Implementing and Testing The PageRank Citation Ranking

Whitney Fish
Pomona College
wjf02006@mymail.pomona.edu

Dan Feblowitz
Pomona College
djf02007@mymail.pomona.edu

ABSTRACT

This paper presents an implementation of Google's PageRank citation ranking algorithm. PageRank is a form of link analysis that assigns each page in the corpus a score representing its importance. A page's score is derived recursively from the scores of pages that link to it. Using these scores, an IR engine can re-rank results to favor more important pages, which are likely to be more useful to the user. In this paper, we provide a means of calculating PageRank for a set of research paper abstracts. For testing, we integrated it into a TF-IDF-based IR engine that searches these abstracts. We produced an automated tester that generates a query and two sets of approximately relevant results for every document in the corpus. We tested our engine's ability to retrieve these results given these queries, and found that PageRank does not consistently improve performance in this context.

1. INTRODUCTION

The world wide web is an immense and heterogenous space of documents. According to the website http://www.worldwidewebsize.com/, which provides daily estimates of the size of the web, on the day this paper was written, the web contained at least 22.48 billion pages. Those pages vary drastically in terms of content, quality and authority - from spam advertisements to the home pages of the New York Times, Wikipedia, and Google.

This difference causes problems for IR engines that rank their results based only on measures of similiarity between a document and query. It is likely that a spam page designed to match a certain type of query scores higher on a similiarity metric than does a well-established site with useful information. For that reason, it is useful to have some idea of the importance of a page on the web.

Fortunately, corpora like the World Wide Web or collections of academic papers have a link structure that can be mined for that information. The number of links to and from a page seems to correspond to the page's influence – think of the number of pages reachable from or linking to the New York Times home page as opposed to a spam page. However, outlinks are replicable features of a page – it is easy for site owners to generate more – and numbers of links vary, depending on a page's size and content, in ways that may not reflect the page's importance. Further, a determined webmaster can even replicate inlinks by creating a dummy page that links to his site.

In order to address these problems and others, Brin et al. [2] introduced PageRank. PageRank is a score for the importance of a web page calculated recursively based on the scores of the pages that link to it. Each link accessible from a given page recieves an equal fraction of the linking page's score. Thus, merely having many outlinks does not allow a page to overly influence other page's scores. With PageRank, a link from the New York Times home page is worth more than a link from an isolated personal home page, and a spam page cannot improve its standing by adding outlinks or keywords. Even links from dummy pages do not bias PageRank, since a page generated to give out links will have few, if any, inlinks, and therefore no rank to distribute. Thus, PageRank avoids many of the pitfalls that made previous forms of citation analysis inappropriate for the web.

2. ALGORITHM DESCRIPTION

The base assumption behind PageRank is that a document's importance can be derived from the importance of the documents that link to it. An effective way to conceptualize this idea is the Random Surfer model. In this model, an average user is randomly navigating through the dataset by clicking on links. The PageRank of a document is the probability that this user will be looking at that document at some time in the future. This basic model has a couple of problems. First, the user has no defined action if it reaches a document with no outlinks (a "dangling node"). Second, if the user only navigates along links, there is a chance of getting stuck indefinitely in small sets of documents that link only to each other ("rank sinks"). The first issue can be solved by having the user jump randomly to any document in the corpus when it reaches a document with no outlinks. The second requires allowing the user to have a fixed chance of "getting bored" and jumping to a random document. [2]

From the random surfer model, PageRank can be naïvely calculated by building a matrix, G, with both dimensions equal to the number of documents, n. Each entry $G[i, j]$ is set to the chance of the random surfer moving from document i to document j. Once G has been constructed, it can be raised to a power, generally on the order of 100, and the values in each column are guaranteed to converge to a single value – that document's true PageRank. Convergence of the matrix can be proven as long as the chance of a random jump is greater than zero. The rate of convergence can be controlled somewhat by varying the chance of a random jump – a higher chance of a random jump will result in the algorithm converging with fewer multiplications. Assuming a relatively sparse link structure, a 15% chance of a random jump will lead to convergence in 50-100 iterations, even when the dataset is very large. (The rate of convergence is acutally independent of the number of documents). [2]

2.1 The Power Method

Unfortunately, multiplication of $n \times n$ matrices is computationally expensive – the best algorithms are slightly above $O(n^{2.3})$, and storage requires $O(n^2)$ memory. This is fine for small corpora, but runs into problems on larger corpora without getting anywhere close to web-scale. For example, an early naïve implementation ran on one thousand documents in about five minutes but ran into memory issues soon thereafter. One way that the algorithm can be improved is the power method, given below.

$\pi^{k+1} = \pi^k * G$ [1] [2]

Since all the rows of G will approach the same value in the basic algorithm, the power method only saves a single row of G, π, and multiplies it by G repeatedly. Dropping the rest of G does not effect the convergence of π to our desired PageRank vector. It provides significant benifits, because now instead of storing two $n \times n$ matrices, the algorithm only needs to store one matrix and the vector π. In addition, the complexity of the multiplication of a vector by a matrix is $O(n^2)$, a noticeable saving.

2.2 Implementation Details

The multiplication of a vector by a large dense matrix still becomes time- and space-prohibitive on large datasets. Fortunately, the web's link structure is extremely sparse, and based on this fact, linear-algebraic manipulation of the power method can yield a far more efficient algorithm. To start, G can be split into three seperate matrices:

$G = H + D + E$

Here, H contains only the chance of the surfer moving from one document to another along a link. D contains only the chance of the surfer making jumps because it has reached a document with no outlinks. Finally, E contains only the chance of the surfer jumping to a document as a result of getting bored (a random jump, not along one of the document's links). H will be very sparse. Using the power method and the decomposed version of G we can now write:

$\pi^{k+1} = \alpha * \pi^k * H + \alpha * \pi^k * D + (1 - \alpha) * \pi^k * E$

where α is the inverse of the probability of the surfer getting bored and making a random jump. Following Brin et al., we use a value of 0.15 for α. This results in convergence within about 60-100 iterations. [2] Notice that the first term now involves multiplying a sparse matrix by a vector. Using a sparse matrix library from the Colt project (http://acs.lbl.gov/˜hoschek/colt/), the sparse matrix can be held in memory and the multiplication can be carried out quickly.

Speeding up the calculations now requires reducing the final two terms in the equation. With this in mind, we define two new vectors: Let e be a vector of all 1's with length equal to the number of documents. Let d be the dangling node vector, where entry $d[i] = 1$ if document i has no outlinks and $d[i] = 0$ otherwise. Using these definitions, we can now re-write D and E as the multiplication of two vectors:

$D = d * e/n$
$E = e * e/n$

Substituting these values into the current equation results in the power method looking like:

$\pi^{k+1} = \alpha * \pi^k * H + \alpha * \pi^k * d * e/n + (1 - \alpha) * \pi^k * e * e/n$

which simplifies to:

$\pi^{k+1} = \alpha * \pi^k * H + (\alpha * \pi^k * d + (1 - \alpha) * \pi^k * e) * e/n$

PageRank still represents probabilities, so the entries of π^k must sum to 1. This also implies that $\pi^k * e = 1$. With this fact we can reach a final form:

$\pi^{k+1} = \alpha * \pi^k * H + (\alpha * \pi^k * d + (1 - \alpha)) * e/n$

[1]. As discussed above, $\alpha * \pi^k * H$ can be computed in an acceptable amount of time – on the order of the number of links rather than the number of documents squared. As for the rest of the equation, the vector multiplication is $O(n)$ plus another $O(n)$ operation to add the value to each entry in π. The equation also works well intuitively. $\alpha * \pi^k$ is the sum of the current PageRank of documents with no outlinks, which needs to be distributed evenly across all PageRanks, and $1 - \alpha$ is the probability that the surfer will make a random jump, which also needs to be distributed evenly across all PageRanks. In terms of memory, between iterations, we only need to store two $O(n)$ vectors and a sparse matrix. On the same machine that took 5 minutes to calculate PageRank for 1,000 documents using the unoptimized algorithm, this algorithm allowed for 30,000 documents in less than 30 seconds.

3. RESULTS

In order to test the effect of PageRank on our search engine, we developed EvalGen, an automated generator for queries and sets of approximately relevant results. For each document in the corpus, EvalGen converts the title to a query and generates two relevant result sets. The first result set includes all documents that have at least one author in common with the original document. The second includes the

[1]Superscripts here indicate the number of iterations (i.e., π^k means "pi after k iterations"). They are NOT exponents.
[2]For the sake of readability and easier understanding, from here forward, we have omitted vector transpositions. For the linear-algebra inclined, the full sequence of equations is included at the end of the document.

original document and all documents cited therein.

Using EvalGen, we produced a query and both relevant result sets for each of the 29,555 documents in the KDD 2003 corpus. We measured precicion at 20, recall at 20, MAP, and R-Precision for each relevant result set both with and without PageRank. For each of these four experiments, we took the average of each metric across all 29,555 queries. The results are given in Table 1.

On average, when the set of papers sharing an author with the query document was assumed to be relevant, re-ranking documents using PageRank caused MAP to decrease by 6.10% and R-Precision to decrease by 5.72% when compared with the raw TF-IDF ranking. However, when the set of papers linked to by the original document was assumed to be relevant, MAP increased by 3.32% and R-Precision by 3.11%.

We were surprised by the small magnitudes of these changes, and that PageRank hurt performance for the authors-based result set. We expected PageRank to improve performance consistently, though we did anticipate a smaller increase for the author-based result set than for the links-based one. This is because using links-based relevant results guarantees that each relevant result has at least one in-link. PageRank is derived from in-links, so link-based relevant results will inherently have some correlation to PageRank. Still, these results are inconsistent with our expectations. There are a number of factors that may have contributed to this inconsistency.

First, Brin et al. [2] note that "The benefits of PageRank are the greatest for underspecified queries." When the amount of information contained in a query is limited, PageRank helps push the most influential results to the top of the list, and it is likely that a user who provides an underspecified query is thinking of such a common case. Our queries, on the other hand, were full titles of research papers, which may provide enough specificity to diminish the benifits of PageRank.

Second, it is possible that the sets generated by EvalGen are not a good representation of the actual results relevant to a given query. The validity of these sets rests on certain assumptions – namely, that authors write groups of papers on generally related topics, and that papers usually cite other papers on related topics. If these assumptions do not hold over the corpus, the sets produced by EvalGen may not be reliable, though we are unsure how this may have caused PageRank to hurt search performance.

Most importantly, PageRank was designed with the web in mind rather than academic papers. It is calculated in such a way to compensate for variation in length and number of links, and to address variation in quality. Research paper abstracts are far more consistent along these dimensions – all of them represent content that has been peer-reviewed and accepted as having a certain level of quality. Web pages, on the other hand, can be created and published with no oversight. Since the quality of the KDD documents was uniform and high, we believe PageRank was not as useful for differentiating them as it would have been for web pages.

4. CONCLUSIONS

In this paper, we described an implementation of the PageRank citation ranking which we applied to a corpus of resarch paper abstracts. We automatically generated queries from the titles of papers and approximations of relevant results sets based on links and common authors. We found that relative to raw TF-IDF ranking, PageRank decreased performance slightly for the author-based results set and increased it slightly for the link-based set.

Further research would be useful to explain why this was the case. An inquiry could also be made into generating different queries and relevant results sets. Testing our implementation on a web corpus could confirm whether or not the confusion arose from the inherent quality differences between web pages and scientific paper abstracts.

5. APPENDIX

Full equations from Section 2:

$$\pi^{(k+1)T} = \pi^{(k)T} * G$$
$$G = H + D + E$$
$$\pi^{(k+1)T} = \alpha * \pi^{(k)T} * H + \alpha * \pi^{(k)T} * A + (1-\alpha) * \pi^{(k)T} * E$$
$$D = d * e^T/n$$
$$E = e * e^T/n$$
$$\pi^{(k+1)T} = \alpha * \pi^{(k)T} * H + \alpha * \pi^{(k)T} * d * e^T/n +$$
$$(1-\alpha) * \pi^{(k)T} * e * e^T/n$$
$$\pi^{(k+1)T} = \alpha*\pi^{(k)T}*H+(\alpha*\pi^{(k)T}*d+(1-\alpha)*\pi^{(k)T}*e)*e^T/n$$
$$\pi^{(k)T} * e = 1$$
$$\pi^{(k+1)T} = \alpha * \pi^{(k)T} * H + (\alpha * \pi^{(k)T} * d + (1-\alpha)) * e^T/n$$

6. REFERENCES

[1] A. N. Langville and C. D. Meyer. The use of the linear algebra by web search engines, 2004.

[2] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1999.

Table 1: Performance

	Relevant Results	Precision@20	Recall@20	MAP	R-Precision
With PageRank	By Authors	0.0986	0.224	0.185	0.195
	By Links	0.110	0.388	0.323	0.324
No PageRank	By Authors	0.0994	0.230	0.197	0.207
	By Links	0.0971	0.368	0.313	0.315
Percent Change	By Authors	-0.883	-2.33	-6.10	-5.72
	By Links	13.5	5.54	3.32	3.11