Web Mining – CSE3024

Page Ranking Algorithm – Simple and Weighted

By Abhijeet Ambadekar (16BCE1156)

Aim:

To find the ranks of the pages using the page ranking algorithm and the weighted page ranking algorithm.

Program Code:

```
from pprint import pprint


class PageRanker(object):
	def __init__(self, dampening=0.85, num_iter=100):
		self.dampening = dampening
		self.num_iter = num_iter


	def fit(self, idx):
		self.idx = idx
		self.idxmat = [self.idx[k] for k in sorted(self.idx.keys())]
		self.idxmap = {i:sorted(self.idx.keys())[i] for i in
range(len(self.idx.keys()))}
		self.inlinks = self.gen_inlinks()
		self.outlinks = self.gen_outlinks()
		self.pageranks = {k:1 for k in self.idx.keys()}



	def gen_inlinks(self):
		inlinks = {}
		for index, k in enumerate(sorted(self.idx.keys())):
			inlinks[k] = sum([self.idxmat[i][index] for i in
range(len(self.idxmat))])


		return inlinks
```

```
	def gen_outlinks(self):
		outlinks = {}
		for k in self.idx.keys():
			outlinks[k] = sum(self.idx[k])

		return outlinks

	def calculatePR(self):
		for i in range(self.num_iter):
			new_page_ranks = {}
			for id1, curr_page in enumerate(sorted(self.idx.keys())):
			# for pg in self.inlinks.keys():
				new_page_ranks[curr_page] = 1 - self.dampening
				sum_of_inlinks = 0
				for id2, page in enumerate(sorted(self.idx.keys())):
				# for page in range(len(self.idxmat)):
					# print(page)
					if self.idxmat[id2][id1]:
						sum_of_inlinks +=
self.pageranks[page]/self.outlinks[page]
				new_page_ranks[curr_page] += self.dampening * sum_of_inlinks

			self.pageranks = new_page_ranks.copy()


	def print_details(self):
		print("The inlinks of the graph are:")
		pprint(self.inlinks)

		print("The outlinks of the graph are:")
		pprint(self.outlinks)
```

```
            print("The page ranks of the page after {} iterations are:
".format(self.num_iter))
            pprint(self.pageranks)


            print("Sum of page ranks", sum(self.pageranks.values()))


if __name__ == "__main__":
      dummy_index = {
            "A" : [0, 1, 1, 1, 0],
            "B" : [1, 0, 1, 1, 0],
            "C" : [0, 0, 0, 1, 0],
            "D" : [0, 0, 1, 0, 1],
            "E" : [0, 1, 1, 1, 0],
      }



      pr = PageRanker(0.85, 1000)
      pr.fit(dummy_index)
      pr.calculatePR()
      pr.print_details()
```

Output:

```
/media/anonymous/Work/Vit/Semester 5/WM/Lab/L4_PageRanker python3 L4_PageRank
er_1156.py
The inlinks of the graph are:
{'A': 1, 'B': 2, 'C': 4, 'D': 4, 'E': 1}
The outlinks of the graph are:
{'A': 3, 'B': 3, 'C': 1, 'D': 2, 'E': 3}
The page ranks of the page after 1000 iterations are:
{'A': 0.29102272727272727,
 'B': 0.4977272727272727,
 'C': 1.4249999999999998,
 'D': 1.8499999999999996,
 'E': 0.9362499999999998}
Sum of page ranks 4.999999999999998
```

Program Code: Weighted Page Ranker

```
from pprint import pprint

from L4_PageRanker_1156 import PageRanker


class WeightedPageRanker(PageRanker):


	def fit(self, idx):

		super(WeightedPageRanker, self).fit(idx)

		self.revidxmap = {self.idxmap[k]:k for k in sorted(self.idxmap.keys())}




	def get_win(self, a, b):

		win_sum = 0

		id1 = self.revidxmap[a]

		for id2, k in enumerate(sorted(self.idx.keys())):

			if self.idxmat[id2][id1]:

				win_sum += self.inlinks[k]

		return self.inlinks[k] / win_sum


	def get_wout(self, a, b):

		win_sum = 0

		id1 = self.revidxmap[a]

		for id2, k in enumerate(sorted(self.idx.keys())):

			if self.idxmat[id2][id1]:

				win_sum += self.outlinks[k]

		return self.outlinks[k] / win_sum


	def calculateWeightedPR(self):

		for i in range(self.num_iter):

			new_page_ranks = {}

			for id1, curr_page in enumerate(sorted(self.idx.keys())):

				new_page_ranks[curr_page] = 1 - self.dampening

				sum_of_inlinks = 0
```

```
                        for id2, page in enumerate(sorted(self.idx.keys())):
                                if self.idxmat[id2][id1]:
                                        sum_of_inlinks += self.pageranks[page] *
self.get_win(curr_page, page) * self.get_wout(curr_page, page)
                        new_page_ranks[curr_page] += self.dampening * sum_of_inlinks


                self.pageranks = new_page_ranks.copy()



if __name__ == "__main__":
        dummy_index = {
                "A" : [0, 1, 1, 1, 0],
                "B" : [1, 0, 1, 1, 0],
                "C" : [0, 0, 0, 1, 0],
                "D" : [0, 0, 1, 0, 1],
                "E" : [0, 1, 1, 1, 0],
        }



        pr = WeightedPageRanker(0.85, 1000)
        pr.fit(dummy_index)
        pr.calculateWeightedPR()
        pr.print_details()
```

Output: