danker

Release 0.4.1

Aug 17, 2019

Contents

danker is a light-weight package/module to compute PageRank on very large graphs with limited hardware resources. The input format are edge lists of the following format:

```
A   B
B   C
C   D
```

The nodes can be denoted as strings or integers. However, depending on the size of the graph and the amount of available memory you may have to index string node as integers and map back after computation:

```
# link file
1   2
2   3
3   4

# index file
1   A
2   B
3   C
4   D
```

A pre-requisite for danker is that the input file is sorted. For this, you can use the Linux sort command. If you want to use the *danker_smallmem()* option you need two copies of the same link file: one sorted by the left column and one sorted by the right column:

```
# Sort by left column
sort --key=1 -o output-left link-file

# Sort by right column
sort --key=2 -o output-right link-file
```

The *init()* function is used to initialize the PageRank computation. The following code shows a minimal example for computing PageRank with the *danker_bigmem()* option (right-sorted file not needed):

```
import danker
start_value, iterations, damping = 0.1, 40, 0.85
pr_dict = danker.init("output-left", start_value, False)
pr_out = danker.danker_bigmem(pr_dict, iterations, damping)
result_loc = (iterations % 2) + 1
for i in pr_out:
    print(i, pr_out[i][result_loc], sep='\t')
```

The following code shows a minimal example for computing PageRank with the *danker_smallmem()* option:

```
import danker
start_value, iterations, damping = 0.1, 40, 0.85
pr_dict = danker.init("output-left", start_value, True)
pr_out = danker.danker_smallmem(pr_dict, "output-right", iterations, damping, start_
↪value)
result_loc = (iterations % 2) + 1
for i in pr_out:
    print(i, pr_out[i][result_loc], sep='\t')
```

exception danker.**InputNotSortedException**(*file_name*, *line1*, *line2*)

Custom exception thrown in case the input file is not correctly sorted.

Parameters

- **file_name** – The name of the file that is not correctly sorted.

- **line1** – The line that should be first (but is not).
- **line2** – The line that should be second (but is not).

danker.**danker_bigmem**(*dictionary*, *iterations*, *damping*)
Compute PageRank with big memory option.

Parameters

- **dictionary** – Python dictionary created with *init()* (smallmem set to False).
- **iterations** – The number of PageRank iterations.
- **damping** – The PageRank damping factor.

Returns The same dictionary that was created by *init()*. The keys are the nodes of the graph. The output score is located at the `(iterations % 2) + 1` position of the respecive list (that is the value of the key).

danker.**danker_smallmem**(*dictionary*, *right_sorted*, *iterations*, *damping*, *start_value*)
Compute PageRank with right sorted file.

Parameters

- **dictionary** – Python dictionary created with *init()* (smallmem set to True).
- **right_sorted** – The same tab-separated link file that was used for *init()* sorted by the right column.
- **iterations** – The number of PageRank iterations.
- **damping** – The PageRank damping factor.
- **start_value** – The PageRank starting value (same as was used for *init()*).

Returns The same dictionary that was created by *init()*. The keys are the nodes of the graph. The output score is located at the `(iterations % 2) + 1` position of the respecive list (that is the value of the key).

danker.**init**(*left_sorted*, *start_value*, *smallmem*)
This function creates the data structure for PageRank computation by indexing every node. Main indexing steps include setting the starting value as well as counting the number of outgoing links for each node.

Parameters

- **left_sorted** – A tab-separated link file that is sorted by the left column.
- **start_value** – The PageRank starting value.
- **smallmem** – This value is interpreted as a boolean that indicates whether the indexing should be done for *danker_smallmem()* (file iteration) or *danker_bigmem()* (in-memory). Default is "False".

Returns

Dictionary with each key referencing a node. The value is a list with the following contents - depending on the smallmem parameter and the intended use:

- *danker_bigmem()* [link_count:int, start_value:float, start_value:float, linked_pages:list]
- *danker_smallmem()* [link_cout:int, start_value:float, start_value:float, touched_in_1st_iteration:boolean]

CHAPTER 1

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

D

I