CS 498: Computational Advertising **Fall 2019**

Homework 3

Handed Out: October 4, 2019 *Due: October* 18, 2019 *11:59 pm*

1 General Instructions

- This assignment is due at 11:59 PM on the due date. We will be using Gradescope and Compass to collect this assignment. The homework MUST be submitted in pdf format on gradescope.

  Contact TAs if you face technical difficulties in submitting the assignment. We shall NOT accept any late submission!

  Please make sure to appropriately map/assign the pages of your submitted pdf to each sub-question listed in the homework outline. Handwritten answers are not acceptable. Name your pdf file as YourNetid-HW3.pdf
- For all questions, you need to explain the logic of your answer/result for every sub-part. A result/answer without any explanation will not receive any points.
- For the programming question of the assignment, you must submit your code in compass. Name your compass submission as YourNetid-LastName.zip.
- It is OK to discuss with your classmates and TAs regarding the methods, but it is NOT OK to work together or share code. Plagiarism is an academic violation to copy, to include text from other sources, including online sources, without proper citation. To get a better idea of what constitutes plagiarism, consult the CS Honor code (http://cs.illinois.edu/academics/honor-code) on academic integrity violations, including examples, and recommended penalties. There is a zero tolerance policy on academic integrity violations; Any student found to be violating this code will be subject to disciplinary action.
- Please use Piazza if you have questions about the homework. Also, feel free to send TAs emails and come to office hours.

Question 1 (3 points)

Explain the difference in auction prices between ads (assuming the ads are in the same category) shown on the search network and display network.

Question 2 (3 points)

How is a VCG auction for three items (say paintings), different from a VCG auction for three ad slots?

Question 3 (3 points)

Recent research shows that users sometimes prefer ads that did not win the auction for a slot. Explain possible reasons for this observation.

Question 4 (3 points)

Why does google ad sense manager offer advertisers an opportunity to automatically generate bids on phrases other than bid phrase chosen by the advertiser (advanced match)?

Question 5 (6 x 3 = 18 points)

In this question, you are required to implement the SimRank algorithm and SimRank with two forms of evidence weights in a programming language of your choice. In each case, your iterations must begin with user updates, alternating with ad similarity updates. We will use this `reference`[https://arxiv.org/pdf/0712.0499.pdf] for the algorithm details. We provide two test cases (a) and (b) as input. You can use the sample test case (a), which is smaller in size, to debug your code. The test case (b) is significantly larger and hence may require an efficient implementation. Thus, you should use partial sum sharing to speed up your implementation.

The input consists of weighted User-Ad links. The first line specifies the total number of links N, followed by N lines each containing three comma separated entries - U, A, S where U and A are integers representing User and Ad Ids, $0 \leq U \leq 1,000,000$ and $0 \leq A \leq 1,000,000$, and score S is a float score value for that link, $0.0 \leq S \leq 1,000.0$. The score is based on how fast the user responded to the Ad (a higher score denotes a greater proclivity).

This is then followed by a single line with two ids, Q_U and Q_A. You need to output the three most similar users to Q_U and the three most similar ads to Q_A with each of the variations of simrank. In case of a tie break, report all users (ads). A sample input looks like:

N
U_1, A_1, S_1
U_2, A_2, S_2
$\ldots$
U_N, A_N, S_N
Q_U, Q_A

Note U_1 is a user-id, while A_1 is the advertisement-id for a specific ad clicked by U_1 and S_1 is the link weight. Note that a given user can click more than one ad, meaning multiple lines could have the same user-id, but no two lines are identical. You are required to build the bipartite weighted User-Ad graph using the above links.

1. **Task One - Simple SimRank iterations**: Implement conventional SimRank and compute the similarities of users to each other and ads to other ads. You need to use the partial sums trick described in the links provided with the assignment description in your implementation. Initialize the algorithm with the usual procedure (similarity of a node to itself is 1 and 0 to all others), and perform $K = 10$ iterations of User and Ad-similarity updates (start with user similarity updates). The constants C_1 and C_2 are both set to 0.8. Let us call the similarity scores obtained after $K = 10$ iterations SIMPLE_SIMRANK_SCORES.
2. **Task Two - Incorporate evidence**: In section 7 of the reference, two forms are introduced for evidence scores (geometric in equation 7.3 and exponential in equation 7.4). You will apply each of these forms to the results obtained after 10 iterations of Simple SimRank and obtain the new similarity scores for users and ads. Let us call these two new sets of scores EVIDENCE_GEOMETRIC_SCORES and EVIDENCE_EXPONENTIAL_SCORES.

Your output should contain 6 lines:

Line 1 - 3 most similar users to Q_U with SIMPLE_SIMRANK_SCORES.
Line 2 - 3 most similar ads to Q_A with SIMPLE_SIMRANK_SCORES.
Line 3 - 3 most similar users to Q_U with EVIDENCE_GEOMETRIC_SCORES.
Line 4 - 3 most similar ads to Q_A with EVIDENCE_GEOMETRIC_SCORES.
Line 5 - 3 most similar users to Q_U with EVIDENCE_EXPONENTIAL_SCORES.
Line 6 - 3 most similar ads to Q_A with EVIDENCE_EXPONENTIAL_SCORES.

The sample test case (a) (with input and output) is provided in `sample_input.txt` and `sample_output.txt`. The large test case (b) on which your code (and results) will be evaluated, is present in the `input_b.txt`. You should submit your code on compass and the code outputs in the pdf submitted to gradescope.