Answers

Part 2

We implemented the Random Surfer algorithm with the Damping factor according to the formula given in class. After building the graph we iterated over our vertex list until the PageRank change from one iteration to another was smaller than epsilon (a constant we sat to be 0.01) for all vertices.

Results for different Damping factors:

For Damping=0:

1. Cnn
2. Yahoo
3. Aol

For Damping = 0.1:

1. Cnn
2. Bbc
3. Nytimes

For Damping = 0.3:

1. Cnn
2. Bbc
3. Nytimes

Execution time (milliseconds): 60-200

For a Damping factor of 0.85 I tried initial PageRanks of 1,5 and 10 and the results were the same for all:

1. Cnn
2. Bbc
3. Nytimes

Part 4

1. We implemented the aggregation function as follows:

For each word in the Boolean expression, we have a sorted Vector of TF scores with correlation to the respective websites.

* If the OR operator is given, the top website score in each iteration is calculated by the Maximum between TF scores of all the given words in that website, which is then multiplied by that website’s PageRank.
* If the AND operator is given, the score is calculated by the average of TF scores of all the given words in that website, which is then multiplied by that website’s PageRank.

In general, when dealing with arbitrary Boolean combinations we will group the Boolean expression into subgroups according to the operator used – AND or OR, and apply the appropriate aggregation function for each smallest subgroup afterwards we will continue to use the appropriate function on the combinations of subgroups until we reach the full expression (a form of recursion if you will).

Our weight function is indeed monotone, since we are using Maximum and Average (which are monotone as discussed in class), and for each website whose TF values are the highest for all words, their average will also be highest among the averages of the other websites. Same goes for the Maximum function.

1. ”spain” AND ”germany”

1) cnn.txt 0.000025930471958717202

2) aol.txt 0.00000999642056755886

3) nydailynews.txt 0.000008257132926359892

”spain” OR ”germany”

1) cnn.txt 0.000028612934575136223

2) latimes.txt 0.000012499999999999999

3) aol.txt 0.000011424480648638699

”spain” OR ”cucumbers”

1) cnn.txt 0.00002324800934229818

2) aol.txt 0.00001713672097295805

3) latimes.txt 0.000012499999999999999

1. If we run the TA algorithm with a non-monotonous aggregation function, there is a chance we may return a website in the top-k relevant sites whose aggregated score does not belong in the top-k scores, while there was another site that should have been in the top-k list instead, but we did not reach it in our algorithm’s run since the threshold counter insured us no other website can have a better score than the threshold (the threshold is based on the assumption that the aggregation function is monotone).

Variables and design choices in our code

In the wdmAss class we implemented the following variables:

flag – an int variable indicating whether the Boolean operator in the TA option is AND(1) or OR(0).

\_initPR – initial pageRank value set to 1.

\_epsilon – a variable for determining when the change in PR values is negligible enough between iterations, set to 0.01.

\_damping – damping factor for the PR calculation (random surfer). Set to 0.3

balancePR – a variable for balancing the effect the PR has on the total score of a website in the TA algo. Set to 0.01