CAS CS 562: Advanced Database Applications

Homework #3
Fall 2020

Due Date: April 24, 2020 at 11:59PM, please submit a PDF and a ZIP file using Gradescope.

Problem 1. (FastMap) [20 Points]

Using the **FastMap** algorithm to embed (map) the following 5-dimensional points into 3-dimensional points. Use the Euclidian distance (L_2 norm) as the distance between the points in the original 5-dimensional space. Show the 3-d points of the mapping for each point.

```
p1: (5.1,3.5,1.4,0.2,3.0)
p2: (4.9,3.0,1.4,0.2,4.0)
p3: (4.7,3.2,1.3,0.2,2.0)
p4: (6.9,3.1,4.9,1.5,2.0)
p5: (5.5,2.3,4.0,1.3,3.0)
p6: (6.5,2.8,4.6,1.5,5.0)
p7: (7.2,3.0,5.8,1.6,7.0)
p8: (7.4,2.8,6.1,1.9,8.0)
```

Problem 2. (SVD and LSI) [50 Points]

In this problem you have to use some existing tools and write some code in order to implement LSI for text documents. The text documents are in:

http://www.cs.bu.edu/faculty/gkollios/cs562s20/datasets/Text/

We explain all the steps in detail next.

1) The first step is to remove the stop words from all the documents. A list of possible stop words can be found here: ”http://www.textfixer.com/resources/common-english-words.txt” in CSV format. You can use existing code (and you own list if you want) or write your own code for this.

2) The second step is to use a stemmer to extract the stems (roots) of each word. A very popular stemmer is the Porter stemmer. You can find many stemmers here: ”http://tartarus.org/martin/PorterStemmer/”. You can use any one that you want.
Hint: You should make slight modifications on the code. Do not print the results on the console. You could write the words to a file and each line is a word. An alternative is to use a stemmer from Python as we show below.

3) Write code to count the frequencies of non-stop words (stems) and construct a term-document matrix A1. For example:

	Doc1	Doc2	Doc3	Doc4
term1	4	1	0	0
term2	0	3	2	5
term3	3	1	4	4
term4	0	3	1	0
term5	1	2	0	4

Another representation is based on the tf*idf representation. Create another matrix A2 that uses this representation.

4) Use SVD to decompose document-term matrices A1 and A2 and create the LSI representations. You can use python (numpy.linalg.svd or scipy.linalg.svd), Matlab (svd) or any other library that you want to compute SVD.

5) We will provide a dataset with a set of documents in text format and you will have to create the LSI representation of this dataset. Explain what is the meaning of each matrix in the SVD decomposition and if you find some important concepts in the dataset. We will provide some query examples and you should report which documents should be retrieved for these queries. Notice that when you compute the similarity of a query and a document in the concept space, you need to normalize the vectors when you use cosine similarity. For each query report the top-3 results (documents).

You should also compare the results for A1 and A2 representations. Are the same or not?

5) Explain the steps that you did and the results of your experiments in a report. Put all your code and the report in a zip file and submit everything using Gradescope.

Hints and Suggestions:
Using Python: You can use either the Porter or the "Snowball" stemmer to extract the stems of the words.
You can read more at http://www.nltk.org and http://www.nltk.org/howto/stem.html.
You can also vectorize the dataset using this library as well and create the matrices A1 and A2.

Some useful code fragments are here:

```
from nltk.stem.snowball import SnowballStemmer
from nltk.tokenize import word_tokenize, sent_tokenize

stemmed_data = [" ".join(SnowballStemmer("english", ignore_stopwords
=True).stem(word)
         for sent in sent_tokenize(message)
        for word in word_tokenize(sent))
        for message in indata]
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer=TfidfVectorizer(stop_words='english',min_df=4,max_df=0.8)
dtm = vectorizer.fit_transform(indata)
```

Using Matlab:Matlab gives an easy way to compute SVD on a document-term matrix A.
(a) You can load the matrix from a CSV file by using the command ”csvread”. The help webpage is ”http://www.mathworks.com/help/matlab/ref/csvread.html”.
(b) Matlab includes SVD function, so you do not need to implement SVD by yourself. The function command is ”svd”. The help webpage is ”http://www.mathworks.com/help/matlab/ref/svd.html”.
(c) You can use the command: ”dlmwrite(filename,M)” to write the matrix M into a file ”filename”

Problem 3. (MapReduce and databases) [30 Points]

1) In the form of relational algebra implemented in SQL, relations are not sets, but bags; that is, tuples are allowed to appear more than once. There are extended definitions of union, intersection, and difference for bags, which we shall define below. Write MapReduce algorithms for computing the following operations on bags R and S:

(a) *Bag Union*, defined to be the bag of tuples in which tuple t appears the sum of the numbers of times it appears in R and S.
(b) *Bag Intersection*, defined to be the bag of tuples in which tuple t appears the minimum of the numbers of times it appears in R and S.
(c) *Bag Difference*, defined to be the bag of tuples in which the number of times a tuple t appears is equal to the number of times it appears in R minus the number of times it appears in S. A tuple that appears more times in S than in R does not appear in the difference.

2) The relational-algebra operation $R(A, B) \bowtie_{B<C} S(C, D)$ produces all tuples (a, b, c, d) such that tuple (a, b) is in relation R, tuple (c, d) is in S, and $b < c$. Give a MapReduce implementation of this operation, assuming R and S are sets.

What to submit

1) Put the answer to Problem 1 and Problem 3 in a PDF file and submit it using Gradescope.
2) Put the code and the report for Problem 2 in a ZIP file and submit it using Gradescope too.