CLARIT TREC-8 Experiments in Searching Web Data

Jeffrey Bennett, Xiang Tong, David A. Evans

CLARITECH Corporation

Abstract CLARITECH submitted two baseline content-only runs and completed two additional content+link runs in the TREC-8 Web Track. These represent our first serious attempt to deal with Web data, and our first automatic runs in several years. The first question was whether CLARIT would perform as well on Web data as on more traditional text. We found that, with extensive pre-processing of the raw data prior to indexing, the automatic retrieval system actually performed better on Web data than on Ad Hoc data. For the link runs, we implemented a version of the HITS algorithm [Kleinberg 1997], originally developed at IBM. Our version optimized HITS for the CLARIT environment, but also reflected some constraints imposed by limited resources. Unable to develop and sufficiently test our own matrix-processing library in time, we used a commercial product for the number crunching. Performance on the link runs was poor, but failure analysis suggests many ways to improve it.

1 Introduction

Even casual inspection of Web data reveals how different it is from traditional newswire or article text. Most obviously, it contains extensive HTML mark-up. Even apparent plain text may conceal many types of meta and tag data, as the example in Figure 1 shows.

```
<head>
<META name="keywords" content="waste water, biosolids,
waste treatment, geneva, marsh creek">
<title>Marsh Creek Waste Water Treatment Plant</title>
</head>
<body bgcolor="#ffff999" text="#006635" link="#993300"
vlink="#CC0099">
<center><a name="top"><img src="marbar.gif"
align="left"></a></center>
<br clear="all">
<img src="margenev.gif" align="left" hspace="35"
vspace="10">
<p></p><br><p></p><br>
<h1> Marsh Creek</h1>
<h1><a name="Marsh Creek Treatment Plant"> Waste
Water Treatment Plant
</a></h1>
```

Figure 1. Example of web text with HTML markup

Clearly, something must be done; a naïve parser, attempting to find words in the above text, might extract terms like "Creek</h1>" "Plant", and "<title>Marsh"—words unlikely to be found in any lexicon.

To address this problem, the CLARITECH web system does extensive pre-processing to "sanitize" the text, while preserving important information encoded in the mark-up. Non-semantic tags, such "<br>", "<body bgcolor>", or " ", are simply discarded; the system processes certain other tags with greater care.

Specifically, it removes keywords, header data (e.g., the site URL, server address, last-modified date, document length) and hyperlinks from the main text, and stores them all in separate fields. (The link field retains an offset into the main text for future context recovery.) The system also records forms, images, and an image count for possible future use. This pre-processing step allows CLARIT to work on the plain text and deliver quite satisfactory performance on the content-only runs, particularly on recall. Our best run was at or above median recall on 88% of the queries. Precision was less impressive, but still slightly above median. Surprisingly, the performance of the automatic system on Web data was better than the same system's performance on the ad hoc corpora. We performed a large number of ad hoc automatic runs (though we made no submissions to this track), and found that our system performed at or slightly above median for TREC-8 queries. This is encouraging, since we have made no attempt to "tune" our system for automatic retrieval.

Having essentially no time for pre-experiment evaluations, we took a very bold approach to the content + link runs. After a brief literature review, we decided to implement a "CLARITized" version of the Hyperlink-Induced Topic Search (HITS) algorithm. The idea is that hyperlinks can be viewed as implicit annotations that encode human judgments about relevance. Confronted with the thousands or even tens of thousands of "relevant" documents that might be returned by a general query, the link information can be used to extract a much smaller number of "authoritative" sources on the topic. These pages are likely to be of greater utility to the user than a "ranked" list of hundreds of nearly indistinguishable documents.

A page linked to by many relevant pages is said to have "authority," even when the page itself is not high-scoring. Conversely, pages containing links to many relevant documents are called "hubs." As with authorities, the hub pages themselves may be low-scoring; they may in fact have little or no content beyond the list of hyperlinks. The premise of the HITS algorithm is that authorities and hubs have more utility than isolated relevant documents, and should therefore be ranked higher.

As a first step, we used the connectivity information supplied by TREC (rather than examining our own link fields in context), and only considered links involving the top 250 or so documents in the ranked list. We used a brute-force method for the link matrix calculations, and then simply front-loaded the top 30 authorities. Our approach was "bold" in that it did not examine the content of the authority pages, and was biased toward densely linked pages that conventional retrieval had missed. (On average, 22 of the 30 authorities came from below the top 1,000.)

2 Experiment design

The content-only runs used standard CLARIT retrieval with pseudo-relevance feedback. We generated the queries automatically using the "title" and "description" fields; assigning a higher weight to title terms. The two runs we submitted differ only in the parameters used for feedback: CL99WebM extracted terms from high-scoring subdocuments in the top 10 documents, and added up to 30 new terms; CL99WebH examined the top 30 documents and added up to 50 terms. The system weighted new terms lower than existing query terms. We submitted the top 1,000 documents returned by a second retrieval using the augmented queries.

The content + link runs started with the output of the baseline runs, then analyzed the hyperlink information in top-ranked documents to identify the top 30 authority and hub pages. To produce the link submissions, we promoted these pages to the top of the ranked lists. We tested the utility of the HITS algorithm in our environment, as well as the effect of some slight modifications we made in order to capitalize on the strengths of CLARIT.

We begin by constructing a base set of N pages (S), and limiting our search for authoritative pages to this set. N is arbitrary; we chose 1,000 as a practical limit (imposed mostly by limited time and computing resources). Using the connectivity information provided by TREC, we expanded the initial result list according to the following algorithm: number the pages {1, 2,…., *n*}, and iterate through them in rank order. Copy the *i*th page to S, then also copy all the documents that the *i*th page links to as well (if they are not already in the S). Proceed until S contains 1,000 unique documents. The average number of links per page in the collection was 6.1; accordingly, the average "depth" of the expansion step was around 230 to 250 documents. We felt this would bias the results toward documents linked to the most highly-ranked pages; expanding further might have turned up densely linked networks of low-ranked, non-relevant documents. Also, we were unable to process large (3,000—5,000 square) matrices using the brute-force methods we employed. The actual depth varied considerably, from a minimum of 41 to a maximum of 410 documents. (Another approach, which we did not explore, would be to mine deeper by examining the links and adding them selectively, rather than simply dumping them all into S.)

We then constructed an n x n "adjacency matrix" A, whose $(i,j)th$ entry is set equal to a non-zero value if page i links to page j, and zero otherwise. The simplest approach would be to set link entries to 1. Our bold approach biased the results toward unretrieved documents by computing CLARIT-term-based similarity between "best-hit" subdocuments when both pages appeared in the top 1,000, and 1 otherwise. Many of these similarity scores were near zero, effectively discounting links between dissimilar documents among the top 1,000.

According to the HITS algorithm [Chakrabarti et al. 1999], we assign authority and hub weights to each document in S. Let the authority score for the *i*th document $x_i = \sum_{\in j \to i} y_j$; the sum of all pages j that link to i. Similarly, compute the *i*th hub score $y_i = \sum_{\in i \to j} x_j$; the sum all pages i links to. There is a natural feedback effect here, since authority and hub pages are closely related: good hubs are good because they point to many authorities; and, authorities are authoritative precisely because they are being linked to by good hubs. Given the contents of the adjacency matrix, and considering the set of hub and authority scores as vectors, we can derive a process that expresses this mutually reinforcing relationship and reduces it to a standard operation in linear algebra. By filling the matrix with similarity scores between 0 and 1 (non-negative values), we guarantee that the matrix processing will converge on pages containing the most dense linkage patterns.

Specifically, the authority vector update formula can be expressed as $x \leftarrow (A^T A)x$; similarly, the hub

vector update function is $y \leftarrow (AA^T)y$. This is equivalent to performing *power iterations* on A^TA and AA^T; such iteration (given non-negative coefficients) converges on the principal eigenvectors of the associated matrices.

To find the most authoritative sources, we generated both matrices, used a commercial package to calculate their principal eigenvectors, and sorted the resulting vectors by decreasing weight. We now had ranked authority and hub vectors for the documents in S. The final step was to merge the two vectors to obtain a single ranked list of "authorities." We did the merge by computing a total score for each document. If document D1 was ranked first in the authority vector and third in the hub vector, it received a score of 1/1 + 1/3 = 1.333. If D2 was ranked second in both vectors, it received a score of ½ + ½ = 1, and so on for all documents. We moved the top N authorities to the top of the original ranked list, where N was arbitrarily (based on a literature review) set to 30, and that was the submission.

3 Retrieval performance

The results show reasonable performance for the content-only runs, and poor performance for the link runs (see Table 1.)

Since our link submissions were not judged, and the HITS algorithm may well find documents that are non-relevant by TREC standards (i.e., collections of links without content), we expected the link run scores to be low, even if our approach was working and returning helpful documents.

Noting that most of the documents in the top 30 were originally ranked below 1,000, we thought that perhaps our approach had been a little too bold. A more conservative algorithm might perform a simple resorting of the original results, without bringing in previously unretrieved documents. We did two follow-up runs using this more timid algorithm; see Table 2 for these results.

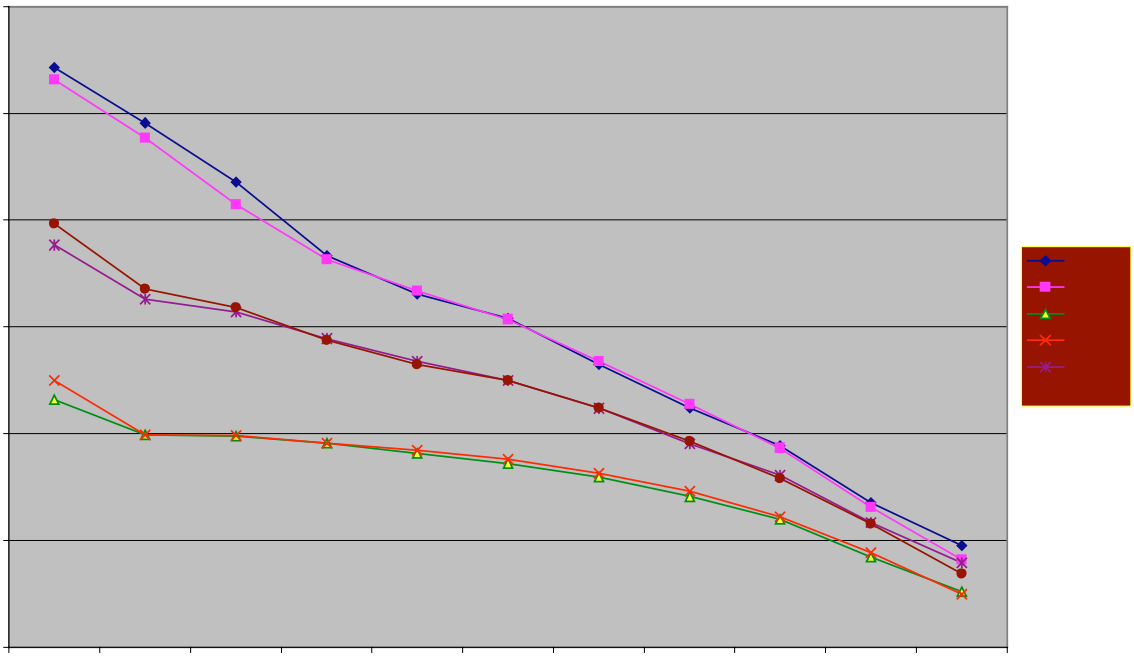
These runs fall precisely in the middle; the most irrelevant documents have been discarded, improving all performance measures, but results still fall far short of the content-only runs. See Figure 1 for the P-R curve.

4 Failure analysis of the link runs

Upon closer analysis, these factors were not sufficient to explain the poor performance. In fact, many pages had links to common "web statistics" and "hit counter" sites. There were also many links to pages giving mutual fund indices and commercial ad sites. Since we did not examine the pages for relevance, and actually preferred pages that were not returned by the conventional search, such irrelevant links formed the densest patterns!

One such site appeared in the top 30 on seven different queries, 11 were referenced by six queries, and 22 were referenced by three queries. Overall, 13% of the top-ranked documents were non-unique. If the queries are independent, all the top-ranked documents should be unique, so this degree of overlap is a sure indication something is wrong.

Our analysis confirms at least that we know the linkage pattern detection is working properly. In future work we could try several different approaches to address this problem. We could generate a stop-list of known statistics, counter, and commercial sites. We could "sanity check" the final results for overlap of top-ranked documents. We could impose a minimum score or other relevance test on documents that were not initially returned (or perhaps we should limit ourselves to resorting documents in the top 1,000, and not look beyond the initial results at all). Another strategy might be to weight the links using an adapted IDF formula. We are looking for "discriminating" links—document sets that link to each other but not to lots of other documents scattered throughout the database. A counter site might be linked to by hundreds of unrelated documents in the database; it would be assigned a very low "IDF" score, and assigned a low value in the matrix (or discarded entirely). We might even use clustering to try to identify related groups of documents within a link network.

5 Conclusion

We were encouraged by our relatively good performance on the content-only runs, particularly since our system has not been optimized to work without user feedback. The link runs were disappointing, but we can see why, and we have many ideas to address the problems. In keeping with our general belief that the next breakthrough in performance will come from customizing an approach for each query from a number of complementary techniques, we will explore the range of conditions for which link analysis is appropriate. It would appear to be most applicable for sorting through large sets of nearly equally high-scoring documents, as would result from very general queries. When the query is more specific, perhaps traditional CLARIT processing is enough; link analysis might tend to decrease performance in such cases. In the TREC-8 manual Ad Hoc task, we discovered that users added boolean