Coding Test - Michela's Answers

1) **Ranking Algorithms comparison:**
   This is an example of Hypothesis Testing:
   H0 (Null hypothesis) = No difference between algorithms;
   H1 = Two ranking algorithms have a different impact on enrollment fraction.

   In order to decide if the observed difference between the enrollment fraction f_A (experimental ranking) and f_B (current ranking) is significant or not we can calculate a confidence interval at a given confidence level (ex. 95% or 99%) around the metric D = f_A – f_B and see if the expected value of D under the null hypothesis, D = 0, is inside or not this interval.
   If not we can reject the null hypothesis at the chosen confidence level!

   Some math and considerations. The confidence interval is given by:
   $D \pm t \cdot s^*$
   Where:
   t is the value of a probability distribution (t- Student or Normal) for a given number of degree of freedom = g and a given probability $(1-\alpha)$, so $t_{g;(1-\alpha)}$.
   g = N_A + N_B (sum of samples entries number);
   if we fix $\alpha=0.05$ => $(1-\alpha) = 0.95$ so, 95% of confidence level.
   Our samples have a high number on entries (100.000) so we can use a normal distribution to calculate t.

   $s/\sqrt{N}$ is the standard error, but in our case we have two distributions, so we have two different s and two different N.
   In order to find the correct standard deviation we can use the Binomial distribution to model our samples, discrete output enrolment YES or NOT.
   So for the properties of binomial distribution:
   $s_A = f_A \cdot (1-f_A)$;
   $s_B = f_B \cdot (1-f_B)$; => $s^* = \sqrt{((N_A \cdot f_A \cdot (1-f_A) + N_B \cdot f_B \cdot (1-f_A))/(N_A+N_B))}$;

   $\sqrt{N} \Leftrightarrow \sqrt{(1/N_A + 1/N_B)} = \sqrt{((N_A +N_B)/N_A \cdot N_B)}$;

   So we have all we need to calculate the confidence interval:
   $[D - t \cdot \sqrt{((N_A \cdot f_A \cdot (1-f_A) + N_B \cdot f_B \cdot (1-f_A))/(N_A+N_B))}/ \sqrt{((N_A +N_B)/N_A \cdot N_B)},$
   $D + t \cdot \sqrt{((N_A \cdot f_A \cdot (1-f_A) + N_B \cdot f_B \cdot (1-f_A))/(N_A+N_B))}/ \sqrt{((N_A +N_B)/N_A \cdot N_B)}]$

   If it contains D=0, the expected value under the null hypothesis, we cannot reject the null and we have to say that the observed difference is not enough significant to justify the change of ranking algorithm.
   Otherwise we can say that the observed difference is significant at 95% confidence level.

2) **SQL coding:**
   To solve the test we can count the occurrences of each string and cut on this. The best SQL command I found to select non repeated strings is:

   "SELECT string_name FROM string_table GROUP BY string_name HAVING COUNT(*) =1"

where:
string_name = name of string variables;
string_table = name of the one column table.

3) **Python coding:**
Same solution as before, counting on the occurrences and cutting on this, but the Python code could be:

```
array =
["a","b","c","c","d","e","ab","f","f","ac","g","ah","i","i","i","i","l","m","n","o","p","p","q","ccc","dd"
,"r","s","t","t","au","v","z","zz","zz"]

import collections
print ([item for item, count in collections.Counter(array).items() if count == 1])
```

I used Python 3.4 version…

4) **Problem solving test:**
This could be a typical machine learning problem. We could use the set of courses with the category label to find a criteria according with the labels have been assigned, than use the same criteria to apply the label to the non labelled courses.

1. First of all we can split the labelled courses sample in two sets, let say 70% training and 30% testing our algorithm, to evaluate possible overfitting;

2. So a possible labeling criteria, that have to be searched on the training set, could be:

    a. select all the courses with the same label (ex. food) and count how many times each word of the course title or of the course outline (or whatever data we have for the courses) is repeated in the text;

    b. for each words find the % of repetition dividing the number of occurrence with the total number of the words in the title of all the courses with the same label (kind of normalization…);

    c. choose the highest % group of words;

    d. In this way we can find a group of words associate to a given label.

3. Now, on the test set, for each course, we search each word of a given group and count how many times this word appear over the total number of words for ex. in the title.

4. We calculate the label probability, for this course, for the given label of which we searched the words, adding the probability of each single words of the group;

5. Repeat the same for each label, (group of words);

6. So we have, for a given course, the probability associate to each label. This could be the output of the algorithm. In this way we can associate to a course, for example, the

label with the highest probability or more generally associate multi-labels according their percentage;

7. Repeat the procedure for all the courses in the test set and check how many time the algorithm success or fails.

8. Study the variation of the number of success with respect to some parameter (ex. number of words searched for each label) in order to see if it is possible improve the labelling power.

9. When the number of success is the highest possible, we could apply the algorithm to unlabeled sample.