Information Processing and Retrieval

Instituto Superior Técnico 2020

Lab 8: Link Analysis

1

Implement a function that takes as input a graph of hiperlinked documents and computes the PageRank vector for all documents.

The input graph is a dictionary, where each key is a document ID d_i and each value is a `set` containing the documents that **link to** document d_i. The output is a dictionary where each key is a document ID d_i and each value is the PageRank value of document d_i.

The function should also take as input the number of iterations to perform and the *damping factor*.

> **Note:** To compute PageRank you also need to know how many outlinks a given document has. This value should also be available to the function.

2

The file pri_cfc.txt is composed of several text documents from the CFC collection[1]. It contains one document per line, and the first word of each line is the document ID.

The file pri_links.txt contains a graph of links between the documents in the CFC collection. Each line contains a set of document IDs, where the first ID is a document d_i and the remaining are the documents **linked by** d_i.

Using this file, compute the PageRank of all documents in the CFC collection. Print all document IDs and their respective PageRank values, sorted descending by PageRank value.

You should experimentally determine how many iterations are necessary for the algorithm to converge.

> **Note:** To compute PageRank of a document d, you need to know which documents link to d. Thus, you need to invert the given link graph.

3

Using the Whoosh search engine from a previous lesson, implement a script that performs searches and returns the results ordered by a combination of textual similarity and PageRank. Use any formula you want to combine both values.

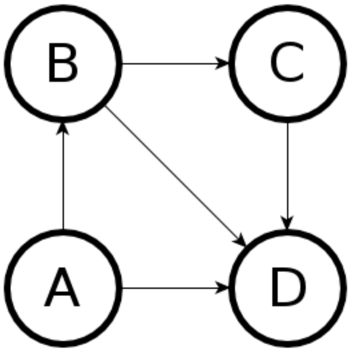
[1] `http://www.dcc.ufmg.br/irbook/cfc.html`

The following example code shows how to perform a query using Whoosh and obtain the textual similarity score:

```
from whoosh.index import open_dir
from whoosh.qparser import *
ix = open_dir("indexdir")
with ix.searcher() as searcher:
    query = QueryParser("content", ix.schema, group=OrGroup).
       parse(u"first␣document")
    results = searcher.search(query, limit=100)
    for i,r in enumerate(results):
        print r, results.score(i)
```

4 Pen and Paper Exercise

Consider the following graph, representing a set of Web pages.



Compute the PageRank for each page. Fix any *sinks* and assume a damping factor $d = 0.1$.

- E is the *adjacency matrix* of the graph.

$$E[u,v] = \begin{cases} \text{1 iff there is a link from } u \text{ to } v \\ \text{0 otherwise} \end{cases}$$

- The out-degree of node u is given by (*row-wise sum*)

$$N_u = \sum_v E[u,v]$$

- Start with an initial *uniform* prestige vector $p_0[u]$

- Compute

$$p_{i+1}[v] = \frac{d}{N} + (1-d) \sum_{(u,v)\in E} \frac{p_i[u]}{N_u}$$