Data Mining
Homework 2

Yao Song
301266041

November 25, 2014

MapReduce

2.3.4

mapper: The mapper will read in the bag and the query constraint. The mapper outputs each item which passes the constraint. The output key-value pair format is [(first name, last name),1], where key is a tuple containing first name and last name.

reducer: The reducer takes all the output from the mapper. The reducer then combines and counts all the items with the same last name. Then the reducer will output the key-value pair [(first name, last name), count] for each distinct last name.

a sample input like the following:
Tom Lee
John Smith
Tom Lee
James Bond
Tom White
Tom White
Tom Blake
with query ”first name = Tom” will output something like this:
(Tom , Blake) 1
(Tom , White) 2
(Tom , Lee) 2

Python code to implement this process appended at last.

Link Analysis

5.4.2

Trust Rank

Assume $\beta = 0.8$. Since only node B is reliable, the teleport set is $S = \{B\}$ and we have $\boldsymbol{e}_s = [0, 1, 0, 0]$. Then the vector $(1 - \beta)\boldsymbol{e}_s/|S|$ has 0.2 for only second component and zero for other parts. The iteration equation can be written as:

$$
\begin{aligned}
v' &= \beta M v + (1-\beta)\boldsymbol{e}_s/|S| \qquad (1) \\
&= \begin{bmatrix} 0 & \frac{2}{5} & \frac{4}{5} & 0 \\ \frac{4}{15} & 0 & 0 & \frac{2}{5} \\ \frac{4}{15} & 0 & 0 & \frac{2}{5} \\ \frac{4}{15} & \frac{2}{5} & 0 & 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ \frac{1}{5} \\ 0 \\ 0 \end{bmatrix} \qquad (2)
\end{aligned}
$$

After calculation, the trust rank is $t = [\frac{198}{735}, \frac{263}{735}, \frac{116}{735}, \frac{158}{735}]$

Spam Mass

The page rank as given in the book is $r = [\frac{3}{9}, \frac{2}{9}, \frac{2}{9}, \frac{2}{9}]$.

So the spam mass is given by the formula $\frac{r-t}{r}$ is $[\frac{94}{490}, \frac{-299}{490}, \frac{142}{490}, \frac{16}{490}]$. A list of these vectors is given below:

Node	Page Rank	Trust Rank	Spam Mass
A	$\frac{3}{9}$	$\frac{198}{735}$	$\frac{94}{490}$
B	$\frac{2}{9}$	$\frac{263}{735}$	$\frac{-299}{490}$
C	$\frac{2}{9}$	$\frac{116}{735}$	$\frac{142}{490}$
D	$\frac{2}{9}$	$\frac{158}{735}$	$\frac{16}{490}$

5.5.1

The link matrix of Fig. 5.1 in the book is given below:

$$L = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \tag{3}$$

After calculation, the limits of a and h are:

$$a = \begin{bmatrix} 0.289 \\ 1.000 \\ 1.000 \\ 0.813 \end{bmatrix}, h = \begin{bmatrix} 1.000 \\ 0.392 \\ 0.103 \\ 0.711 \end{bmatrix}. \tag{4}$$

Recommender System

9.2.1

a

The cosine of the angel between a and b is :

$$\frac{8.2008 + 160000\alpha^2 + 24\beta^2}{\sqrt{9.3636 + 250000\alpha^2 + 36\beta^2}\sqrt{7.1824 + 102400\alpha^2 + 16\beta^2}} \tag{5}$$

The cosine of the angel between a and c is :

$$\frac{8.9352 + 320000\alpha^2 + 36\beta^2}{\sqrt{9.3636 + 250000\alpha^2 + 36\beta^2}\sqrt{8.5264 + 409600\alpha^2 + 36\beta^2}} \tag{6}$$

The cosine of the angel between b and c is :

$$\frac{7.8256 + 204800\alpha^2 + 24\beta^2}{\sqrt{7.1824 + 102400\alpha^2 + 16\beta^2}\sqrt{8.5264 + 409600\alpha^2 + 36\beta^2}} \tag{7}$$

b

if $\alpha = 1$ and $\beta = 1$, the data is listed below:

x,y	a,b	a,c	b,c
$\cos(\theta)$	0.999997333284	0.999995343121	0.999987853375
$\theta(degrees)$	0.140345459308	0.181185239071	0.292152614283

c

if $\alpha = 0.01$ and $\beta = 0.5$, the data is listed below:

x,y	a,b	a,c	b,c
$\cos(\theta)$	0.990881500541	0.991554714333	0.969177921994
$\theta(degrees)$	7.74400574467	7.45370628496	15.8939985811

d

In this case, $\alpha = 3.0/1460$ and $\beta = 3/16$, the data is listed below:

x,y	a,b	a,c	b,c
$\cos(\theta)$	0.994390403954	0.995613999454	0.982246918405
$\theta(degrees)$	6.0718677355	5.37067531447	10.814438368

9.2.3

a

The average rating of the user is $(4 + 2 + 5)/3 = 3.67$. So the normalized rating of the user is
$A : 0.33, B : -1.67, C : 1.33$

b

The user profile is computed as follows:
for the processor speed: (3.06*4+2.68*2+2.92*5)/(4+2+5) = 2.92
for the disk size: (500*4+320*2+640*5)/(4+2+5) = 530
for the main memory size: (6*4+4*2+6*5)/(4+2+5) = 5
So the user profile is (2.92, 530, 5)

Q1

The cosine similarity between Alice and other users are listed blow:

	u1	u2	u3	u4
similarity	0.76	0.99	0.91	0.31

With the threshold 0.75, we know the neighbor are user1 and user2. We do not need to consider user3 because it does not rate item 5.

Thus the estimated rating of Alice on item 5 would be
$(5 + 4 + 4)/3.0 + (0.76 * (3 - 11/4.0) + 0.99 * (5 - 16/4.0))/(0.76 + 0.99) \approx 5$
So Alice's predicated rating on item 5 is 5.

Finding Similar Items

3.3.3

a

The minhash signature is listed in the following table.

	s_1	s_2	s_3	s_4
h_1	5	1	1	1
h_2	2	2	2	2
h_3	0	1	4	0

b

h_3 is a true permutation as shown in the following table.

Element	h_1	h_2	h_3
0	1	2	2
1	3	5	1
2	5	2	0
3	1	5	5
4	3	2	4
5	5	5	3

c

The comparison of estimated similarities and Jaccard Similarities are listed below.

	esimated sim	Jaccard sim
s_1, s_2	1/3	2/6
s_1, s_3	1/3	2/6
s_1, s_4	2/3	3/6
s_2, s_3	2/3	2/6
s_2, s_4	2/3	3/6
s_3, s_4	2/3	3/6

3.4.2

The exact and estimated value of s for each case are listed in the table below:

	exact	estimated
(3,10)	0.41	0.46
(6,20)	0.57	0.61
(5,50)	0.42	0.46

We can observe that the exact value of s is always smaller than the corresponding estimated one.

The code can be run by calling

```
hadoop jar share/hadoop/tools/lib/hadoop-streaming-2.5.1.jar
-file mapper.py -mapper 'mapper.py James'
-file reducer.py -reducer reducer.py
-input inputfolder  -output outputfolder
```

Mapper:

```
import sys

myarg = sys.argv[1]

for line in sys.stdin:
	line = line.strip()
	first, last = line.split()
	if first == myarg:
		print '%s \t %s \t 1' % (first, last)
```

Reducer:

```
#!/usr/bin/env python

from operator import itemgetter
import sys

current_last = None
current_count = 0
last = None

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    first, last, count = line.split('\t')

    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        print 'wrong count number!!!!!!!!!!!!!!!!!!!!!!!!'
        break

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_last == last:
        current_count += count
    else:
        if current_last:
            # write result to STDOUT
            print '(%s,%s)\t%s' % (first, current_last, current_count)
        current_count = count
        current_last = last

# do not forget to output the last word if needed!
if current_last == last:
    print '(%s,%s)\t%s' % (first, current_last, current_count)
```