

Web Mining course, Academic year 2010-2011

Written exam — Solutions

Mauro Brunato

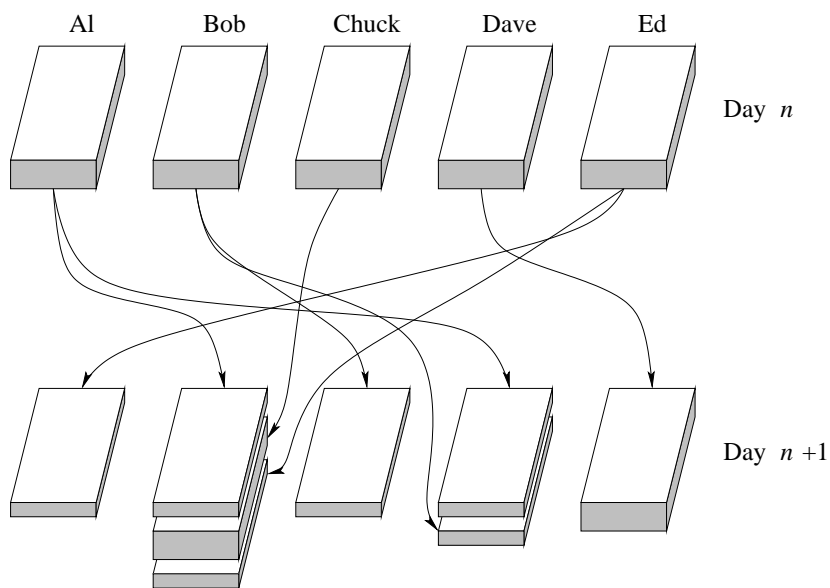
Tuesday, June 14, 2011

Exercise 1

Five friends (Al, Bob, Chuck, Dave and Ed), like play a trading cards game every morning. In order to make the game more interesting, they decide to exchange their decks of cards every day before playing, according to the following rules:

- Al splits his deck in two equal parts and gives the two halves to Bob and to Dave;
- Bob splits his deck in two equal parts and gives the two halves to Chuck and to Dave;
- Chuck gives his deck to Bob;
- Dave gives his deck to Ed;
- Ed splits his deck in two equal parts and gives the two halves to Bob and to Al.

Here is an example of how the decks are dealt every morning:



Let N be the overall number of cards owned by the five friends. Assume that at every step all decks can be precisely split in half.

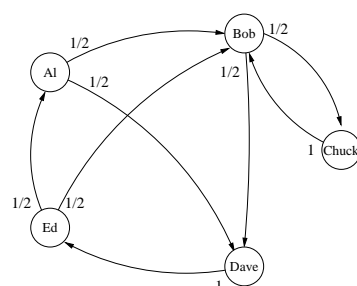
1.1) Show that, no matter how the cards are initially distributed, after a (possibly large) number of days the number of cards owned by each of the five friends stabilizes (i.e., does not change anymore).

1.2) Determine the number of cards in each player's deck when it stabilizes, expressed as a fraction of N . Check the solution for $N = 380$.

1.3) Suppose now that Bob changes his own policy and gives his whole deck to Chuck every morning. Prove that the number of cards owned by each friend does not necessarily stabilize. What is the only stable combination, given as the fraction of N owned by each friend? Check the solution for $N = 380$.

Solution 1

Let N_A, \dots, N_E the number of cards in the deck of Al, \dots , Ed at a given day, so that $N = N_A + \dots + N_E$. Then the decks of cards are redistributed each morning according to the following diagram:



Rather than considering the absolute numbers, the exercise suggests to use the fractions

$$p_A = \frac{N_A}{N}, \dots, p_E = \frac{N_E}{N}$$

representing the relative “wealth” of each friend. Such fractions can be seen as a measure of “prestige” of the owners, whose distribution follows exactly the rules of the PageRank model, or they can be interpreted as relative frequencies, thus representing the probability that a given card belongs to a given user’s deck in that particular day. In this sense, the stochastic matrix

$$\begin{pmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \end{pmatrix}$$

represents the transition probabilities for a given card (if the card belongs to Al, then it goes to Bob with probability $\frac{1}{2}$, and to Dave with probability $\frac{1}{2}$).

Therefore, the probabilities at day i obey the following recurrence equation:

$$\begin{pmatrix} p_A^{(i+1)} \\ p_B^{(i+1)} \\ p_C^{(i+1)} \\ p_D^{(i+1)} \\ p_E^{(i+1)} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 1 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} p_A^{(i)} \\ p_B^{(i)} \\ p_C^{(i)} \\ p_D^{(i)} \\ p_E^{(i)} \end{pmatrix}.$$

1.1) The graph above (or, equivalently, the stochastic matrix) is both irreducible and aperiodic. Therefore, power iterations converge to a unique value, corresponding to the principal eigenvector of the matrix.

1.2) We solve the following linear system:

$$\begin{cases} p_A = \frac{1}{2}p_E \\ p_B = \frac{1}{2}p_A + p_C + \frac{1}{2}p_E \\ p_C = \frac{1}{2}p_B \\ p_D = \frac{1}{2}p_A + \frac{1}{2}p_B \\ p_E = p_D \\ p_A + p_B + p_C + p_D + p_E = 1 \end{cases} \Rightarrow \begin{cases} p_A = \frac{2}{19} \\ p_B = \frac{6}{19} \\ p_C = \frac{3}{19} \\ p_D = \frac{4}{19} \\ p_E = \frac{4}{19}. \end{cases}$$

If $N = 380$, then $N_A = 40$, $N_B = 120$, $N_C = 60$, $N_D = 80$, $N_E = 80$, and one can verify that these numbers won’t change after decks are swapped according to the rules.

1.3) if we remove the arrow from Bob to Dave, the graph isn’t irreducible and aperiodic anymore (consider the lengths of paths from Chuck to Bob), and one can verify that if we start with $N_C = N$ (all other players having no cards) the number of cards never stabilizes because the whole deck will be repeatedly swapped between Bob and Chuck. The only stable point is the principal eigenvector of matrix

$$\begin{pmatrix} 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 1 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

which is obtained by solving the following linear system:

$$\begin{cases} p_A = \frac{1}{2}p_E \\ p_B = \frac{1}{2}p_A + p_C + \frac{1}{2}p_E \\ p_C = p_B \\ p_D = \frac{1}{2}p_A \\ p_E = p_D \\ p_A + p_B + p_C + p_D + p_E = 1 \end{cases} \Rightarrow \begin{cases} p_A = 0 \\ p_B = \frac{1}{2} \\ p_C = \frac{1}{2} \\ p_D = 0 \\ p_E = 0, \end{cases}$$

corresponding to the case in which Bob and Chuck have half the cards each (190 cards, in the example), and keep swapping their decks.

Exercise 2

Let D be any set, and $s : D \times D \rightarrow \mathbb{R}$ a similarity measure between any two elements of D . Discuss some possible ways to generalize s to a similarity measure between any pair of subsets of D (possibly excluding the empty set).

Solution 2

Some solutions discussed during the course were:

- The minimum element similarity:

$$s : A, B \subseteq D \mapsto \min_{\substack{a \in A \\ b \in B}} s(a, b)$$

- The maximum element similarity:

$$s : A, B \subseteq D \mapsto \max_{\substack{a \in A \\ b \in B}} s(a, b)$$

- The average similarity between elements in A and elements in B :

$$s : A, B \subseteq D \mapsto \frac{1}{|A| \cdot |B|} \sum_{\substack{a \in A \\ b \in B}} s(a, b).$$

- The self-similarity of the union (average similarity between all pairs of elements in $A \cup B$):

$$s : A, B \subseteq D \mapsto \frac{1}{|A \cup B|(|A \cup B| - 1)} \sum_{\substack{a, b \in A \cup B \\ a \neq b}} s(a, b).$$

The advantages of the latter, in terms of computational complexity, are discussed in the book.

Exercise 3

Google Analytics logs the time of permanence of each user on each visited web page on a simple text file whose lines are in the following format:

$$page_ID \quad user_ISO_country_code \quad time_of_permanence$$

Consider, for example, the following lines:

```
13287 IT 32
36453 UK 12
32789 IT 45
435 FR 10
2348 UK 30
```

The first one says that the webpage having ID 13287 has been visited for 32 seconds by a user coming from Italy, etc.

The log file is very large and it is stored in a distributed file system (or, possibly, there are many log files).

3.1 Provide a basic MapReduce implementation (i.e., mapper, combiner, reducer) that computes the average permanence time per country code. For example, the outcome of the above log file should be the following:

```
IT 38.5
UK 21
FR 10
```

meaning that users from Italy spent 38.5 seconds on average ($\frac{32+45}{2}$) on each visited page.

3.2 Provide an example in which using a combiner function results in a clear advantage for the framework.

3.3 Suppose that ten thousand machines are available. However, the ISO country codes are only 248. Propose a solution for using all machines in the Reduce phase.

Use any language or pseudolanguage you like, as long as the algorithm is clear.

Solution 3

3.1 The mapper will extract tokens from the input file and provide (key,value) pairs where the key is the ISO country ID, while the value is the visiting time:

$$\text{map} : \Sigma^* \rightarrow (\Sigma^* \times \mathbb{R})^*$$

where input lines and country IDs are both considered as strings (an optimized implementation would probably convert country IDs to integers). We assume that the mapper is invoked for each line, and that strings can be split into blank-separated tokens by the method *split()*, and converted to real numbers by the method *toReal()*:

1. **function** map (string *line*)
2. $\left[\text{string}[] \text{tokens} \leftarrow \text{line.split}() \right.$
3. $\left. \text{emit} (\text{tokens}[1], \text{tokens}[2].\text{toReal}()) \right]$

Observe that *tokens*[0], the page ID, is discarded. Next, a reducer receives all integers associated to the same key and provides a real number (the average):

$$\text{reduce} : \Sigma^* \times \mathbb{R}^* \rightarrow \Sigma^* \times \mathbb{R}.$$

1. **function** reduce (string *key*, integer[] *values*)
2. $\left[\text{real } \text{sum} \leftarrow 0.0 \right.$
3. $\left[\text{integer } \text{count} \leftarrow 0 \right.$
4. **for each** *v* in *values*
5. $\left[\text{sum} \leftarrow \text{sum} + v \right.$
6. $\left[\text{count} \leftarrow \text{count} + 1 \right.$
7. $\left. \text{emit} (\text{key}, \text{sum}/\text{count}) \right]$

3.2) When the same mapper emits lots of pairs related to the same country code, a combiner could precompute a preliminary average. However, the reducer would not be able to compute the correct average. A better approach would be to separately maintain the sum and the number of occurrences (so that the value is a pair composed by a real sum and an integer count) at all stages but the final one:

$$\begin{aligned} \text{map} : \quad \Sigma^* &\rightarrow \left(\Sigma^* \times (\mathbb{R} \times \mathbb{N})\right)^* \\ \text{combine} : \quad \Sigma^* \times (\mathbb{R} \times \mathbb{N})^* &\rightarrow \Sigma^* \times (\mathbb{R} \times \mathbb{N}) \\ \text{reduce} : \quad \Sigma^* \times (\mathbb{R} \times \mathbb{N})^* &\rightarrow \Sigma^* \times \mathbb{R} \end{aligned}$$

```

1. function map (string line)
2.   [ string[] tokens ← line.split()
3.   emit (tokens[1], new pair(tokens[2].toReal(),1))

1. function combine (string key, pair<real,integer>[] values)
2.   [ real sum ← 0.0
3.   [ integer count ← 0
4.   for each v in values
5.     [ sum ← sum + v.first
6.     [ count ← count + v.second
7.   emit (key, new pair(sum,count))

1. function reduce (string key, pair<real,integer>[] values)
2.   [ real sum ← 0.0
3.   [ integer count ← 0
4.   for each v in values
5.     [ sum ← sum + v.first
6.     [ count ← count + v.second
7.   emit (key, sum/count)

```

3.3) One possibility, out of many, is the following.

In the Reduce phase, a single reducer will receive all values associated to the same key; therefore, no more than 248 reducers will be used. However, it is possible to envision a framework modification in which each of the 248 reducers is fed by $\lfloor \frac{10000}{248} \rfloor - 1$ combiner instances organized into a preprocessing hierarchy each receiving a fraction of the overall inputs, thereby implementing a “reducer tree” for every ISO country code.