Photo Tag recommendation system

Multimedia Systems and Applications 4/M Coursework Submission

Student name: Terence Aqachmar

Student ID/Matriculation number: 2087939a

**1. Introduction**

I built this Photo Tag recommendation system using Python 2.7.5, Pandas for handling the data as well as Numpy and the math package for some operations.

The packages needed for the execution of the code are: Python 2.7.5, upgraded version of pip, pandas, numpy

**2. Tag Recommendation Strategy**

- Explain the most popular tag suggestion technique  
Using the co-occurrence matrix, I simply iterate over the desired rows (water, people, london) and compare each values of each column with each other in order to get the maximum value.  
I then add the column name that has the maximum to some list that stores the top 5 most popular tags that co occur with the tag currently selected. This same tag that got added to that latter list is also removed from the tag list used to iterate over for the same tag until we get 5 tags. We then repeat the action until we get the 5 top tags.

- Give the pseudo-code

1. initialize tagList with all the tags in order of appearance in tags.csv

2. get co-occurence matrix: matrix

3. init selectedTagsList := ['water', 'people', 'london']

4. initialize popularTagsMatrix with headers as selectedTags (size(selectedTags)\*5)

5. for each selectedTag in selectedTagList:

6. rowIndex := slectedTag.index

7. topTagCount := 0

8. tagList1 = elements of the set of tagList

9. while topTagCount is less than 5:

10. init maxCurrentTag

11. init maxOccurence

12. for each currentTag in tagList1:

13. if matrix[selectedTag][currentTag] is less than maxOccurence

14. then continue loop, goto next currentTag  
15. else:

16. maxCurrentTag := currentTag

17. maxOccurrence := matrix[selectedTag][currentTag]

18. goto next currentTag

19. remove maxCurrentTag from tagList1

20. popularTagsMatrix[selectedTag][topTagCount] = tuple(maxCurrentTag, maxOccurence)

21. topTagCount++

22. insert selectedTag in tagList

23. tagList1 := elements of the set of tagList

24. go to top of the for loop at 5 next selectedTag

2.1 Tag Suggestion with popularity and significance

-Explain the most popular tag suggestion technique

Here we run the tag suggestion exactly the same way as above except that instead of comparing the absolute values from the co-occurrence matrix we compare their significance compared to the whole collection of images using this formula log(I/I(X)) called inverse document frequency. So when we were selecting the maximum value of tags that co-occur a selected tag we selected the maximum IDF value of those tags using the formula (log(I/I(X)) where I is the total number of images and I(X) the number of images with the tag in question; we then multiply this value by the number of co-occurrence with the selected tag and collect the top 5.  
Doing it this way gives less importance to tags that are more common and more importance to the ones that occur usually with that particular tag

-Give the pseudo code

1. initialize tagList with all the tags in order of appearance in tags.csv

2.tagsNumber from tags\_modified.csv with headers as ['n']

3. get co-occurence matrix: matrix

4. init selectedTagsList := ['water', 'people', 'london']

5. initialize popularTagsMatrixIDF with headers as selectedTags (size(selectedTags)\*5)

6. for each selectedTag in selectedTagList:

7. rowIndex := slectedTag.index

8. topTagCount := 0

9. tagList1 = elements of the set of tagList

10. while topTagCount is less than 5:

11. init maxCurrentTag

12. init maxOccurence

13. for each currentTag in tagList1:

14 valueIDF:=matrix[slectedTag][currentTag]\*(log(10000/tagsNumber['n'] [currentTag]))

15. if valueIDF is less than maxOccurence

15. then continue loop, goto next currentTag  
16. else:

17. maxCurrentTag := currentTag

18. maxOccurrence := matrix[selectedTag][currentTag]

19. goto next currentTag

20. remove maxCurrentTag from tagList1

21. popularTagsMatrixIDF[selectedTag][topTagCount] = tuple(maxCurrentTag, maxOccurence)

22. topTagCount++

23. insert selectedTag in tagList

24. tagList1 := elements of the set of tagList

25. go to top of the for loop at 5 next selectedTag

**3. Code description**

a. Give the explanation of the code

matrix.py

we first construct this ordered list of tags → tagList

Then we initialize the matrix that we will have in a dataframe with symmetric axes (using tagList)

Then we set them all to zero apart from the diagonal axis which is set to null  
  
After opening photo\_tags.csv  
we start iterating the csv file and increment the values of tag co-occurrence for each photo. We have to exclude the tag we currently selected for the second for loop

we then write the dataframe into matrix.csv

task2.py

b. put code in appendix

**4. Include top 5 tags for each tag in the following list**

a. water  
nature, 74

blue 71

reflection 63

lake62

landscape 62

b. people

portrait 28

street 27

bw 24

2007 23

explore 21

c. london

explore 32

graffiti 15

geotagged 15

architecture14

street 13

**5. Top 5 tags, based on step 2.1, for each tag in the following list**

5.1 Tag suggestions

a. water

lake 270

reflection 238

nature 236

landscape 224

blue 208

b. people

street 99

portrait 87

bw 70

2007 66

man 53

c. london

explore 72

graffiti 56

geotagged 54

architecture 52

street 47

5.2 Reflections

d. Criticism of most-popular tag suggestion with justification

e. Justification of the results for step 5.1

**6. Improving Recommendations**

a. possible strategies using the timestamp and co-ordinates

We could improve the system by counting dates and times of the day and giving them weight. Some tags like sunset would be occurring more often in the evening for example

b. other possibilities  
We could use language semantics and natural language processing. Techniques like word embedding and other k-nearest neighbors based features could be used in order to determine which tag are likely to co-occur to others. We could also use

7. Appendix – Actual code for recommendation

a. Please comment the code properly

Task 1, creation of the co-occurrence matrix:  
matrix.py

import pandas

import numpy

import math

#this script builds the co-occurrence matrix

#ordered list of tags

tagsDf = pandas.read\_csv('csv/tags.csv', names = ["tags", "n"])

tagList = []

for i in tagsDf.tags:

tagList.append(i)

#creation of co-occurence matrix

matrix = pandas.DataFrame(data = None, index = tagList, columns = tagList)

#initialize values (NaN for the symmetrical axis)

for i in tagList:

matrix[i]=0

matrix[i][i] = numpy.nan

#count co\_occurence

idCount = 1

photo\_tags = pandas.read\_csv('csv/photos\_tags.csv', names = ["id", "tag"])

#getting all the photos and their tags in order to count theor co-occurrence

while idCount<=24999:

if photo\_tags[photo\_tags['id']==idCount].empty:

pass

else:

#get all the tags for that photo id

common\_tags = photo\_tags[photo\_tags['id']==idCount].tag

for tag1 in common\_tags:

#map each tag with all the remaining ones

common\_tags2 = list(set(common\_tags) - set(tag1))

for tag2 in common\_tags2:

matrix[tag1][tag2]+=1

#mapping is done by adding 1 to the co-occurrence

idCount+=1

matrix.to\_csv('matrix.csv')

#you end up with a symmetric matrix with null values in the diagonal

Task 2 most popular recommendation based on popularity  
task2.py  
import pandas

import numpy

#create ordered tagList

tagsNo = pandas.read\_csv('csv/tags.csv', names = ["n"])

tagsNo.to\_csv('csv/tags\_modified.csv', sep = ',')

#tags.csv is easier to handle if read properly in a DataFrame with the right headers

tagsNo = pandas.DataFrame.from\_csv('csv/tags\_modified.csv')

tagList = []

for i in tagsNo.index:

tagList.append(i)

tags = ["water","people","london"]

#copy only the elements

tagList1 = tagList[:]

#this will be our output, it does have a form of a matrix order len(tags)\*5

popTags = pandas.DataFrame(columns = tags, index=range(5))

matrix = pandas.DataFrame.from\_csv('matrix.csv')

for selectedTag in tags:

#save index so that we can re insert it at the end of the loop

selectedTagIndex = tagList.index(selectedTag)

n = 0

#no point of comparing it to null values

tagList.remove(selectedTag)

tagList1 = tagList[:]

while n<5:

maxCurrentTag = ""

maxOccurrence = 0

for currentTag in tagList1:

#if the co-occurrence is not the maximum go to the next one

if matrix[selectedTag][currentTag]<maxOccurrence:

continue

maxCurrentTag = currentTag

maxOccurrence = matrix[selectedTag][currentTag]

tagList1.remove(maxCurrentTag)

popTags[selectedTag][n] = (maxCurrentTag, maxOccurrence)

n += 1

tagList.insert(selectedTagIndex, selectedTag)

tagList1 = tagList[:]

print popTags  
  
  
Task 3 most popular recommendation based on popularity and significance  
task3.py

import pandas

import numpy

import math

#create ordered tagList

tagsNo = pandas.read\_csv('csv/tags.csv', names = ["n"])

tagsNo.to\_csv('csv/tags\_modified.csv', sep = ',')

#tags.csv is easier to handle if read properly in a DataFrame with the right headers

tagsNo = pandas.DataFrame.from\_csv('csv/tags\_modified.csv')

tagList = []

for i in tagsNo.index:

tagList.append(i)

#copy only the elements

tags = ["water","people","london"]

tagList1 = tagList[:]

#this will be our output, it does have a form of a matrix order len(tags)\*5

popTagsIDF = pandas.DataFrame(columns = tags, index=range(5))

matrix = pandas.DataFrame.from\_csv('matrix.csv')

for selectedTag in tags:

#save index so that we can re insert it at the end of the loop

selectedTagIndex = tagList.index(selectedTag)

n = 0

tagList.remove(selectedTag)

tagList1 = tagList[:]

while n<5:

maxCurrentTag = ""

maxOccurrence = 0

for currentTag in tagList1:

# use of the formula log(I/I(X))

valueIDF = float(float(matrix[selectedTag][currentTag]) \* float(math.log(10000/float(tagsNo['n'][currentTag]))))

#if the co-occurrence IDF is not the maximum go to the next one

if valueIDF<maxOccurrence:

continue

maxCurrentTag = currentTag

maxOccurrence = valueIDF

tagList1.remove(maxCurrentTag)

popTagsIDF[selectedTag][n] = (maxCurrentTag, float(maxOccurrence))

n += 1

tagList.insert(selectedTagIndex, selectedTag) #put back the selected tag in the list for next selected tags

tagList1 = tagList[:]

print popTagsIDF