber/kleinber.html>
- [8] S. Brin & L. Page - **"The Anatomy of a Large-Scale Hypertextual Web Search Engine"**
<http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm>
- [9] Adamic, Lada A. - **"The Small World Web"**
<http://www.parc.xerox.com/istl/groups/iea/www/smallworld.html>
- [10] S. Brin & L. Page - **"The Pagerank Citation Ranking: Bringing Order to the Web"**
<http://www-db.stanford.edu/~backrub/pageranksub.ps>