ComS 535x:
Project Report #3

Due on Apr 8, 2015

Instructor: Professor Aduri, Pavankumar

Yuanyuan Tang, Chenguang He

Contents

1 WikiCrawler **3**

1 High level description/pseudo code of your crawling algorithm. In your pseudo code you can use sentences such as ”request page from server”, ”if the page has key words”. Ignore robots.txt and handling network errors in pseudo code 3

2 ”Data structures that you used to maintain ”visited” and Q in the crawling algorithm.” 4

2 PageRank **5**

1 Data structure you used to represent the graph. 5

2 Pseudo code for the page rank algorithm 5

3 WikiTennisRanker **7**

4 MyWikiRanker **7**

WikiCrawler

1 High level description/pseudo code of your crawling algorithm. In your pseudo code you can use sentences such as "request page from server", "if the page has key words". Ignore robots.txt and handling network errors in pseudo code

Algorithm 1 WikiCrawler

```
procedure WIKICRAWLER(seedUrl, keywords)
    trash ← ∅
    visited ← ∅
    robots ← all disallow url in seedUrl's robots.txt files.
    queue ← ∅
    max ← 0
    numOfEdges ← 0
    RequestToWiki ← 0
    queue ← download(seedUrl)
    while queue is not empty and visited size is less than max do
        p ← queue.poll()
        subs gets all pages that match the restrictions and are not in robot and contains all keywords. increase
    max by the visit pages.
        for each page in subs do
            if page is not in visited and visited size <= max then
                visited add page
                queue add page
            end if
        end for
    end while
    for each page in visited do
        extract links relations in page.
        print relations.
    end for
end procedure
```

2 ”Data structures that you used to maintain ”visited” and Q in the crawling algorithm.”

I use hashset to maintains visited and use queue to maintains Q because :

1. both visited and Q are NOT needed to maintain elements in order. Therefore Hashset is the fastest way to get element.
2. In order to implement hash, i design the class Page with hashcode and equal method
3. Because we do more accesses than insertion, we use hashset to get the best performance.
4. Because the nodes we need to test are relatively small, hashset is better than BloomFilter which result in over design.
5. Using Queue to maintains Q is relatively obvious choice.

PageRank

1 Data structure you used to represent the graph.

I used $HashMap < String, HashSet < String >>$ to represent the graph. The name of each vertex was stored as key value in the HashMap. The name of all the vertices which was connected from the vertex was stored in a $HashSet < String >$. In this way, the graph was represent in the form of adjacency list.

2 Pseudo code for the page rank algorithm

Algorithm 2 calculate Page Rank(graph, epsilon)

procedure CALCULATE PAGE RANK($graph, \epsilon$)
  Initialize a vector P_0
  Initialize a vector P_1
  Set P_0 to the uniform probability vector $[\frac{1}{N}, ..., \frac{1}{N}]$.
  $P_1 \leftarrow P_0$
  n$\leftarrow$ 0
  converged $\leftarrow$ false
  while not converged **do**
    compute Rank1 $(graph, P_0)$.
    if $Norm(P_1, P_0) \leq \epsilon$ **then**
      converged $\leftarrow$ true
    end if
    n++;
    $P_0 \leftarrow P_1$
  end while
  return P_1
end procedure

Algorithm 3 compute Rank1 from Rank0

```
procedure COMPUTE RANK1(graph, Rank_0)
    β ← 0.8
    N ← number of vertices in the graph
    Set Rank_1 to the uniform probability vector [(1−β)/N, ..., (1−β)/N]
    for p ← every page in graph do
        Q ← all the pages that are linked from p
        if |Q| ≠ 0 then
            for q ← all pages in Q do
                Rank_1(q) = Rank_1(q) + β Rank_0(p)/|Q|
            end for
        end if
        if |Q| = 0 then
            for q ← all pages in Q do
                Rank_1(q) = Rank_1(q) + β Rank_0(p)/N
            end for
        end if
    end for
    return Rank_1
end procedure
```

WikiTennisCrawler

1 Time taken by your crawler, Number of nodes and edges in the graph constructed.

Number of Vertices is 1000.
Number of Requst to Wiki: 6087 requests
Numberof Edges added: 96402 Edges
Time used:1940.671 seconds (including sleeping)

MyWikiCrawler

1 Root url, Time taken, number of nodes and edge in the graph

seed Url: /wiki/basketball
Keywords: basketball nba
Number of Requst to Wiki: 5763 requests
Number of Edges added: 103051 Edges
Time used:992.504 seconds (including sleeping)

WikiTennisRanker

The number of vertices in the graph is 1000.
when $\epsilon = 0.1$,
The number of iterations is 2.
The highest rank page is: /wiki/France
The highest in degree page is /wiki/Australia
The highest out degree page is /wiki/Rod_Laver
The similarity between top100pagerank and top100inDegree is 0.8018018018018018
The similarity between top100pagerank and top100outDegree is 0.42857142857142855
The similarity between top100inDegree and top100outDegree is 0.45985401459854014
when $\epsilon = 0.05$,
The number of iterations is 3.
The highest rank page is: /wiki/France
The highest in degree page is /wiki/Australia
The highest out degree page is /wiki/Rod_Laver
The similarity between top100pagerank and top100inDegree is 0.7391304347826086
The similarity between top100pagerank and top100outDegree is 0.42857142857142855
The similarity between top100inDegree and top100outDegree is 0.45985401459854014

Since top100inDegree and top100outDegree contain the same terms no matter what ϵ we choose, the Jaccard similarity of those two sets would be the same. We also found that the larger the ϵ be, the fewer iterations we have. We also found that from Jaccard similarity, top100pagerank was much more similar with top100inDegree than with top100outDegree. This is consistent with the definition of page rank: the more pages link to a certain page, the higher the Page Rank of this page.

MyWikiRanker

the number of vertices in the graph is 1000
When $\epsilon = 0.1$
The number of iteration is 3.
The highest rank page is /wiki/Basketball
The highest in-degree page is /wiki/Basketball
The highest out-degree page is /wiki/Outline_of_basketball
The similarity between top100pagerank and top100inDegree is 0.47058823529411764
The similarity between top100pagerank and top100outDegree is 0.17647058823529413
The similarity between top100inDegree and top100outDegree is 0.4084507042253521

When $\epsilon = 0.05$
The number of iteration is 3.
The highest rank page is: /wiki/Basketball
The highest in degree page is /wiki/Basketball
The highest out degree page is /wiki/Outline_of_basketball
The similarity between top100pagerank and top100inDegree is 0.47058823529411764
The similarity between top100pagerank and top100outDegree is 0.17647058823529413
The similarity between top100inDegree and top100outDegree is 0.4084507042253521

We found that the highest page rank, in-degree page and out-degree page did not change with ϵ .