Photo Tag recommendation system
Multimedia Systems and Applications 4/M Coursework

Robert Allison
110285a

Introduction
This assessed coursework was to develop a system to recommend photo tags based on existing user tags. Building a co-occurrence table to catalogue occurrences of 100 different tags wit each other, we should be able to suggest to top five most common tags that occur with any given input tag. In order to offset the problems with popular tags, we can also assign a Inverse Document Frequency number to each tag, to ensure only the most relevant tags are returned.

Tag recommendation
To return the most popular tags for a given tag t, we construct a co-occurrence table of all the tags as they occur with each other in a given photoset. Then given t, we search the table row t and find the five tags with the highest co-occurrence values and return them in descending order.

Pseudo-Code:
(Given dictionary of co-occuring tags for tag t)
Create an empty list (of lists)
For each item in sorted list of dictionary values in reverse order
    Append to list each tag and its occurrence value (in list)
Return first five entries of the list of lists

Popularity and significance
To avoid returning only popular tags, we can assign to each tag an 'IDF' value, that indicates its occurrence with proportion to the total number of images the tag occurs in. This is calculated by Log(I/I(X)) where I is the number of images in the collection, and I(X) is the number of images tagged with tag X. Therefore, popular tags are discounted.

Pseudo-Code:
For ease of calculation, store with each tag its I/I(X) value.
Later, Replace each tag's I/I(X) value with Log of that value (Log(I/I(X)))

For the given tag, store its slice of the IDF co-oc table in a dictionary
Pass this dictionary into the function to return top five tags

Code description
This solution uses three data structures:

photos{}: a dictionary of each photo with list of its tags
tags: a{} dictionary of all the tags and their I/I(x) values
co_occur{}: A dictionary of dictionaries to store co-occurrence values for each tag

Solution:
Since this a brief program we define some re-usable functions then write out the main code in the body of the program.
First we read in the appropriate data values for photos{} and tags{} from the provided csv files storing in photos{} each photo and a list of its tags. In tags{} we store each tag. Then, iterating over tags, we use the size of the photos{} dictionary to calculate the I/I(X) value for each tag and append that to its dictionary entry.

Then, iterating over the collection of photos, we compare all of that tags in each photo and build a co-occurrence table in the co_occur dictionary, where a single entry is keyed by a tag, and its value is a list of all other tags and their co-occurrence values.

This is done so that using these entries, we can then join each entry with comma-separation to produce an appropriate delineated output.

For the purpose of this coursework text input was not necessary, so we simply pass the relevant tags ('water', 'london', 'people') into the functions (as described above) that find the top five and top idf suggested tags and output them in a easy-to-read format and end the program.

Code provided in 'Main.py' provided with this report.

Top five suggested tags:

Water
Suggested Tags:
nature 74
blue 71
reflection 63
landscape 62
lake 62

People
Suggested Tags:
portrait 28
street 27
bw 24
2007 23
explore 21

London
Suggested Tags:
explore 32
geotagged 15
graffiti 15
architecture 14
street 13

TF-IDF Tags:

Water
Relevant Tags (TF-IDF):
lake 270.22
reflection 238.48
nature 236.19
landscape 224.40
blue 208.03

People
Relevant Tags (TF-IDF):
street 99.17
portrait 87.14
bw 70.82
2007 66.63
man 53.69

London
Relevant Tags (TF-IDF):
explore 72.34
graffiti 56.71
geotagged 54.18
architecture 52.04
street 47.75

Reflections

Most popular tagging with regards to popularity and significance seems to produce, at least in this small dataset , relatively the same tags as normal co-occurrence values, with the exception of one or two more relevant tags included in the results. However TF-IDF often places certain tags on the same list higher, so we can see these regarded as more important that typical co-occurrence tagging.
The TF-IDF tags in this dataset are weighted correctly with the highest values selected as the most relevant, but also their IDF values have a greater range of variance, which allows for more nuance when correctly selecting significant tags, or give the most relevant tags a clear difference against more common, popular tags.

Recommendations for Improvement

The most popular tagging system as-is works well in theory with TF-IDF, but it can still fail to capture contextual information through tags, such as location, time, season, etc. This could be improved with more accessible metadata such as timestamps to calculate suggested time-of-day tags, or geotagging to suggest accurate locations. These can be achieved by stripping the underlying data from the image when it's processed, as more and more images, particularly those uploaded from mobile phones, are including this sort of data with them. Another possible source of information is user-provided feedback, such as accompanying posts on social media or comments that may contain useful data pertaining to the image, and may in itself also facilitate further suggestion of tags.

Code

In lieu of an appendix the code is provided in a commented “Main.py” file provided with this report.