COMP-SCI 5542 (SP17) - Big Data Analytics and Applications

Tutorial 3 Assignment (Due 02/08/17 by 11:59 PM)

***Dayu Wang* (45)**

1. Spark Programming

1.1. Linear Regression Model

1.1.1. Description of My Own Data Sets

The two parameters I selected are the **hours of sleeping** for every chimpanzee and the **hours of grooming/being groomed** for every chimpanzee. We define that **grooming is positive and being groomed is negative**. We recorded the two parameters for **10 consecutive days**, and the results are saved in the **`LinearRegressionInput.data`** file. An example of the format of the input file is shown in Table 1.

Table 1. Description of the input file for linear regression study of the chimpanzees' daily activities of sleeping *versus* grooming.

Chimp ID (Unique)	Day 1		Day 2			Day 10	
	Sleeping	Grooming	Sleeping	Grooming		Sleeping	Grooming
1	7.64	−0.50	8.55	−1.50	…	7.20	−0.87
2	8.62	0.00	8.23	−0.42		9.73	−0.60
3	13.58	4.87	11.23	4.66		11.29	5.03
4	11.20	2.85	11.30	1.35		13.97	5.22
5	9.97	0.80	10.53	0.00		10.78	1.23
6	7.80	−0.50	8.89	−0.56		8.44	−0.22

1.1.2. Output Results

```
Training MSE = 98.03528060089936
Test MSE = 85.1906435317521
```

(Full Output File: **`LinearRegressionOutput.txt`**)

The results tells us that the linear correlation between the hours of sleeping and the hours of grooming/being groomed is **not** very god, which indicates that a simple linear correlative study is **not sufficient** to describe the relative activities for those chimpanzees at this point.

However, since **the data set was made up by me**, it is just good for study of the Spark Linear Regression Models. In reality, the two parameters may or may not be linearly relative, which required a larger size of data and more various approaches of monitoring chimpanzees' daily activities.

1.2. K-Means Clustering Model

1.2.1. Description of My Own Data Sets

First of all, let's assume that **the entire zoo is almost on the same altitude**, which means only two values (*x* and *y*) will determine the location of a chimpanzee. Then, we would like to know the **top 3 favorite locations** that chimpanzee Jane would like to stay. So, we monitored the **locations of Jane every hour for 10 consecutive days**. Let's say the entire zoo is 1000-by-1000 square feet, which bound the *x* and *y* coordinates to be less than 1000. An example of the data set is shown in Table 2. The full data set is stored in the file **`KMeansInput.txt`**.

Table 2. Description of the input file for K-Means clustering study of a chimpanzee's daily activities related to its locations.

Date	Time	Location-*x*	Location-*y*
12/01/16	00:00	568.3	723.5
12/01/16	01:00	567.3	726.0
…			
12/10/16	23:00	505.2	676.2

1.2.2. Output Results

```
Within Set Sum of Squared Errors = 1.4288214511715658E7
```

(Full Output File: **`KMeansOutput.txt`**)

The results tell us that the error is quite small, which indicates that the K-Means clustering study was successful.

2. Video Annotation

2.1. Algorithm Description to Find the Mostly Mentioned Concept in the Video

Since the workflow design is to carve the entire videos into several critical frames, then the each frame is recognized by Clarifai API, we believe that the concept that appeared the most times, or the total confidence frequency is the maximum, is the main topic of the entire video. Therefore, my approach is to combine all the frequencies together for the same keyword.

The data structure I opted is a sequential container that stores key-value pairs. In Java implementation, I chose to use **`ArrayList<Pair<String, Float>>`** as the sequential container. After running the entire application, the final result is saved in a text file, which gives a one-sentence summary of the video.

Here, I listed the pseudo-code algorithm to find the topic of the video below, which gives an O(n^2) time complexity, where *n* is the number of total generated concepts (including duplicates) by the Clarifai API.

```
Algorithm Create-Key-Value-Pairs(Keywords[n], Values[n])
1   Map ← Ø
2   for i ← 1 to n
3       size ← Map.size
4       found ← false
5       for j ← 1 to size
6           if Map[j].first == Keywords[i]
7               then  Map[j].second ← Map[j].second + Values[i]
8                     found ← true
9           endif
10      endfor
11      if found == false
12          then Map ← Map ∪ (Keywords[i], Values[i])
13      endif
14  endfor
15  Topic ← Map[1].first
16  maxFrequency ← Map[1].second
17  for k ← 2 to n
18      if Map[k].second > maxFrequency
19          then Topic ← Map[k].first
20               maxFrequency ← Map[k].second
21      endif
22  endfor
23  return String("The video was mainly talking about " + Topic + ".")
24  End
```

2.2. Output Results (Table 3)

Table 3. Result of Video Summarization Application Using Clarifai API.

Input File	Topic	Total Frequency
`sample.mkv` (first time)	`wildlife`	`1.99`
`sample.mkv` (second time)	`mammal`	`1.98`
`VideoInput.mkv`	`people`	`0.99`

From the result, we can see that even though the input file was the same, the output result could be different. Also, this application can only deal with very short videos (less than 10 seconds). For longer videos, the system always threw an error, saying that "*Out of Memory*". Accordingly, we can conclude that video annotation is actually a very resource-consuming process.