MERC/IST 2015

Extracção e Análise de Dados na Web

Lab 04: 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 `aula04_links.txt` contains a graph of links between the documents in the CFC collection (used in previous lessons). 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 index created in the previous lesson, implement a script that performs searches and returns the results ordered by a combination of textual similarity (computed by Whoosh) and PageRank. Use any formula you want to combine both values.

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

Note: this item is optional and will not be evaluated.

As in the previous lesson, implement a script that reads the `aula03_queries.txt` file and, for each query, executes the corresponding search and measures the precision, recall and F1 of the results. This time, use your PageRank+Textual combination formula to rank the results.

The script should print out each query and its evaluation values and a final average value for each measure. Try out different combination functions and different damping factors.