Ranking Documents with Eigenvector Centrality

You are given a corpus of Amazon food reviews. Using R, process the data and build a command line search engine. The system when given a query, should retrieve the top 10 documents relevant to the query.

Objectives:

Static ranking of authors of reviews (Eigenvector Centrality)

Steps for finding the static ranks.

The similarity of the documents with the query can be retrieved and can be used to rank the results. But often the queries are too short (say 2 or 3 words), that it is often difficult to rely on the word similarity to rank the retrieved documents. Here we use the notion of credibility of a given review(document) to rank the retrieved results. We define the credibility of a review as the credibility of its author. Now to calculate the credibility of an author we use Eigenvector centrality, a precursor of pagerank.

We assume there is a weighted graph **G(V,E,M).**

Vertex set **V** = The set of all authors available in the dataset. The vertices are labelled by the **review/userId** key from the dataset.

Edge Set **E** = An edge exist between two nodes, if two authors have written a review for the same product. The information can be obtained through the key **product/productId**.

Edge Weights **M** = Every edge has a weight associated with itself, which is what we call as the transition probability. To calculate the transition probability, use the metric **influence(i,j)**, as given below. Please note that the function is asymmetric.

Influence depends on following factors

$Rw_{a_{k,p}}$	-	Review by author k, for product p.
$Rt_{a_{k,p}}$	-	Rating given by author k, for product p along with his/her review
$hlp_{a_{k,p}}$	-	Helpfulness score for the review written by author k for product p. Its the numerator value for the key in the dataset
V_p	-	Number of unique authors who has written a review for the product p

To calculate the influence of one user to another,

$$influence(a_i, a_j) = \sum_{p \epsilon Products} influence_p(a_i, a_j)$$

where $influence_p$ is

$$influence_p(a_i, a_j) = \begin{cases} 0 & \text{if } Rw_{a_{i,p}} \text{ or } Rw_{a_{j,p}} \text{ does not exist} \\ 0.1 & \text{else if } Rt_{a_{i,p}} \text{ or } Rt_{a_{j,p}} \text{ or } hlp_{a_{i,p}} \text{ is } 0 \\ \frac{(5-abs(Rt_{a_{i,p}} - Rt_{a_{j,p}}))}{5*(V_p-1)} * hlp_{a_{i,p}} & \text{Otherwise} \end{cases}$$

With n authors, we can form a matrix **M** of size n^2, such that the (i,j)th entry of the matrix represents normalised value of influence(a_i,a_j).

$$m_{i,j} = \frac{influence(a_i, a_j)}{\sum_{j=1}^{n} influence(a_i, a_j)}$$

To calculate the Eigenvector centrality, perform the following steps.

1. pagerank vector **r -** Define a vector $\mathbf{r}_{nX\,1}$, where a is the number of authors present in the corpus. initialize each entry in the vector with value 1/n (uniformly distributed)

2. transition probability matrix $\mathbf{M}_{nX\,n}$ -
calculate the centrality iteratively- **r(k+1) = M*r(k),** where k is the iteration step.

iterate step 2 till steady state is reached i.e, until **abs(r_i(k+1) - r_i(k)) < 0.00001**, for all i in r.

Which means every element should be similar by 4th decimal point.

Retrieving results.

When a query is input by the user.

1. Input the query, and represent the query as a vector. Don't forget to perform stemming and stopword removal on the query as well.
2. Find the cosine similarity between the query vectors and document vectors
3. Select the top 25 results, based on cosine similarity, and then rank the vectors by the product of eigenvector centrality and similarity. Keep the top 10 results.

Rank Lists:

From the current and previous assignments we have the following entities at our disposal:

1. L_1 - top 25 results based on cosine similarity from unigram term - document matrix
2. L_2 - top 25 results based on unigram and bigram (after filtering of bigrams) term-document matrix. Unigram weight = 0.7, bigram weight = 0.3 (you may take the weights to be 0.55 and 0.45 respectively as well, just report the weight in your assignment document)
3. L_3 - Eigenvector centrality values which in turn are the values for individual reviews.

Produce the following ranked lists for every query provided:

1. R1 - Ranked list of top 10 results based on L_1
2. R2 - Ranked list of top 10 results based on L_2
3. R3 - Ranked list of top 10 results based on L_3 x L_1, i.e. for the 25 results in L_1, find the product of corresponding eigenvector centralities and cosine similarities, and rank the result according to the product value.
4. R4 - Ranked list of top 10 results based on L_3 x L_2

FInd the Spearman coefficient correlation for each query provided between R1&R2, R1&R3 and R2&R4, and also plot the scatter plot for all the three pairs.

Spearman coefficient correlation

Spearman coefficient correlation is a nonparametric measure of statistical dependence between two variables. It assesses how well the relationship between two variables can be described using a monotonic function. It is often used as a statistical method to aid with either proving or disproving a hypothesis. Here we use it to see, how well different rank lists relate to each other. The value can be calculated as:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}. \quad \text{where } d_i = x_i - y_i$$

Eigenvector Centrality

The value **λ** is an eigenvalue of matrix A if there exists a nonzero vector x, such that Ax = **λ**x. Vector x is an eigenvector of matrix A
The largest eigenvalue is called the principal eigenvalue. The corresponding eigenvector is the principal eigenvector which corresponds to the direction of maximum change.

Eigenvector centrality is a measure of influence of a node in a network. It assigns relative scores to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. The idea is to define centrality of the vertex as the sum of centralities of its neighbours.

To define the centrality better, we will have to be aware of the **Perron-Frobenius Theorem**

If A is an n x n nonnegative primitive matrix, then there is a largest eigenvalue
$\boldsymbol{\lambda_0}$ such that
(i) $\boldsymbol{\lambda_0}$ is positive
(ii) $\boldsymbol{\lambda_0}$ has a unique (up to a constant) eigenvector $\mathbf{v_1}$, which may be taken to have all positive entries
(iii)$\boldsymbol{\lambda_0}$ is non-degenerate
(iv)$\boldsymbol{\lambda_0} > |\boldsymbol{\lambda}|$, for any eigen value $\boldsymbol{\lambda}$ not equals $\boldsymbol{\lambda_0}$

We now proceed to define the centrality value of a vertex as a sum of centralities of its neighbours. To begin with, we initially guess that a vertex i has a centrality $x_i(0)$.

We gradually improve this estimate by employing a Markov model, and continue in this manner until no more improvement is observed. The improvement made at step t is defined as,

$$\begin{aligned} x_i(t) &= \sum_j A_{ij} x_j(t-1) \\ \Rightarrow x(t) &= \mathbf{A} x(t-1) \\ &= \mathbf{A}^t x(0) \end{aligned}$$

Deliverables:

Submit the R scripts used for ranking the authors, indexing the documents (previous assignment) and the script that can intake a query and retrieve the top 10 results as a single

compressed file. The scripts should be properly commented and indentations should be followed.
Also submit the plots for all the pairs, for the first 3 queries (9 different plots).