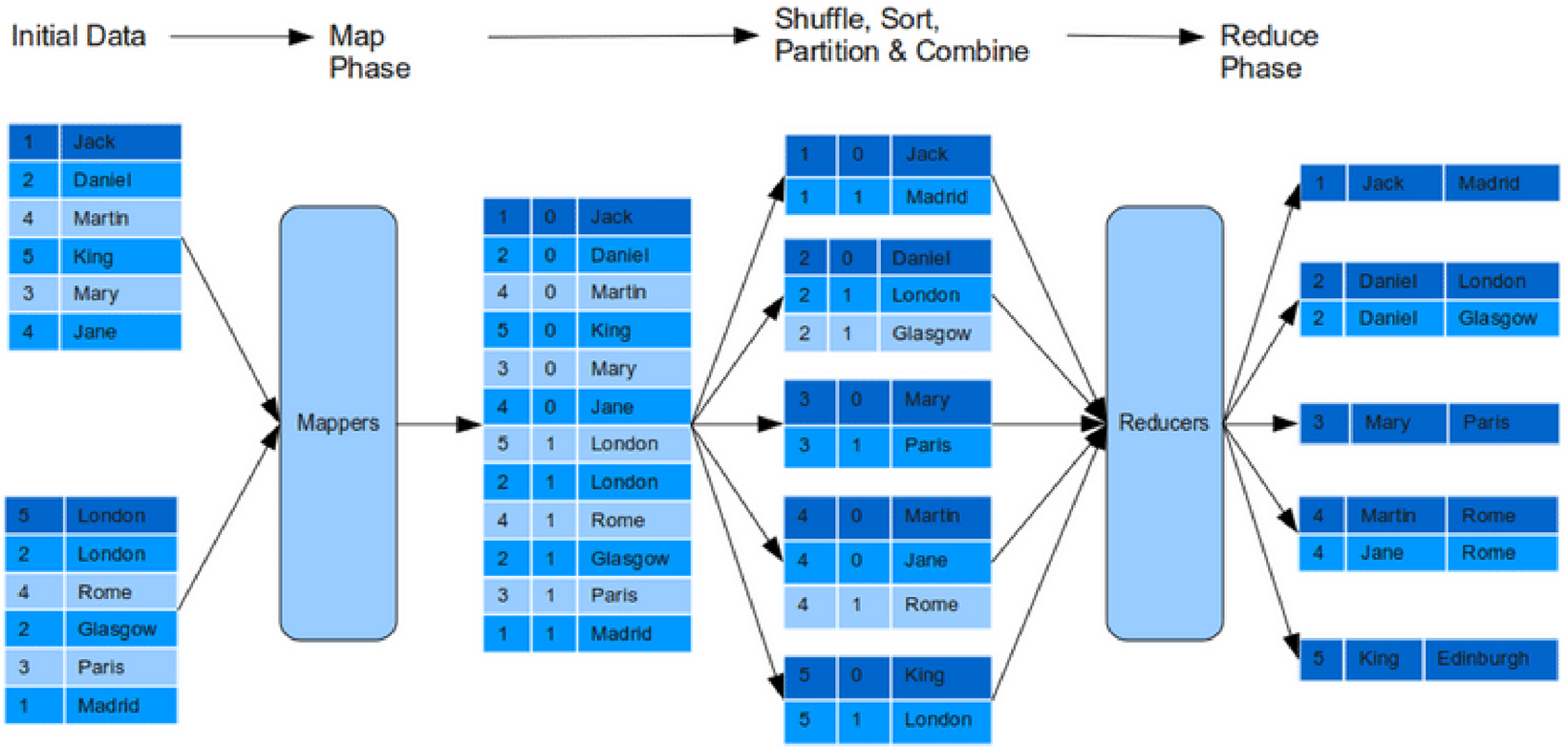
Maxime Le Paumier
Adrien Maîtammar

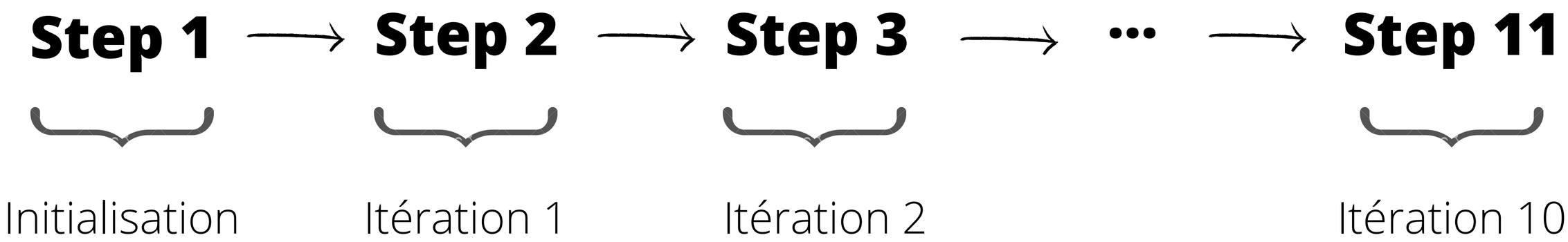
13 décembre 2021

PAGE RANK ET MAP REDUCE

MAP REDUCE


Initial Data
Map Phase
Shuffle, Sort, Partition & Combine
Reduce Phase
1 Jack
2 Daniel
4 Martin
5 King
3 Mary
4 Jane
5 London
2 London
4 Rome
2 Glasgow
3 Paris
1 Madrid
Mappers
1 0 Jack
2 0 Daniel
4 0 Martin
5 0 King
3 0 Mary
4 0 Jane
5 1 London
2 1 London
4 1 Rome
2 1 Glasgow
3 1 Paris
1 1 Madrid
1 0 Jack
1 1 Madrid
2 0 Daniel
2 1 London
2 1 Glasgow
3 0 Mary
3 1 Paris
4 0 Martin
4 0 Jane
4 1 Rome
5 0 King
5 1 London
Reducers
1 Jack Madrid
2 Daniel London
2 Daniel Glasgow
3 Mary Paris
4 Martin Rome
4 Jane Rome
5 King Edinburgh

ALGORITHME


Step 1
Step 2
Step 3
...
Step 11
Initialisation
Itération 1
Itération 2
Itération 10

SQUELETTE DU PROGRAMME

```python
from mrjob.job import MRJob
from mrjob.step import MRStep

class PageRank(MRJob):
    c = 0.15
    nIt = 10
    nodesInstances = set()

    def nodeInit(self,_, line):
        ...

    def rankInit(self, nodeId, AdjacencyList):
        ...

    def mapper(self, nodeId, node):
        ...

    def reducer(self, nodeId, values):
        ...

    def steps(self):
        return [MRStep(mapper=self.nodeInit, reducer=self.rankInit)] +\
        PageRank.nIt * [MRStep(mapper=self.mapper, reducer=self.reducer)]

if __name__ == '__main__':
    PageRank.run()
```

INITIALISATION

```python
def nodeInit(self,_, line):
    lineSplit = line.split('\t',maxsplit=1)
    nodeFrom, nodeTo = lineSplit
    PageRank.nodesInstances.add(nodeFrom)
    PageRank.nodesInstances.add(nodeTo)
    yield nodeFrom, nodeTo

def rankInit(self, nodeId, AdjacencyList):
    node = {'rank':1/len(PageRank.nodesInstances),'AdjacencyList':list(AdjacencyList)}
    yield nodeId, node
```

Si des nœuds ne renvoient pas vers d'autre site alors il ne serront pas initialisé à cette étape.

ensemble des nodeTo
groupé
sur les clés nodeFrom

```python
from mrjob.job import MRJob
from mrjob.step import MRStep

class PageRank(MRJob):
    c = 0.15
    nIt = 10
    nodesInstances = set()

    def nodeInit(self,_, line):
        ...

    def rankInit(self, nodeId, AdjacencyList):
        ...

    def mapper(self, nodeId, node):
        ...

    def reducer(self, nodeId, values):
        ...

    def steps(self):
        return [MRStep(mapper=self.nodeInit, reducer=self.rankInit)] +\
        PageRank.nIt * [MRStep(mapper=self.mapper, reducer=self.reducer)]

if __name__ == '__main__':
    PageRank.run()
```

ITÉRATIONS

```python
def mapper(self, nodeId, node):
    yield nodeId, ('node',node)
    if node['AdjacencyList']:
        contribution = node['rank']/len(node['AdjacencyList'])
        for neighbourId in node['AdjacencyList']:
            yield neighbourId, ('contribution',contribution)

def reducer(self, nodeId, values):
    contributions = 0
    node = {'rank':1/len(PageRank.nodesInstances),'AdjacencyList':list()}
    for value in values:
        if value[0]=='node':
            node = value[1]
        else:
            contributions+=value[1]
    node['rank'] = PageRank.c*node['rank'] + (1-PageRank.c)*contributions
    yield nodeId, node
```

Les noeuds non-initialisé le sont à la première itération

ITÉRATIONS

```python
def mapper(self, nodeId, node):
    yield nodeId, ('node',node)
    if node['AdjacencyList']:
        contribution = node['rank']/len(node['AdjacencyList'])
        for neighbourId in node['AdjacencyList']:
            yield neighbourId, ('contribution',contribution)

def reducer(self, nodeId, values):
    contributions = 0
    node = {'rank':1/len(PageRank.nodesInstances),'AdjacencyList':list()}
    for value in values:
        if value[0]=='node':
            node = value[1]
        else:
            contributions+=value[1]
    node['rank'] = PageRank.c*node['rank'] + (1-PageRank.c)*contributions
    yield nodeId, node
```

→ **values** = Iterator(("contrib.", contrib), ..., ("node", node), ..., ("contrib.", contrib))

ITÉRATIONS

```
def mapper(self, nodeId, node):
    yield nodeId, ('node',node)
    if node['AdjacencyList']:
        contribution = node['rank']/len(node['AdjacencyList'])
        for neighbourId in node['AdjacencyList']:
            yield neighbourId, ('contribution',contribution)

def reducer(self, nodeId, values):
    contributions = 0
    node = {'rank':1/len(PageRank.nodesInstances),'AdjacencyList':list()}
    for value in values:
        if value[0]=='node':
            node = value[1]
        else:
            contributions+=value[1]
    node['rank'] = PageRank.c*node['rank'] + (1-PageRank.c)*contributions
    yield nodeId, node
```

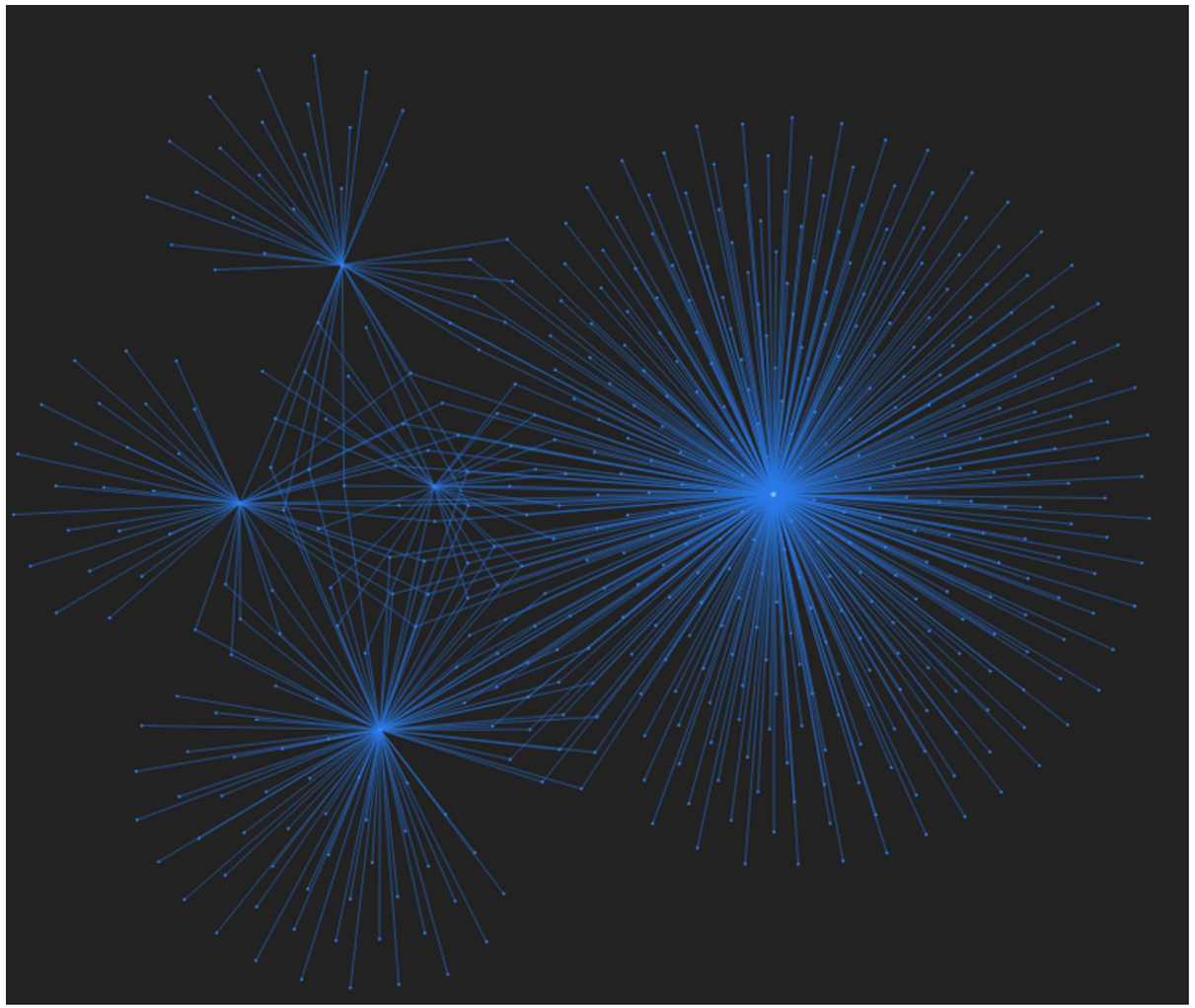
Mise à jour du page rank

```python
from mrjob.job import MRJob
from mrjob.step import MRStep

class PageRank(MRJob):
    c = 0.15
    nIt = 10
    nodesInstances = set()

    def nodeInit(self,_, line):
        ...

    def rankInit(self, nodeId, AdjacencyList):
        ...

    def mapper(self, nodeId, node):
        ...

    def reducer(self, nodeId, values):
        ...

    def steps(self):
        return [MRStep(mapper=self.nodeInit, reducer=self.rankInit)] +\
        PageRank.nIt * [MRStep(mapper=self.mapper, reducer=self.reducer)]

if __name__ == '__main__':
    PageRank.run()
```

VISUALISATION

Top 5

Id.	Page rank
18	0.004124
4415	0.001900
737	0.001758
790	0.001529
1753	0.001511



Maxime Le Paumier
Adrien Maîtammar

13 décembre 2021

PAGE RANK ET MAP REDUCE