HW 4 - PageRank

Total points: 100
Issued: 10/25/2018 Due: 11/15/2018

The goal of this programming assignment is to use PageRank to derive a ranking of words in a document based on their PageRank scores. The PageRank score of a word serves as an indicator of the importance of the word in the text.

For this assignment, you will use a collection consisting of titles and abstracts of research articles published in the WWW conference. The dataset is available for download from blackboard. Each document is POS tagged.

Tasks:

1. Write a program that loads each document into a word graph (either directed or undirected). In doing so, tokenize on whitespace, remove stopwords, and keep only the nouns and adjectives corresponding to {NN, NNS, NNP, NNPS, JJ}. Apply a stemmer on every word. For each candidate word, create a node in the graph. Add an edge in the graph between two words if they are adjacent in the original text.

2. Run PageRank on each word graph corresponding to each document in the collection as follows:

   - Initialization: $\mathbf{s} = [s(v_1), \cdots, s(v_n)] = [\frac{1}{n}, \cdots, \frac{1}{n}]$, where $n = |V|$.
   - Score nodes in a graph using their PageRank obtained by iteratively computing the equation below. Assume the weight w_{ij} of an edge (v_i, v_j) is calculated as the number of times the corresponding words w_i and w_j are adjacent in text.

   $$s(v_i) = \alpha \sum_{v_j \in Adj(v_i)} \frac{w_{ji}}{\sum_{v_k \in Adj(v_j)} w_{jk}} s(v_j) + (1 - \alpha) p_i, \tag{1}$$

   where α is a damping factor ($\alpha = 0.85$) and $p_i = \frac{1}{n}$.

3. After the PageRank convergence or a fixed number of iterations is reached (e.g., 10 iterations), form n-grams of length up to 3 (unigrams, bigrams and trigrams) from words adjacent in text and score n-grams or phrases using the sum of scores of individual words that comprise the phrase.

4. Calculate the MRR for the entire collection for top-k ranked n-grams or phrases, where k ranges from 1 to 10, using the gold-standard (author annotated data) provided in blackboard:

   - Mean reciprocal rank, MRR

   $$MRR = \frac{1}{|D|} \sum_{d=1}^{|D|} \frac{1}{r_d}$$

   r_d is the rank at which the first correct prediction was found for $d \in D$.

5. [Extra-credit - 30 points] Compare the MRR of the above PageRank algorithm with the MRR of a ranking of words based on their TF-IDF ranking scheme. Calculate the TF component from each document and the IDF component from the entire collection.

Submission instructions:

1. Write a README file including:
   - A detailed note about the functionality of each of the above programs,
   - Complete instructions on how to run them
   - MRR values for each k ranging from 1 to 10.
2. Make sure you include your name in each program and in the README file.
3. Make sure all your programs run correctly on the CS machines. You will lose 40 points if your code is not running on these machines. The path to the data should be an input parameter, and not hardcoded.
4. Submit your assignment through blackboard.

Notes:

1. Ignore any documents that do not have a gold file (i.e., for which there is no ground truth). You can have an if statement in your code that checks for the existence of the gold file and if the file does not exist, continue to the next file. A less attractive solution is to remove the files with no gold from the folder "abstracts."
2. Please make sure you run a PageRank for each file in the "abstracts" folder. There is no one single PageRank run on the entire collection, but instead one PageRank for each file. The goal is to rank candidate phrases based on their PageRank scores for each document independently. The MRR of each file contributes to the final MRR that you report. In computing the final MRR, we take the average of all individual MRRs.
3. An edge in the graph is added between two nodes, if the corresponding words are adjacent in text (one by the other, or consecutive). For adjacency, you can consider: (i) the entire document as a sequence, or (ii) each sentence as a sequence. For example, for (i) "...w1/NN. w2/NN..." will result in the bigram w1 w2, whereas no bigram will be formed for (ii) because of the "." (end of sentence).
4. Please make sure you stem both the text and the gold. The only text processing on the gold is stemming. That is, do not explicitly remove any stopwords from the gold, if there are any (unless your stemmer will automatically do that).
5. For MRR@k calculation, after you form the phrases and sum up the scores from individual tokens, rank the phrases from highest to lowest scores, and take top-k predicted keyphrases (please make sure you do this for all k ranging from 1 to 10).

   To find a match in the gold, iterate through the k predicted keyphrases to check if there is an exact match anywhere in the gold (i.e., do not check only the first keyphrase in the

gold, but all of them until you find an exact match). If there is such a match, record the rank of this first correct prediction, calculate the MRR for this individual file as the inverse of this rank, add this individual MRR to the global MRR, and continue to the next file (or alternatively, calculate the MRRs for each k from 1 to 10 for the current file and then continue to the next file).

When computing MRR, you always check for a match from the text (or predicted phrases) to the gold. That is, for the predicted keyphrases, check if there is a match anywhere in the gold (remember, the gold is an unordered set).

If no matches are found between predicted keyphrases and gold keyphrases, MRR is 0 for that document.

6. For *tf-idf* calculation, you have two options: (i) calculate the *tf-idf* for each individual token and then sum up the scores, or (ii) calculate the *tf-idf* for the entire phrase. I recommend (i). However, either way is fine. There is no need to normalize the *tf-idf scores*.