ExerciseRound7

October 28, 2022

[1]:
```
# This cell is used for creating a button that hides/unhides code cells to␣
 ↪quickly look only the results.
# Works only with Jupyter Notebooks.

import os
from IPython.display import HTML

HTML('''<script>
code_show=true;
function code_toggle() {
if (code_show){
$('div.input').hide();
} else {
$('div.input').show();
}
code_show = !code_show
}
$( document ).ready(code_toggle);
</script>
<form action="javascript:code_toggle()"><input type="submit" value="Click here␣
 ↪to toggle on/off the raw code."></form>''')
```

[1]:
```
<IPython.core.display.HTML object>
```

[2]:
```
# Description:
#   Exercise7 notebook.
#
# Copyright (C) 2018 Santiago Cortes, Juha Ylioinas
#
# This software is distributed under the GNU General Public
# Licence (version 2 or later); please refer to the file
# Licence.txt, included with the software, for details.

# Preparations
import numpy as np

# Select data directory
```

```
if os.path.isdir('/coursedata'):
    # JupyterHub
    course_data_dir = '/coursedata'
elif os.path.isdir('../../../coursedata'):
    # Local installation
    course_data_dir = '../../../coursedata'
else:
    # Docker
    course_data_dir = '/home/jovyan/work/coursedata/'

print('The data directory is %s' % course_data_dir)
data_dir = os.path.join(course_data_dir, 'exercise-07-data')
print('Data stored in %s' % data_dir)
```

```
The data directory is /coursedata
Data stored in /coursedata/exercise-07-data
```

1 CS-E4850 Computer Vision Exercise Round 7

The problems should be solved before the exercise session and solutions returned via MyCourses. For this exercise round, upload this notebook(pdf and .ipynb versions) containing your source codes (for Exercise 1) and your answer to the question of Exercise2, and all the answers to the questions of Exercise 3 (VGG practical), see part[1-3].ipynb. Note that it's not necessary to upload part1.ipynb, part2.ipynb or part3.ipynb, because all of the necessary questions related to them are contained in this notebook and you're not expected to do any coding in Exercises 2 and 3.

1.1 Exercise 1 - Comparing bags-of-words with tf-idf weighting

Assume that we have an indexed collection of documents containing the five terms of the following table where the second row indicates the percentage of documents in which each term appears.

term	cat	dog	mammals	mouse	pet
% of documents	5	20	2	10	60

Now, given the query $Q = \{mouse, cat, pet, mammals\}$, compute the similarity between Q and the following example documents $D1$, $D2$, $D3$, by using the cosine similarity measure and tf-idf weights (i.e. term frequency - inverse document frequency) for the bag-of-words histogram representations of the documents and the query.

- $D1$ = Cat is a pet, dog is a pet, and mouse may be a pet too.
- $D2$ = Cat, dog and mouse are all mammals.
- $D3$ = Cat and dog get along well, but cat may eat a mouse.

Ignore other words except the five terms. You may proceed with the following steps:

a) Compute and report the inverse document frequency (idf) for each of the five terms. Use the logarithm with base 2. (idf is the logarithm on slide 69 of Lecture 6.)
b) Compute the term frequencies for the query and each document.

c) Form the tf-idf weighted word occurrence histograms for the query and documents.
d) Evaluate the cosine similarity between the query and each document (i.e. normalized scalar product between the weighted occurrence histograms as shown on slide 45).
e) Report the relative ranking of the documents. (You should get similarities 0.95, 0.64, and 0.63, but you need to determine which corresponds to which document.)

[3]:
```
## Comparing  bags-of-words  with  tf-idf  weighting
##--your-code-starts-here--##

# data
q = ["mouse","cat","pet","mammals"]

documents = [
    "Cat is a pet, dog is a pet, and mouse may be a pet too.",
    "Cat, dog and mouse are all mammals.",
    "Cat and dog get along well, but cat may eat a mouse."
]

terms = ["cat","dog","mammals","mouse","pet"]

document_percent = {"cat":0.05,"dog":0.2,"mammals":0.02,"mouse":0.1,"pet":0.6}

# turn documents into bags of words, keeping only given terms
d_bags = []
for d in documents:
    unfiltered_bag = d.lower().replace(',','').replace('.','').split()
    filtered_bag = []
    for w in unfiltered_bag:
        if w in terms:
            filtered_bag.append(w)
    d_bags.append(filtered_bag)

d_bags
```

[3]:
```
[['cat', 'pet', 'dog', 'pet', 'mouse', 'pet'],
 ['cat', 'dog', 'mouse', 'mammals'],
 ['cat', 'dog', 'cat', 'mouse']]
```

[4]:
```
# a) calculate inverse document frequency
idf = {t:np.log2(1/n) for (t,n) in document_percent.items()}

idf
```

[4]:
```
{'cat': 4.321928094887363,
 'dog': 2.321928094887362,
 'mammals': 5.643856189774724,
 'mouse': 3.321928094887362,
 'pet': 0.7369655941662062}
```

[5]:
```
# b) calculate term frequencies
d_bags.append(q)

tf = []
for d in d_bags:
    occurrences = {t:0 for t in terms}
    for t in terms:
        if t in d:
            occurrences[t] += 1
    term_freqs = {t:n/len(d) for (t,n) in occurrences.items()}
    tf.append(term_freqs)

tf
```

[5]:
```
[{'cat': 0.16666666666666666,
  'dog': 0.16666666666666666,
  'mammals': 0.0,
  'mouse': 0.16666666666666666,
  'pet': 0.16666666666666666},
 {'cat': 0.25, 'dog': 0.25, 'mammals': 0.25, 'mouse': 0.25, 'pet': 0.0},
 {'cat': 0.25, 'dog': 0.25, 'mammals': 0.0, 'mouse': 0.25, 'pet': 0.0},
 {'cat': 0.25, 'dog': 0.0, 'mammals': 0.25, 'mouse': 0.25, 'pet': 0.25}]
```

[6]:
```
# c) generate histograms
titles = ["Doc 1", "Doc 2", "Doc 3", "Query"]
tf_idf = [[tf[i][t]*idf[t] for t in terms] for i in range(len(d_bags))]

tf_idf
```

[6]:
```
[[0.7203213491478937,
  0.3869880158145603,
  0.0,
  0.553654682481227,
  0.12282759902770103],
 [1.0804820237218407,
  0.5804820237218405,
  1.410964047443681,
  0.8304820237218405,
  0.0],
 [1.0804820237218407, 0.5804820237218405, 0.0, 0.8304820237218405, 0.0],
 [1.0804820237218407,
  0.0,
  1.410964047443681,
  0.8304820237218405,
  0.18424139854155155]]
```

```
[7]: # d) cosine similarity between query and documents
     cos_sim = []
     for i in range(len(documents)):
         num = np.dot(tf_idf[i], tf_idf[-1])
         denom = np.linalg.norm(tf_idf[i])*np.linalg.norm(tf_idf[-1])
         cos_sim.append(num/denom)

     cos_sim
```

```
[7]: [0.6430234838611961, 0.9546948111493487, 0.6363473188622558]
```

```
[10]: # e) relative ranking
      sorted_cos_sim = sorted(cos_sim, reverse=True)

      print("Relative ranking of documents is as follows:")
      for i, cs in enumerate(sorted_cos_sim):
          print("Number {} is document {}".format(i+1, cos_sim.index(cs)+1))

      ##--your-code-ends-here--##
```

```
Relative ranking of documents is as follows:
Number 1 is document 2
Number 2 is document 1
Number 3 is document 3
```

1.2 Exercise 2 - Precision and recall

There is a database of 10000 images and a user, who is only interested in images which contain a car. It is known that there are 500 such images in the database. An automatic image retrieval system retrieves 300 car images and 50 other images from the database. Determine and report the precision and recall of the retrieval system in this particularcase.

Type your answer here:

We have that precision = #relevant/#returned, so in this case

precision = 300/(300+50) = 0.857

We have that recall = #relevant/#total_relevant, so in this case

recall = 300/(500) = 0.6

1.3 Exercise 3 - VGG practical on object instance recognition

See the questions in part[1-3].ipynb and write your answers here.

Type your answers here:

1.3.1 Part1:

Stage I.A

- The change in density of detections across the image is a byproduct of the fact that SIFT, while invariant to translation, scaling, and rotation, is not invariant to intensity changes. Since in the real pair of images the two pictures have very different intensity values, with the second image even being partially in sunlight and partially in shadow, SIFT is not able to correctly identify as many frames, which will likely be a problem for matching. Some kind of normalization on the intensity values, or turning the image to greyscale, might help mitigate the problem.
- A feature may be detected multiple times with different orientations if the orientation of the edges in it is not clear. This may be the result for example of "noisy" areas with no distinct straight lines.

Stage I.B

- The descriptors are computed over a larger region compared to the detection in order to also take into account the larger surrounding region of the frames, thus also considering the "context" and improving the matching and differentiation of points in the image.
- By observing the pairings of points between the two images we can surmise that the mismatches are indeed influenced by the changes in intensity, as areas appear to be more likely to be associated with other areas of similar intensity regardless of actual similarity. Using more sophisticated matching methods, such as those seen in previous exercise sessions, may yield better results.

Stage I.C

- The remaining mismatches appear to still be regions with intensity values very different from the rest of the image. Lowering the nnThreshold value results in less mismatches but might also risks missing some true positives. Using more advanced algorithms such as RANSAC may be more effective.

1.3.2 Part2:

- While the transformations move beyond affine to planar homographies, using the combination of SIFT descriptors and affine adaptation (producing elliptical descriptors that supplement SIFT's ability to match areas of the pictures) still yields much improved results compared to simply using SIFT, and still manages to produce acceptable matching when the transformation is not excessively extreme (with diminishing result as the warping worsens).

1.3.3 Part3:

Stage III.A

- The number of clusters affects the performance of the algorithm because a larger vocabulary will improve matching by ensuring that only the most similar features are grouped under the same cluster, but will also result in longer computations as each image has to be compared with a higher number of items.
- The time required to convert descriptors into visual words is not considered as it is low enough to not be significant, and the computation for the most part needs only to be carried out at the beginning.
- The computational time is for the most part related only to the number of clusters and their size, the latter of which may vary linearly with database size.

Stage III.B

- 14 images were erroneously matched. The first image likely has a score of ~1 because it is the same image used for the query, and thus it stands to reason that it would have maximum similarity with itself.

Stage III.C

- As the score does not anymore reflect a similarity measure (but instead the number of inliers) it does not have an upper limit of 1.
- Since we are merely rescoring the top 25 images the set of matched images is of course the same, but we can see their ordering is now much improved, with the correct matches being displayed first as they posess the highest scores.

[ ]:
```
```