MapReduce

DS Team

誰有問題關於MR？

誰跟誰互為好友

```
AAAA    BBBB,CCCC,DDDD,EEEE,XXXX,ZZZZ
BBBB    CCCC,DDDD
DDDD    SSSS,XXXX,AAAA,ZZZZ
EEEE    CCCC,FFFF,ZZZZ
XXXX    AAAA
ZZZZ    EEEE,CCCC
```

**AAAA 的朋友有 BBBB, CCCC, DDDD, EEEE, XXXX, ZZZZ

** BBBB 的朋友有 CCCC, DDDD

標準做法

```
import re

rows = {}
with open("friend.txt", "rb") as in_file:
    for line in in_file:
        me, friends = re.split("\s+", line.strip())

        rows.setdefault(me, set())
        for friend in friends.split(","):
            rows[me].add(friend)

for me, friends in rows.items():
    for friend in friends:
        if friend in rows and me in rows[friend]:
            print("{} and {} are friends".format(me, friend))
```

MR做法

```
AAAA     BBBB,CCCC,DDDD,EEEE,XXXX,ZZZZ
BBBB     CCCC,DDDD
DDDD     SSSS,XXXX,AAAA,ZZZZ
EEEE     CCCC,FFFF,ZZZZ
XXXX     AAAA
ZZZZ     EEEE,CCCC
```

```
AAAA-BBBB 1
AAAA-CCCC 1
AAAA-DDDD 1
AAAA-EEEE 1
AAAA-XXXX 1
AAAA-ZZZZ 1
BBBB-CCCC 1
BBBB-DDDD 1
…
…
```

```
('AAAA-XXXX', 2)
('EEEE-ZZZZ', 2)
('AAAA-DDDD', 2)
```

MR做法

```
rows = sc.textFile(“…./friend.txt”)

def make_relation(x):
    me, friends = re.split("\s+", x.strip())
    return (me, friends.split(","))

def make_tuple(x):
    one, two = x[0], x[1]
    if one < two:
        return ("{}-{}".format(one, two), 1)
    else:
        return ("{}-{}".format(two, one), 1)

relations = rows.map(make_relation).flatMapValues(lambda x: x)
many_relations = relations.map(make_tuple).reduceByKey(lambda x,y: x+y)
both_relation = many_relations.filter(lambda x: x[1] > 1)

for relation in both_relation.collect():
    print relation
```

標準做法

大數據演算法工程師

MR做法

表格轉置

A	B	C	D
foo	one	small	1
foo	one	large	2
foo	one	large	2
foo	two	small	3
foo	two	small	3
bar	one	large	4
bar	one	small	5
bar	two	small	6
bar	two	large	7

		large	small
foo	two	0	6
bar	two	7	6
foo	one	4	1
bar	one	4	5

標準做法

```
#!/usr/bin/python

import numpy as np
import pandas as pd

df = pd.read_csv("table.tsv", sep=" ")

pivot = pd.pivot_table(df, index=["A", "B"], values="D", columns=["C"],
aggfunc=np.sum)
```

```
          C     large   small
=====================
   A     B
bar one      4          5
      two      7          6
foo one      4          1
      two     NaN     6
```

MR做法（1）

```
[u'foo', u'one', u'small', u'1']
[u'foo', u'one', u'large', u'2']
[u'foo', u'one', u'large', u'2']
[u'foo', u'two', u'small', u'3']
[u'foo', u'two', u'small', u'3']
[u'bar', u'one', u'large', u'4']
[u'bar', u'one', u'small', u'5']
[u'bar', u'two', u'small', u'6']
[u'bar', u'two', u'large', u'7']
```

1

```
((u'foo', u'one'), (1, 0))
((u'foo', u'one'), (0, 2))
((u'foo', u'one'), (0, 2))
((u'foo', u'two'), (3, 0))
((u'foo', u'two'), (3, 0))
((u'bar', u'one'), (0, 4))
((u'bar', u'one'), (5, 0))
((u'bar', u'two'), (6, 0))
((u'bar', u'two'), (0, 7))
```

2

```
(u'foo', u'one') [1, 4]
(u'foo', u'two') [6, 0]
(u'bar', u'two') [6, 7]
(u'bar', u'one') [5, 4]
```

MR做法（1）

```
import re

filepath = ….

table = sc.textFile(filepath).map(lambda x: re.split("\s+", x)).filter(lambda x: x[3].isdigit())
```

```
c = table.map(lambda x: x[2]).distinct().collect()
brc = sc.broadcast(c)
```

```
reshaped = table.map(reshape)

for row in reshaped.groupByKey().collect():
    values = [0, 0]
    for v in row[1]:
        for idx, sub_v in enumerate(v):
            values[idx] += sub_v

    print row[0], values
```

```
def reshape(t):
    out = [t[0], t[1]]
    for v in brc.value:
        if v == t[2]:
            out.append(t[3])
        else:
            out.append(0)

    return (out[0], out[1]), (int(out[2]),
int(out[3]))
```

MR做法（2）

```
def add(t1,t2):
    j=()
    for idx, v in enumerate(t1):
        j += (t1[idx]+t2[idx],)

    return j

def seq(current, next):
    return addtup(current, next)

def comb(p, f):
    return addtup(p, f)
```

```
reshaped = table.map(reshape)
pivot = reshaped.aggregateByKey([0 for i in c], seq, comb, 1, seq, comb, 1)

for i in pivot.collect():
    print i
```

Entropy

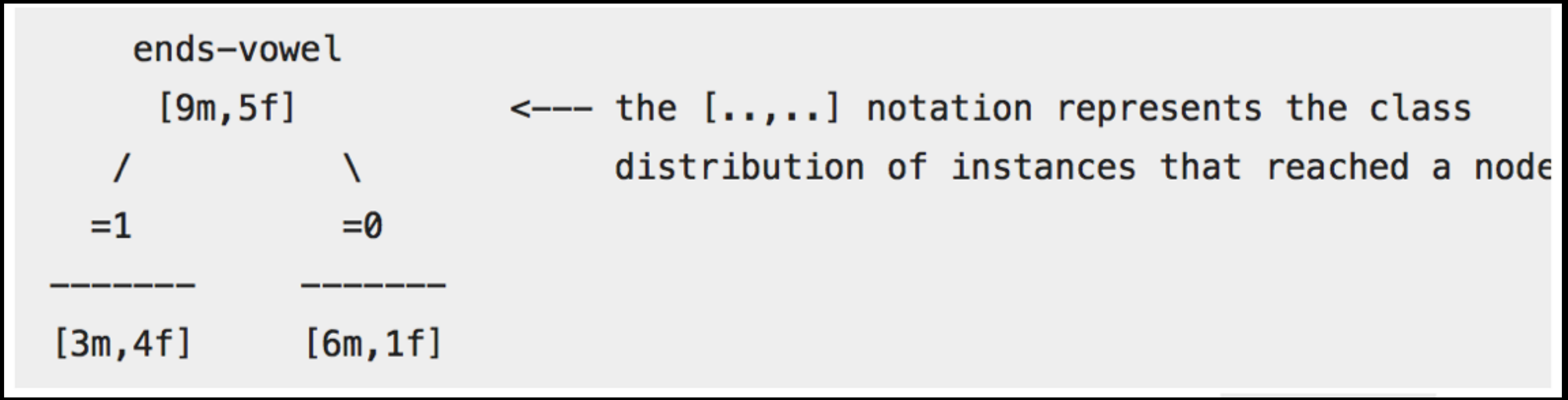
Entropy

↓

Information Gain

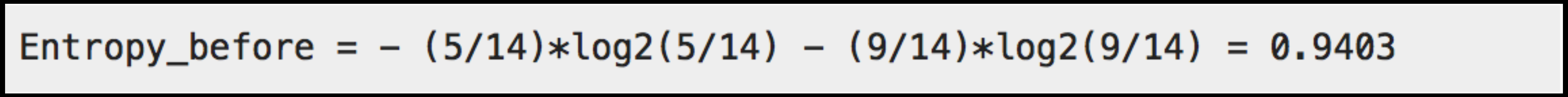
Entropy

Information Gain



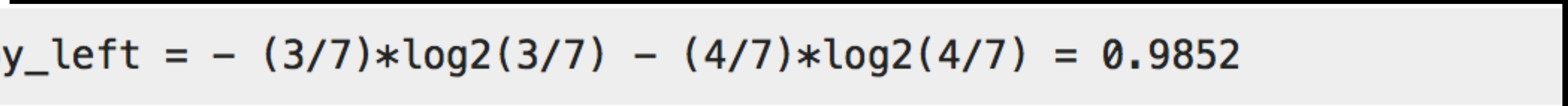
```
    ends-vowel
     [9m,5f]          <--- the [..,..] notation represents the class
   /         \             distribution of instances that reached a node
  =1          =0
-------     -------
[3m,4f]     [6m,1f]
```

1



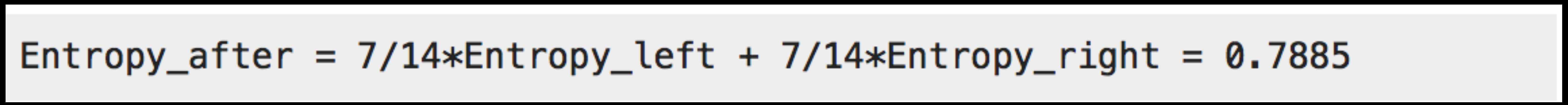
```
Entropy_before = - (5/14)*log2(5/14) - (9/14)*log2(9/14) = 0.9403
```

2



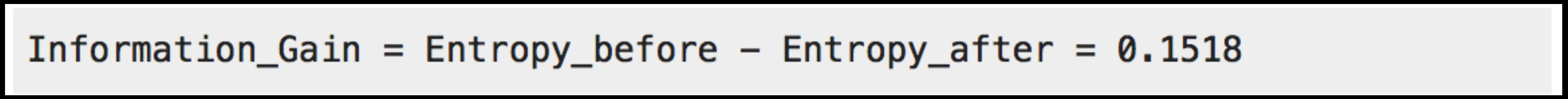
```
Entropy_left = - (3/7)*log2(3/7) - (4/7)*log2(4/7) = 0.9852
```

3



```
Entropy_after = 7/14*Entropy_left + 7/14*Entropy_right = 0.7885
```

4



```
Information_Gain = Entropy_before - Entropy_after = 0.1518
```

MR做法

```
ends-vowel gender
1          m
1          m
1          m
1          m
1          m
1          m
1          m
1          m
1          m
0          f
0          f
0          f
0          f
0          f
```