file/after/tracklang.stypackage/after/bidipackage/after/luabidi

Data Mining Homework 3 Report

Introduction

Finding the most important pages on the internet has been an important part of the success of many companies such as Google and Yahoo!. Even today, many companies spend time making sure that their site appears at the top of search engine results. Deciding the importance of a webiste individually presents a difficult challenge. For this reason, the field of link analysis, which analyzes the importance of different nodes in a graph, takes a look at the children and parents of each node. For a given page, the pages that link to it and the pages it links to can inform us about the relative importance of a site. The graph of the internet can be seen as a directed graph where each page, represented as a node, is connected to other pages via incoming and outgoing links.

This assignment involved implementing influential link analysis algorithms on a series of graphs, calculating the values relevant to that algorithm. The three algorithms investigated were HITS, PageRank, and SimRank. Following is a discussion of the algorithms, their implementation and results, and further discussion on the strenghts and weaknesses of each.

Algorithm Analysis

HITS

The HITS algorithm was one of the first algorithms to analyze pages. This algorithm relies on a few key observations: some pages are not authoritative in themselves, but they link to many important webpages, i.e. their outlinks point to other authoritative pages. The HITS algorithm considers two factors for each page: its **authoritativeness** and **hubness**. The former is the measure of importance from its inlinks, while the former is the importance of the site's outlinks.

These two factors rely on a mutual recursion

$$\text{auth}(p) = \sum_{c \in par[p]} \text{hub}(c) \tag{1}$$

and

$$\text{hub}(p) = \sum_{a \in ch[p]} \text{auth}(c) \tag{2}$$

where p is the page in question. We implement the algorithm in an iterative manner rather than recursive. We start of by considering a group of nodes, and each iteration we examine the authority and hubness of its children and parents, respectively. The algorithm stops when the sum of the difference between the previous hubs and authorities drops beneath a threhsold, which in this assignment is set to 1.

In our code, we initialize the authorities and hubs as an array of ones. Then, we loop through the vertices in our graph, and update the hubness and authority for each node v. Node v's authorities and hubness is the sum of parent's hubs and children's authorities, respectively. Since we update the authorities and hubs every iteration, over time this means we are taking into account nodes farther away. The values of nodes farther away in the graph are propagated to the parents and children every iteration. The algorithm stops when the difference between iteration falls below a threshold ϵ.

The authority and hubs value for node 1 using HITS can be more easily changed by adding outlinks or inlinks respectively. By adding outlinks, we can increase its hub score, whereas having more nodes link to 1 increases its authority score.

PageRank

The PageRank algorithm became one of the most recognized link analysis algorithms due to its use in Google. PageRank does not have the idea of hubs and authorities; rather, it defines PageRank as a function of the parent nodes' PageRank. Specifically, for a page p, its PageRank r is defined as

$$r(p) = \sum_{w \in pa[p]} \frac{r(w)}{\|ch[p]\|} \tag{3}$$

We first initialize a matrix `M` by dividing each node's row by the number of children. Thus if a node i has 4 outgoing links, all the non-zero values of row i will be $\frac{1}{4}$. We then initialize $\mathbf{x}$ as the vector containing the PageRank values for each node and normalize it.

The PageRank for a page p_i is given by

$$PR(p_i) = \frac{d}{n} + (1-d) \sum_{l_{j,i} \in M} \frac{PR(P_j)}{Out(p_j)} \tag{4}$$

where $l_{i,j}$ indicates if there is a link between pages i and j. Matrix `M` already contains information about links of each node, and $\mathbf{x}$ contains the PageRank of the previous iteration. Thus Equation 4 can be interpreted as taking the product of `M` and $\mathbf{x}$. The algorithm iterates until the difference in $\mathbf{x}$ between iterations drops beneath a threhsold, which we interpret as converging. $\mathbf{x}$ is the set of eigenvalues.

SimRank

This algorithm measures the relation between each pair of nodes. For each pair of nodes i, j we loop over the array and calculate their SimRank value for many operations. This value depends on the inlinks and outlinks of each node. In our code we calculate the SimRank value between each pair of

nodes for a certain amount of iterations. The `sim_scores` matrix is first set as the identity matrix, since each node is fully related to itself.

Afterwards, for each pair of nodes, we get s for each pair of their parents and store the total sum as $s(i, j)$.

Discussion

Implementation

The inputs to each of these algorithms is a graph of nodes. In `graph.py` the `Graph` class stores the parents and children for each node. It also provides some getters as well as calculating the adjacency matrix when needed. This graph class gets initiated when we first read the input file and gets passed to each algorithm.

Each of the algorithms quantifies importance differently. For HITS, we keep two vectors, one for the hubness and authorities, respectively. However, both PageRank and SimRank keep a two dimensional matrix for their calcualation. In our estimation, HITS utilizes less memory since the growth in array sizes is linear with an increase in nodes.

Time efficiency is a different matter. For SimRank, we use a fixed number of iteraitons whereas for the other algorithms we set a threshold for termination. This way SimRank will have a more predictable run time, but may not be as complete a description of the webpages as the other algorithms. HITS and PageRank both set a certain threhsold depending on the change between consecutive iterations. PageRank calculates a matrix multiplication every iteration and normalizes $\mathbf{x}$ to measure the difference from the previous iteration. HITS and SimRank on the other hand, operate on the columns and rows at one time. However, HITS requires an operation on the hubs and authorities matrix every iteration.

We measure the iterations taken for convergence of both PageRank and HITS, given below.

	graph_1	graph_2	graph_3	graph_4	graph_5	graph_6
PageRank	38	7	7	6	4	4
HITS	2	2	2	2	2	2

From the graph, PageRank takes many iterations to converge from `graph_1`. That graph is one where nodes are connected in a linear fashion (1→2,...,5 →6). The threshold is set at .005 for all tests, so something in the graph structure is causing the iterations to incrase. Increasing the threshold to .05

results in only an average of 4 iterations for the same graph. We believe that due to the similarity in score for each node, the values converge at a much slower rate.

By adding extra nodes and edges between existing nodes, we can improve the score of certain nodes. We take node 1 from graphs 1-3 as an example. By increasing the incoming connections to node 1 from certain other nodes, we can increase its PageRank score. In `graph_1.txt`, we noticed that the middle nodes have a higher value. If we add edges between say, nodes 3 and 4 to 1, we can slightly increase node 1's PageRank.

The method for the other files is similar. In `graph_2.txt` we can erase node 2's connection to node 3 and have nodes 4 and 5 have an edge to node 1 to slightly increase its PageRank. In `graph_3.txt`, increasing node 1's PageRank is harder due to the strong connections of each node. By having node 1 point to every other node and deleting most of the nodes other nodes have with each other we can slightly increase node 1's PageRank.

Further Questions

Link analysis algorithms have been an important way to find relevant content on the web. Nevertheless, there are trade-offs and limitations to these kinds of algorithms. The clearest limitation is the struggle that new websites face to gain a high PageRank or SimRank value when not many authoritative sites link to it yet. Despite the "true" relevance or authoritativeness of a new website (e.g. a new international organization), it would take some time for other sites to link to it before it can get a more representative score on these algorithms. In HITS, a new site might appear as a hub, since it might link to many auhtoritative sites without many sites pointing to it at all. New sites would face an uphill battle to gain the score that might best reflect their contents. Throughout the years many schemes to increase a page's appearance on search engine results have appeared, with people creating fake sites pointing to each other to increase the overall visibility and then selling links to these authoritative fake sites. Despite the potential for exploits, these algorithms have yet provided a better browsing experience for countless people throughout the past few decades, so their usefulness has been clearly shown.

One potential method to avoid the problem for new sites could be estimating its initial score for any algorithm with some machine learning model. Given a page's inlinks and outlinks we could create a model that estimates its score based on previously seen pages. This model could run on a given page when we index it for the first time, and could also run periodically to remain up-to-date. Training data for such a model would be plentiful given the variety of pages on the internet.

Whether these algorithms really find the most important pages poses an interesting question. First, we have to define "important" in the context of the internet. Do we mean popular? Scientific or medical? Sites that hold "important" information might not be the most popular, so a clear definition is

important. Assuming we mean popular sites are important, it depends on the number of inlinks and outlinks. Further assuming that popular websites have more incoming links than outgoing, auhtoritative websites would be the most popular, which HITS can easily provide.

Another consideration is the computation time. Given the constantly changing nature of the internet, having to run the algorithms constantly on such big matrices quickly becomes expensive. For a company looking to use these algorithms, the size of the graph could play a role in choosing one algorithm over another.

Conclusion

This assignment focused on implementing some of the most important link analysis algorithms, such as HITS, PageRank, and SimRank. All of these algorithms have contributed to the way we naviagate around networks such as the internet, although they can be used for other networks, such as citation networks. All of these algorithms view the number of incoming and outgoing links to determine the importance of a page. By implementing these algorithms on different graphs, we were able to see how the graph structure influences the behavior of the algorithms and how each page's scores can be increased.

Furthermore, we explored some of the downsides of these algorithms and the challenges they face during deployment. The modern web, with so many pages and connections, presents a large challenge in implementing these algorithms. With newer techniques, the size of the internet might not be the issue it was before, since a model can estimate the score of a page based on the pages it has seen previously.

With the internet and other networks playing an increasing role in our lives, finding the most relevant information is of great importance. However, before we do so we need to recognize important information. The connections we can draw from seeing connections in graphs can lead to even more insights about how we categorize data.