COL764 Assignment 3 Report

Kshitij Alwadhi (2019EE10577)

11th November 2021

1 Introduction

In this assignment, the goal is to compute prior similarity-based authority of the documents and qualitatively analyze the reuslts. We will be performing the following tasks:

1. Computing a similarity graph between each pair of documents by using two similarity functions, Cosine and Jaccard.
2. Compute the PageRank scores for all the documents by using this similarity graph.

2 Similarity Graph

We compute the similarity between two documents by the following two metrics:

2.1 Jaccard/Term overlap similarity

In this metric, the similarity between two documents is given by:

$$Sim_{jaccard}(d_1, d2) = \frac{|TermSet_{d1} \cap TermSet_{d2}|}{|TermSet_{d1} \cup TermSet_{d2}|}$$

2.2 Cosine/TF-IDF Similarity

This is the same metric we computed in previous assignments.

$$tf_{i,j} = log_2(1 + f_{i,j})$$

$$idf_j = log_2(1 + \frac{|D|}{df_j})$$

$$\vec{d_j} = (w_{1,j}, w_{2,j}, ..., w_{t,j})$$

$$\text{TF-IDF Score} = tf_{i,j} * idf_i$$

$$Sim_{cosine}(d_1, d_2) = \frac{\vec{d_1}.\vec{d_2}}{\|\vec{d_1}\|\|\vec{d_2}\|}$$

3 Pre-processing

Similar to what was done in the previous assignments, we follow a two step process to pre-process our data.

1. **Tokenization**: We split the given text by using the delimiters given in the assignment and extract the tokens.
2. **Stemming**: We stem the tokens into their root form by using Porter stemmer.

4 Algorithmic Details

4.1 Creation of similarity graph

Firstly, we create a graph for the documents given to us with the nodes being the documents given to us and the edges being the similarity between the two documents given by the respective similarity metrics. This graph is stored in the form of a list of weighted edges with each tuple being: docid1, docid2, similarity. Since the similarity is irrespective of the order of the documents, what we get is an un-directed weighted graph.

4.2 Feeding the similarity graph to the NetworkX library

This step is pretty straightforward. The NetworkX library has a command to directly take the input of list of weighted edges and then it creates the NetworkX graph directly from them.

4.3 Computing the PageRank scores

Even though this step is also a straight command in the NetworkX library. It's interesting to look into what goes on behind the scenes.

Usually what happens in the PageRank algorithm is that the internet is represented as this graph with each node representing the website and connections from this website to other websites denoted by the nodes. These edges are directed in nature in the classical setting, however, since we have an undirected graph, the library **automatically duplicates every edge** such that we have one edge going from doc1 to doc2 and another edge with same weight going from doc2 to doc1.

Now we have a random surfer model which goes from one node (document) to another with 85% of the jumps being controlled by the links(edges) going from the particular document to other documents and 15% of the jumps being completely random. This is controlled by the damping factor setting which we have set to 0.85 as given in the problem statement. In an unweighted setting, the page rank is computed as follows:

$$PR(u) = (1-d) + d * \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

That is, the PageRank value of a page u is dependent on the PageRank values for each page v contained in the set B_u which contains all the pages linking to page u divided by L(v) which is the number of links from page v.

Now if we have a weighted page rank algorithm, we extend our original PageRank algorithm to account for the weighted edges. Instead of dividing the rank value of a document equally amongst its out-linking documents, we assign higher rank values to the documents which have greater similarity. This is analogous to saying that the random surfer in our random surfer model will be making jumps based on the different probabilities (given by normalizing the edge weights (similarities) so that the sum of probabilities of all the possible jumps from a node is 1). The modified formula for computing the pagerank is given by:

$$PR(u) = (1-d) + d * \sum_{v \in B_u} PR(v) * W_{v,u}$$

Here, d is our damping factor. In a nutshell, we can say that, by increasing the weight of an incoming edge, the page rank of that particular node(document) also increases where the edge is incoming.

All these experiments were done using the NetworkX library and my experience was very positive. It was directly able to take in the weighted undirected graph and did all the processing in the background.

5 PageRank scores

In this section, we present the PageRank scores of the top-20 documents for the two similarity metrics.

5.1 Cosine similarity

Document ID	PageRank
talk.politics.misc/178908	0.0002784764512353384
talk.politics.misc/179058	0.000277218652699401
soc.religion.christian/21496	0.0002709493864725567
talk.politics.misc/179073	0.00026264785460049223
talk.politics.mideast/77198	0.0002625466214912809
talk.politics.mideast/77195	0.0002607402038470326
talk.religion.misc/84223	0.00025653363394910765
sci.crypt/15812	0.00025186447110405635
soc.religion.christian/21597	0.00025150699398976623
alt.atheism/54233	0.0002512820148730642
soc.religion.christian/21458	0.00025107185687754435
talk.politics.mideast/77186	0.00025099623335860985
comp.sys.mac.hardware/52004	0.0002488599512339949
talk.religion.misc/84079	0.0002488570104312421
alt.atheism/53639	0.0002487781131867361
talk.politics.misc/179034	0.00024716695831360696
alt.atheism/53637	0.0002471265235362192
soc.religion.christian/21748	0.00024712577362992033
soc.religion.christian/21536	0.00024692492973181734
soc.religion.christian/21577	0.00024665962208910737

Table 1: PageRank scores for Cosine similarity

5.2 Jaccard similarity

Document ID	PageRank
sci.electronics/54247	0.0001765969312897099
sci.med/59271	0.00017057567968156126
sci.med/59454	0.0001704373884469384
talk.religion.misc/84349	0.00016919325721150954
rec.autos/103727	0.0001691358541639659
sci.med/59407	0.00016698266476718356
alt.atheism/54160	0.00016679864293296797
rec.sport.baseball/104999	0.00016650315731600433
comp.sys.ibm.pc.hardware/60807	0.00016624344198441782
comp.os.ms-windows.misc/10201	0.00016611616849914832
rec.sport.baseball/104986	0.00016606870480758253
comp.sys.ibm.pc.hardware/60804	0.00016598206140609252
sci.electronics/54164	0.0001659503797826 2315
talk.politics.guns/54554	0.00016541777695633968
comp.sys.mac.hardware/52241	0.00016510577530912693
comp.sys.mac.hardware/52443	0.00016462291664395923
comp.graphics/38863	0.00016460824705453558
soc.religion.christian/21586	0.00016456787087400354
comp.sys.ibm.pc.hardware/60741	0.00016445760858606684
rec.sport.hockey/54735	0.0001644062823921802

Table 2: PageRank scores for Jaccard similarity

6 Analysis of the results

Having a higher PageRank implies that in our random surfer model, the random surfer is more likely to land on that particular document after infinite jumps. We can make the following specific conclusions for the kind of documents that have a high PageRank scores:

1. In the classical implementation of PageRank algorithm we had the following formula:

$$PR(u) = (1-d) + d * \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

   From this we can say that a document will have a high PageRank score if it has an edge from a page with high page rank and also, the page rank of that page is not distributed to a lot of pages, i.e. L(v) should be small (less number of outgoing edges). The more the number of such edges, the higher will be the page rank. If a document is referenced by a lot of documents then its PageRank will automatically be higher (more number of incoming edges).

2. From the algorithmic point of view, we see that by increasing the weight of an incoming edge, the page rank of that particular node(document) also increases where the edge is incoming. The weight can be increased by increasing the similarity between the docs. So clearly, those documents which have a greater similarity score with a larger number of documents will have a greater PageRank score. This is evident from the random surfer model as the probability of visiting those pages will also be on the higher side and hence we will have a greater PageRank.

$$PR(u) = (1-d) + d * \sum_{v \in B_u} PR(v) * W_{v,u}$$

   The above formula illustrates this. If the weight is larger for an edge with a high PageRanked document, then the document will get a higher PageRank.

3. One more thing that is clear from inspection, the documents which are on the top of the list of Cosine similarity have a considerably larger size than the rest of the documents. This shows that the content present in them is larger as compared to the other documents i.e. possibility of containing a wide variety of words, and hence it will contain more non-zero terms in the document vector on avg and also the term frequency will also be higher. Even though we are normalizing things but this will have an impact on the dot product we take while computing the cosine similarity.

7 References

1. **Wikipedia: PageRank** https://en.wikipedia.org/wiki/PageRank
2. **GeekForGeeks: Weighted Pagerank** www.geeksforgeeks.org/weighted-pagerank-algorithm/
3. **NetworkX Documentation** https://networkx.org/documentation/stable/tutorial.html