Der PageRank Suchalgorithmus

*Mishkin Faustini*

**Kurzbeschreibung**

Dieses Papier dient dazu, verschiedene Implementierungen des PageRank-Algorithmus aufzuzeigen, um die Vor-und Nachteile der einzelnen Ansätze im Hinblick auf Skalierbarkeit und die Speicherplatz/Zeit-Kosten der Ermittlung zu zeigen. Es gibt mehrere Möglichkeiten, den PageRank zu berechnen; iterativ, algebraisch, mit inverser Iteration oder der Potenzmethode. Die Beweggründe für die einzelnen Verfahren werden diskutiert, jeder Algorithmus wird zusammen mit allen Implementierungsproblemen gezeigt und die Versuchsergebnisse werden diskutiert.

**Einführung/Beweggründe**

Einer der wichtigsten Algorithmen der modernen EDV ist PageRank, der Algorithmus der von Larry Page und Sergey Brin als Teil eines Forschungsprojektes an der Universität Stanford entwickelt wurde. Mittlerweile wird PageRank als Rückgrat von Googles Suchmaschinen-Operationen verwendet, was der wohl nützlichste Dienst im Internet ist.

**Grundlagen und Theorie**

**Grundformulierung**

PageRank ist eine Wahrscheinlichkeitsverteilung, die verwendet wird, um die Wahrscheinlichkeit, dass eine Person zufällig eine bestimmte Webseite besuchen würde abzubilden. Die Idee ist, sich einen zufälligen Web-Surfer vorzustellen, der eine Seite besucht und zufällig auf Links klickt um andere Seiten zu, dann zufällig auf eine neue Seite geht, und den Prozess wiederholt. Die Wahrscheinlichkeit, dass der Surfer eine bestimmte Seite besucht ist der PageRank. In dieser Hinsicht können wir diesen Prozess als eine Markov-Kette auffassen, wo die Stati Webseiten und die Übergänge die Links zwischen den Seiten sind. Also, wie sieht die Wahrscheinlichkeitsverteilung aus? Eine Wahrscheinlichkeit kann als Zahlenwert im Bereich von 0 bis 1.0, ausgedrückt werden. Somit ist eine Seite mit dem Wert 0.7 gleichbedeutend mit einer Chance von 70%, dass ein Benutzer die Seite zufällig besuchen würde.

A

C

D

B

0.25

0.25

0.25

0.25

**Figure 1 – Set W: Our four pages and their links**

Suppose that we have a set of pages W that we wish to run through the PageRank algorithm. To begin we assume that each page has equal probability of being chosen on a random surfer walk. Thus, each page begins with a PageRank of 0.25. If we name the pages A, B, C and D and pages B, C and D link to page A then,

Wobei PR(X) der PageRank(x) und L(X) die Anzahl der ausgehenden Links auf Seite X ist. So haben wir im oben erwähnten einfachen Fall:

Allgemeiner kann dies ausgedrückt werden als:

**Dampening Factor**

PageRank has included a factor to simulate the odds that a user (or our random web surfer) stops clicking links. Thus there is a chance that at any given page the user will stop clicking. The way we can simulate this factor is by using a *dampening factor.*

The dampening factor that has been tried and tested in numerous studies happens to be about 0.85. This slightly changes our formula:

Wobei *n* die Gesamtanzahl der Seiten im System , und *d* der Dämpfungsfaktor ist.

**Matrix-Bildung und Berechnung**

Wenn wir eine Matrix G haben, welche die Nachbarschaftsmatrix ist welche die Verbindungen von allen unseren Webseiten darstellt, dann können wir die Anzahl der eingehenden und ausgehenden Links für eine bestimmte Webseite i oder j in unserer Matrix bestimmen durch das Formulieren der zwei Gleichungen r*i* = *i*g*ij*,c*j* = *i*g*ij*

To solve we must first convert our equation into a modified adjacency matrix, A, and PageRank values will then be the dominant eigenvectors of our matrix. A is an n-by-n matrix that:

Because we have a matrix which holds the transition probability between pages and the sum of its columns equal one we can conclude the following by *Perron-Frobenius theorem*

Where X is a unique matrix if we have a scaling factor such that, *i*x*i* = 1 The resulting solution to *x* is the PageRank calculation.

**Algorithmus**

Es sei darauf hingewiesen, dass es verschiedene Algorithmen gibt, um das PageRank-Problem zu lösen. Hier werden wir auf der Potenzmethode und Inverse Iteration eingehen. Das folgende sind MATLAB Implementierungen der PageRank-Algorithmus-Berechnung:

***Pseudo-code***

**Potenzmethode**

% Eliminate any self-referential links

% c = out-degree, r = in-degree

% Scale column sums to be 1 (or 0 where there are no out links).

% Calculate the following..

% G = p\*G\*D;

% x = initial equal link value (1/n)

% xprev = 1;

% while sum(abs(xprev-x)) > 0.001

% xprev = x;

% x = G\*x + e\*(z\*x);

% end

% Normalize so that sum(x) == 1.

**Inverse Iteration**

% Eliminate any self-referential links

% c = out-degree, r = in-degree

% Scale column sums to be 1 (or 0 where there are no out links).

% Calculate delta = (1-p)/n

% Calculate A = p\*g\*D + delta

% solve e =(I-A)\*x

% Normalize so that sum(x) == 1.

*Note: See code files at end of paper for implementation:* pagerank\_powermethod.m, pagerank\_inverseIteration.m, PageRank.cs

**Umsetzungsfragen**

One of the largest implementation issues is the sheer size of the datasets on which PageRank is being calculated. Companies like Google cannot directly use matrix solvers to compute PageRank because their datasets are much too large. Instead a method such as the Power Method is used where a broad sweep over the database can be calculated in several passes.

A second issue with the PageRank algorithm is that it favors older more well established pages over newer pages. If a new page enters the system it will have relatively few outbound and inbound links as compared to a site that has a number of links in and out of it.

**Versuchsresultate**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** | **F** |
| **A** |  |  | X |  | X |  |
| **B** | X |  |  |  |  |  |
| **C** |  | X |  | X |  |  |
| **D** |  | X |  |  | X | X |
| **E** |  |  |  |  |  |  |
| **F** |  |  |  |  |  |  |

A

B

E

F

D

C

**Figure 2 - PageRank Example Scenario**

**Corresponding link adjacency matrix**

Für dieses Beispiel werden wir an eine Grund Link-Struktur betrachten und versuchen, den PageRank mit MATLAB zu berechnen.

|  |  |  |  |
| --- | --- | --- | --- |
| **Page** | **PageRank** | **In** | **Out** |
| **A** | 0.3210 | 2 | 2 |
| **C** | 0.1705 | 1 | 2 |
| **D** | 0.1066 | 1 | 3 |
| **B** | 0.1368 | 2 | 1 |
| **F** | 0.0643 | 1 | 0 |
| **E** | 0.2007 | 2 | 1 |

After running pagerank\_powermethod(U,G) using our adjacency matrix we have the above PageRank results. The results show that page F which has little connection and little chance of being clicked on has the lowest PageRank while A has the highest PageRank.

Thus we can see that the PageRank is being calculated correctly.

The following MATLAB code can reconstruct this scenario:

>> i = [2 6 3 4 4 5 6 1 1];

>> U = {'A','C','D','B','F','E'};

>> j = [1 1 2 2 3 3 3 4 6];

>> n=6;

>> G = sparse(i,j,1,n,n);

>> pagerank\_powermethod(U,G)

**Concluding Remarks**

There are many tweaks that can be done to further enhance PageRank’s performance in terms of calculation efficiency and results. The basic PageRank algorithm is an interesting algorithm to learn. The algorithm itself is far simpler than one may think but the difficulty seems to be in calculating the PageRank efficiently for massive datasets.

A deeper understanding of the inner workings of how to efficiently compute PageRank for massive datasets would be a natural progression from this point.

**Acknowledgements**

Special thanks to professor Professor Zhaojun Bai and the University of California at Davis to whom has introduced me to Scientific Computation and Cleve Moler for his very informative book Numerical Computing with MATLAB for which I have learned from all quarter.

**References**

Numerical Computing with MATLAB by Cleve Moler

http://www.mathworks.com/moler/chapters.html

L. Page, S. Brin, R. Motwani and T. Winograd “The PageRank Citation Ranking: Bringing Order to the Web”, Stanford Digital Library working paper SIDL-WP-1999-0120 (version of 11/11/1999). See: http://www-diglib.stanford.edu/cgibin/get/SIDL-WP-1999-0120

A. Arasu, J. Novak, A. Tomkins and J. Tomlin, “PageRank Computation and the Structure of the Web: Experiments and Algorithms”, Technical Report, IBM Almaden Research Center, Nov. 2001.

<http://wenku.baidu.com/view/a1981d93daef5ef7ba0d3c6e.html>