

Web-Informationssysteme, WS 2009/10

Seminar-Übung 4: Map/Reduce, PageRank

Besprechung am Mi 18.11.2009

— SÜ-4.1 —

Was Ihr wissen müßt über “Map/Reduce”

Wiederholung

- Welches **Problem** versucht Map/Reduce zu lösen?
- Erklären Sie die Herkunft des **Namens** Map/Reduce!
- Beschreiben Sie den **Datenfluss** in einem Map/Reduce System mit mehreren Workern!
- Beschreiben Sie, wie der **PageRank** eines gegebenen Web-Graphen mit Hilfe von Map/Reduce berechnet werden kann, insbesondere wie die Ein- und Ausgabedaten der verschiedenen Map- bzw. Reduce-Tasks aussehen.
- Geben Sie Beispiele für Probleme, die sich schlecht mit Hilfe von Map/Reduce lösen lassen! Können Sie eine Gemeinsamkeit dieser Probleme finden?

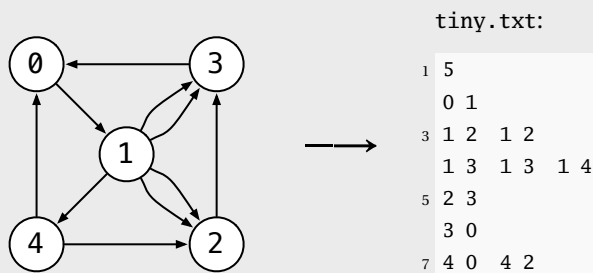
— SÜ-4.2 —

Implementierung PageRank

Programmierung

In dieser Aufgabe sollen Sie den in der Vorlesung vorgestellten Algorithmus zur *iterativen* Berechnung des **PageRanks** implementieren. Dabei gehen wir davon aus, dass nur Web-Graphen ohne *dangling nodes*, also ohne Knoten, die keine ausgehenden Kanten haben, vorkommen.

Als Eingabe des Algorithmus soll eine Repräsentation des Webs in der folgenden Form dienen:



Die erste Zeile gibt die Anzahl der Seiten N , die folgenden N Zeilen Paare von Webseiten zur Repräsentation von Links (z.B. 4 0 4 2 repräsentiert einen Link von Seite 4 auf Seite 0 und einen Link von Seite 4 auf Seite 2). Man beachte, dass mehrere Links zwischen denselben Seiten zugelassen sind und behandelt werden sollen.

Wie in der Vorlesung zerlegen wir den Algorithmus in zwei Teile:

1. *Berechnung der Google-Matrix*: Im ersten Schritt soll die Google-Matrix *mit random leap* wie in der Vorlesung berechnet werden.

EXAMPLE GOOGLE-MATRIX:

Für obigen Web-Graphen ergibt sich die Google-Matrix G (mit *random leap probability* $\alpha = 0.1$) wie folgt:

$$\begin{aligned}
 A^T &= \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, & D &= \begin{pmatrix} 1 \\ 5 \\ 1 \\ 1 \\ 2 \end{pmatrix}, & H &= \begin{pmatrix} 0 & 0 & 0 & 1 & 0.5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0.4 & 0 & 0 & 0.5 \\ 0 & 0.4 & 1 & 0 & 0 \\ 0 & 0.2 & 0 & 0 & 0 \end{pmatrix}, \\
 G &= \begin{pmatrix} 0 & 0 & 0 & 0.90 & 0.45 \\ 0.90 & 0 & 0 & 0 & 0 \\ 0 & 0.36 & 0 & 0 & 0.45 \\ 0 & 0.36 & 0.90 & 0 & 0 \\ 0 & 0.18 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \end{pmatrix} = \begin{pmatrix} 0.02 & 0.02 & 0.02 & 0.92 & 0.47 \\ 0.92 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.38 & 0.02 & 0.02 & 0.47 \\ 0.02 & 0.38 & 0.92 & 0.02 & 0.02 \\ 0.02 & 0.20 & 0.02 & 0.02 & 0.02 \end{pmatrix}
 \end{aligned}$$

2. *Iterative Berechnung des PageRanks*: Mit der soeben berechneten Google-Matrix G und einem beliebigen Startvector (z.B. $\mathbf{p}^0 = (1, 0, \dots, 0)^T$) wird eine Näherung des PageRank-Vektors iterativ berechnet durch die

Formel:

$$\mathbf{p}^{k+1} = G\mathbf{p}^k$$

Der Algorithmus soll nach einer vorgegebenen Anzahl von Iteration abbrechen und die erreichte Approximation für jede der N Seiten ausgeben.

Das Programm soll die Anzahl der Iteration und den Namen der Datei, die eine Repräsentation des Web-Graphens wie oben enthält, von der Kommandozeile lesen.

Bei der Eingabe

```
java t04.t04CLIDriver 30 tiny.txt
```

soll das Programm beispielsweise die folgende Ausgabe liefern:

```
1 Transition Matrix -----
   Dimensions: 5 5
3   0.02000 0.02000 0.02000 0.92000 0.47000
   0.92000 0.02000 0.02000 0.02000 0.02000
5   0.02000 0.38000 0.02000 0.02000 0.47000
   0.02000 0.38000 0.92000 0.02000 0.02000
7   0.02000 0.20000 0.02000 0.02000 0.02000
   PageRank vector (approx.) -----
9   after 1 iterations:    0.02000 0.92000 0.02000 0.02000 0.02000
   after 2 iterations:    0.04700 0.03800 0.36020 0.36920 0.18560
11  after 3 iterations:    0.43580 0.06230 0.11720 0.35786 0.02684
   after 4 iterations:    0.35415 0.41222 0.05451 0.14791 0.03121
13  after 5 iterations:    0.16716 0.33874 0.18245 0.21745 0.09420
   after 6 iterations:    0.25810 0.17045 0.18434 0.30615 0.08097
15  after 7 iterations:    0.33197 0.25229 0.11780 0.24726 0.05068
   after 8 iterations:    0.26534 0.31877 0.13363 0.21684 0.06541
17  after 9 iterations:    0.24459 0.25881 0.16419 0.25503 0.07738
   after 10 iterations:   0.28434 0.24013 0.14799 0.26095 0.06659
19  after 11 iterations:   0.28481 0.27591 0.13641 0.23964 0.06322
   after 12 iterations:   0.26413 0.27633 0.14778 0.24210 0.06966
21  after 13 iterations:   0.26924 0.25771 0.15083 0.25248 0.06974
   after 14 iterations:   0.27862 0.26231 0.14416 0.24852 0.06639
23  after 15 iterations:   0.27355 0.27075 0.14431 0.24418 0.06722
   after 16 iterations:   0.27001 0.26619 0.14772 0.24735 0.06874
25  after 17 iterations:   0.27354 0.26301 0.14676 0.24878 0.06791
   after 18 iterations:   0.27446 0.26619 0.14524 0.24677 0.06734
27  after 19 iterations:   0.27239 0.26701 0.14613 0.24655 0.06791
   after 20 iterations:   0.27245 0.26515 0.14669 0.24764 0.06806
29  after 21 iterations:   0.27351 0.26521 0.14608 0.24747 0.06773
   after 22 iterations:   0.27320 0.26616 0.14595 0.24695 0.06774
31  after 23 iterations:   0.27274 0.26588 0.14630 0.24717 0.06791
   after 24 iterations:   0.27302 0.26546 0.14628 0.24739 0.06786
33  after 25 iterations:   0.27318 0.26571 0.14610 0.24722 0.06778
   after 26 iterations:   0.27300 0.26587 0.14616 0.24715 0.06783
35  after 27 iterations:   0.27296 0.26570 0.14623 0.24726 0.06786
   after 28 iterations:   0.27306 0.26566 0.14619 0.24726 0.06783
37  after 29 iterations:   0.27306 0.26576 0.14616 0.24721 0.06782
   after 30 iterations:   0.27300 0.26575 0.14619 0.24722 0.06784
39 -- final page rank vector (after 30 iterations):
   0.27300 0.26575 0.14619 0.24722 0.06784
```

Hinweis: In den Beilagen zu diesem Übungsblatt finden Sie zwei Eingaben (**tiny.txt** und **medium.txt**) zum Testen Ihrer Implementation. Wie immer finden Sie auch .java Dateien, die Sie als Ausgangspunkt für eine eigenen Lösung verwenden können, indem Sie die mit @TODO markierten Teile ersetzen.