CPSC-340 Machine Learning and Data Mining 2008

Homework # 2
Due Monday, Oct 6 1pm.

NAME:_____

Signature:_____

STD. NUM: _____

General guidelines for homeworks:

You are encouraged to discuss the problems with others in the class, but all write-ups are to be done on your own.
Homework grades will be based not only on getting the "correct answer," but also on good writing style and clear presentation of your solution. It is your responsibility to make sure that the graders can easily follow your line of reasoning.
Try every problem. Even if you can't solve the problem, you will receive partial credit for explaining why you got stuck on a promising line of attack. More importantly, you will get valuable feedback that will help you learn the material.
Please acknowledge the people with whom you discussed the problems and what sources you used to help you solve the problem (e.g. books from the library). This won't affect your grade but is important as academic honesty.
When dealing with python exercises, please attach a printout with all your code and show your results clearly.

1. **(Image compression)**

   Convert a photo of yours to .png format. Then load it with python and compute its SVD compression as explained in the lecture notes and in class. Choose a number of eigenvalues (truncation threshold) and EXPLAIN your choice below. Hand in the explanation, the original image and the compressed image.

   (i) Explanation:

   (ii) What is the compression factor?

2. **(Latent Semantic Indexing with the SVD)**

In this exercise, you will program a semantic search engine. The corpus of webpages is available at

`http://www.cs.ubc.ca/~nando/340-2008/homework.html`

You should download the data in pickle format (a useful format for files in python). The corpus consists of 100 webpages (urls). Each webpage has a few tags and their respective frequency of occurrence. The following code snipet loads the data:

```
from pylab import *
from scipy import *
import cPickle as pickle
from math import acos, degrees

def loadData():
    """
    INPUT: loads the file lsiData.pkl
    OUTPUT: - data: dict[URL] -> list of tuples(tag,frequency)
            data.keys()[i] returns the i-th URL.
            data.values()[i] returns the tuples(tag,frequency) for the i-th URL.
    """
    data = pickle.load(open("lsiData.pkl"))
    return data
```

Note that all the data is stored as a dictionary. You should familiarize yourself with these powerful data structures. To give you some assistance, the following function extracts the list of tags and websites from the dictionary:

```
def extractData(data):
    """
    INPUT: data
    OUTPUT: - url: list of urls
            - m: number of urls
            - n: number of distinct tags (vocabulary size)
            - tag: alphabetically sorted list of tags
    """
    url = data.keys()   # List of urls.
    m = len(url)        # Number of urls.
    tag = set()         # Initialize set of tags.
    for i,ithUrl in enumerate(url):
        tagPairs = data[ithUrl]
        tags = set([tagPairs[j][0] for j,jthPair in enumerate(tagPairs)])
        tag.update(tags) # add new tags to set of tags.
    tag = list(tag)      # convert the set of tags to a list of tags.
```

```
tag = sort(tag)      # sort tags in alphabetical order.
n = len(tag)
return url, tag, m, n
```

(i) Generate the term-frequency matrix for this dataset. As done in question 1, plot this matrix as an image. Remember to normalize the co-occurrence matrix.

(ii) Building the search engine: Compute the truncated SVD of the term-frequency matrix and truncate to an acceptable level. Here, again, you need to choose a reasonable truncation level and explain your choice. Plot the compressed matrix as an image. You must hand both images. Explanation:

(iii) Querying the search engine: Implement a program that takes as input one or more tags and returns the closest URLs to the query. Do this by using the truncated SVD of the previous part to project the query and compute the angle between the query and all the compressed webpages. Hand in all your code.

(iv) What are the top 5 webpages retrieved by the query "australia"?

(iv) What are the top 5 pages retrieved by the composite query "python animals"?