DATA SCIENCE AND MACHINE LEARNING (MSC)

DAMA60: Algorithmic Techniques and Systems for Data Science and Machine Learning

Academic Year: 2023–2024

#4 Written Assignment	
Submission Deadline	**Wed, 27 March 2024 (23:59 EET)**
Student Name	***Kritikos Antonios***

Remarks

The deadline is definitive.

An indicative solution will be posted online along with the return of the graded assignments.

The assignment is due via the STUDY submission system. **You are expected to turn in a document (.DOC, .ODT, .PDF) and a file containing a Python program:**

- 1 document file (this document) with your name in the area indicated by "Student Name" and the answers to all the questions, along with the Python code snippets for Topic 5.
- 1 file with the complete Python program that answers Topic 5.

You should not make any changes in the written assignment file other than providing your own answers. You should also type all of your answers into Word or compatible word processors (such as Apache OpenOffice https://www.openoffice.org/ or LibreOffice https://www.libreoffice.org/) and not attach any handwritten notes as pictures into your work. Make sure to name all the files (DOC/DOCX/ODT files and Python program file) with **your last name first followed by a dash symbol and the names of each component at the end**. For example, for the student with last name Aggelou the files should be named as follows: Aggelou-HW4.doc (or Aggelou-HW4.pdf), Aggelou-Topic5.py.

Please ensure that all Python code you submit for this assignment opens, displays and executes successfully within the IDLE environment — the official Integrated Development and Learning Environment provided with Python 3. This consistency in the execution environment is necessary for a thorough and equitable evaluation of your work.

Topic	Points	Grades
1. Online Quiz	40	
2. Collaborative Filtering	15	
3. Recommender Systems Evaluation	15	
4. Similarity Search in Graphs with RWR algorithm	15	
5. Python Implementation of a Recommender System	15	
TOTAL	**100**	**/100**

Topic 1: Online Quiz

(40 points) Complete the corresponding online quiz available at:

https://study.eap.gr/mod/quiz/view.php?id=27178

You have one effort and unlimited time to complete the quiz, up to the submission deadline.

Topic 2: Collaborative Filtering

(15 total points) Table 1 shown below is a user-item rating matrix indicating how various users have rated street food companies. The rating is a so called Likert scale where 5 is the greatest rating and 0 the lowest. An empty cell indicates a missing value. The user-item rating matrix is used to implement an **item-based collaborative filtering** approach. Based on the presented user-item rating matrix, answer the next 3 questions:

Table 1: User-Item rating matrix

	The Last Slice	Karminio	Falafellas	Tarantino Sandwiches & Fries	Smak
Maria	2		4	2	4
Tim	4	5	3	4	2
Sonia	5	3	4	2	1
Peter	5	4	2	1	3
Eva	5	2	3	4	2

(a) (3 points) Calculate the average rating for Karminio by completing the missing value indicated by a question mark **?1** in the table below. Report the final result with 1 decimal place of accuracy (do not round up or down the number).

Your answer:

The Last Slice	Karmino	Falafellas	Tarantino Sandwiches & Fries	Smak
4.2	**3.5**	3.2	2.6	2.4

(b) (6 points) Compute the item-item similarity for the pairs shown in the box below and indicated by numbered question marks **?1**, **?2** and **?3**, using the Pearson correlation and Cosine similarity as the similarity measures. Do not normalize the ratings before calculating the cosine similarity measure. Report the results with 4 decimal places of accuracy (do not round up or down the numbers).

Your answer:

	Pearson correlation	***Cosine similarity***
The Last Slice, Falafellas	-0.5041	0.9075
The Last Slice, Tarantino Sandwiches & Fries	**0.0572**	**0.8813**
Falafellas, Tarantino Sandwiches & Fries	**0.0891**	0.8926

(c) (6 points) Assuming that the neighborhood size is 2, predict the rating of Maria for Karminio indicated by **?1** below, when item-based collaborative filtering is used based on the Cosine similarity as the similarity measure. For predicting the rating, use the formula 9.16 in section 9.2.2 of "*R. Zafarani, M.A. Abbasi & H. Liu (2014). Social media mining: an introduction. Cambridge University Press*". Do not normalize the ratings before calculating the similarities. Report the final result with 1 decimal place of accuracy (do not round up or down the number).

Your answer:

$$r_{Maria,\ Karminio} = \mathbf{4.3}$$

Topic 3: Recommender Systems Evaluation

(15 total points) Answer the following 2 questions.

(a) **(6 points)** Assume the following user-item rating matrix containing the subset of real ratings for books as reported by 3 different users. The rating is a Likert scale where 5 is the highest rating and 0 the lowest. In the matrix, empty cells indicate missing values:

	Dune	Solaris	Metro 2033	Foundation	The Hitchhiker's Guide to the Galaxy
Tom	5		4	4	5
Alice	4.5	4	3	4.5	
Mike		5	3	4	3

For the same users and items, an imaginary recommendation system has predicted the following ratings:

	Dune	Solaris	Metro 2033	Foundation	The Hitchhiker's Guide to the Galaxy
Tom	4	4	3	3	1
Alice	3	3.5	2	2	3
Mike	2	5	5	2	3

i. Calculate the Mean Absolute Error (MAE) for the predicted ratings[1]:

Your answer:

MAE = **1.12**

ii. Calculate the Root Mean Squared Error (RMSE) for the predicted ratings[1]:

Your answer:

RMSE = **1.40**

[1] Report the error with 2 decimal point accuracy. Do not round up or down the calculated value.

(b) **(9 points)** Let's assume the following top-8 ranking of items as produced by a recommendation system. The table displays also the relevance of each item (column "Relevance as specified by user") as specified manually by a user using a Likert scale where 5 is the highest relevance score and 0 the lowest. Make the assumption that a user liked a recommendation if s/he gave it a relevance score of 5. Any score smaller than 5 indicates that the user did not like the recommendation.

Fill all the cells containing the numbered question marks **?1**,**?2** and **?3** as simplified ratios with the correct values in columns Precision@N, Recall@N and DCG_{pos}.

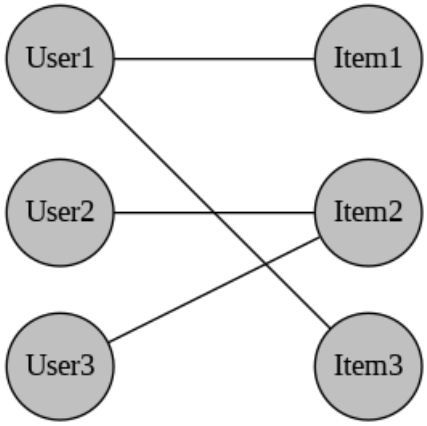
Your answer:

Top-N position	Relevance as specified by user	Precision@N	Recall@N	DCG_{pos}
1	*3*	*0*	*0*	*3*
2	*5*			
3	*5*	*2/3*		***0***
4	*4*		*2/3*	
5	*2*	***2/3***		
6	*3*			
7	*5*			
8	*1*		***0***	

Topic 4: Similarity Search in Graphs with RWR algorithm

(15 total points) Answer the following questions.

(a) **(12 points)** Assume the following graph representing users, items and affinities: an edge between a User and an Item indicates that the User# liked Item#.



Apply the algorithm Random Walk with Restart to find the similarity/relevance of node named **Item 2** to all other nodes in the graph after 4 (four) steps with parameter ***β*** **set to 0.9** and assuming that nodes are indexed in the following way:

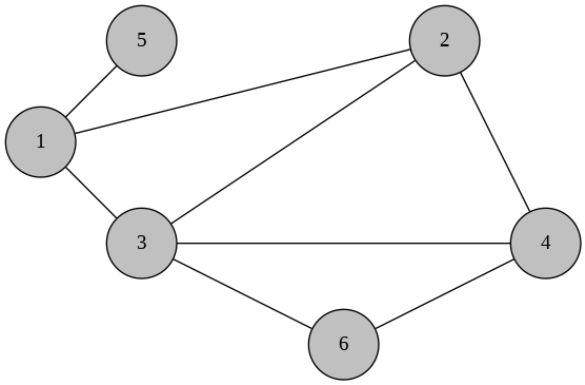
Node	Index
User1	1
User2	2
User3	3
Item1	4
Item2	5
Item3	6

Calculate the 6 missing values indicated by numbered question marks **?1**, **?2**, **?3** etc. in the vectors at each step below, and fill in the associated cells in the following table. Use an accuracy of 4 decimal digits where necessary.

Your answer:

Initial step	Step 1	Step 2	Step 3	Step 4
$\begin{bmatrix}0\\0\\\mathbf{0}\\0\\\mathbf{1}\\0\end{bmatrix}$	$\begin{bmatrix}0\\0.45\\0.45\\0\\\mathbf{0.1}\\0\end{bmatrix}$	$\begin{bmatrix}0\\0.045\\\mathbf{0.045}\\0\\0.91\\0\end{bmatrix}$	$\begin{bmatrix}0\\0.4095\\0.4095\\0\\\mathbf{0.181}\\0\end{bmatrix}$	$\begin{bmatrix}0\\0.0814\\0.0814\\0\\\mathbf{0.8371}\\0\end{bmatrix}$

(b) **(3 points)** For the graph shown below, calculate for each edge its contribution to the betweenness when considering only the shortest paths starting from node 5.
Use the algorithm described in section 10.2.4 of "*J. Leskovec, A. Rajaraman & J.D. Ullman (2020).* Mining of Massive Datasets *(3rd edition). Cambridge University Press*". Apply the algorithm only for node 5 (i.e. starting only from node 5) and fill in **the missing values** (indicated by numbered question marks **?1**, etc) in the following table Report the final result with 1 decimal place of accuracy (do not round up or down the number):



Your answer:

Edge	Contribution
e(3,4)	0.5
e(1,3)	**0.5**
e(2,4)	0.5
e(1,2)	**1**
e(1,5)	**1**

Topic 5: Python Implementation of a Recommender System

(15 total points) In this topic you will use libraries of Python to implement a trivial content-based recommender system for movies.

Movies are represented using their titles and a natural language description of the genres they belong to (called the description). The content-based recommendation system calculates similarities of movies based on the cosine similarity of their descriptions. Before calculating similarities, descriptions for movies are vectorized using the TF-IDF (Term Frequency - Inverse Document Frequency) score. File Topic5.py contains the incomplete Python code that you have to use to fill in your answers. Make sure that you have installed version 1.0.0 or later of the scikit-learn package in order to avoid potential warnings[2].

(a) (**4 points**) Complete the following Python code, by filling the 2 gaps indicated by the numbered question marks, to calculate the TF-IDF scores for movie descriptions.

Your answer:

```
# Required libraries
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
from operator import itemgetter # for sorting the list of tuples

# List of movie titles with their respective
# descriptions. This is the movie dataset where
# description is the content.
# For example, data['title'][0] has as description data['description'][0]
data = {'title': ['Predator','The Thing','The Lobster','Edge of tomorrow','Eyes wide
shut','Alien','Godzilla minus one'],
        'description': ['action thriller scifi','horror scifi','romance thriller scifi','scifi
action','thriller mystery','scifi horror','horror scifi'] }

# Instantiate a TF-IDF object and calculate the
# TF-IDF score for each term (word) found in descriptions
tfidfVectorizer = TfidfVectorizer(stop_words='english')
tfidfMatrix = tfidfVectorizer.?1( ?2 )

# Display the TF-IDF score for each word in
# each description in a human friendly way.
print('a) TF-IDF scores')
# Adjust number of decimal digits for floating point
# numbers in data frame.
pd.set_option('display.float_format', '{:.4f}'.format)
print( pd.DataFrame(tfidfMatrix.toarray(), index=data['title'],
                    columns=list(tfidfVectorizer.get_feature_names_out()) ) )
```

[2] You can see the version number of the installed `scikit-learn` package on your system by executing `print(sklearn.__version__)` (NOTE: double leading and trailing underscores) on the IDLE shell or from within a Python script after importing the module.

(b) (**2 points**) Calculate the cosine similarities for each pair of movie descriptions by filling the single gap indicated by the numbered question mark:

Your answer:

```
# Calculate cosine similarities for each pair
# of movies.
similarities = cosine_similarity(tfidfMatrix)
# Adjust number of decimal points
np.set_printoptions(precision=4)
print('\nb) Pairwise output of cosine similarities:\n{}\n'.format(similarities))
```

(c) (**3 points**) After having calculated cosine similarities, fill appropriately the 3 gaps below indicated by the numbered question marks, in order to recommend movies most similar to a specific one defined here by the `targetMovie` variable. Below, and just as an example, movies with the greatest cosine similarity to 'The Lobster' will be recommended:

Your answer:

```
# Get recommendations for this movie.
# Case sensitivity matters.
targetMovie = 'The Lobster'

# Get only relevant similarities from the
# calculated similarity matrix as a list
idx = data['title'].index(targetMovie)
simList = list( similarities[:, idx ] )

# Add (movie index, similarity) to the list
# of candidate recommendations
candidateRecommendations = []
for i, s in enumerate(simList):
    # ignore target movie
    if i != idx:
       candidateRecommendations.append( (i, s) )

# Inplace sorting of movies in descending order
# of cosine similarity
candidateRecommendations.sort(key=itemgetter(1), reverse=True)


# Print out recommendations formatted using title,
# similarity in descending order of similarity.

print(f'c) Recommendations for movie "{targetMovie}":')
# Print a header
print('\t{:<20}|{:<10}'.format('Title', 'Similarity') )
print('\t{:<20}|{:<10}'.format(20*'-', 10*'-') )
# Print movie title and similarity
for m in candidateRecommendations:
    movieTitlePos, movieSimilarity = m
    print('\t{:<20}|{:<10.4f}'.format(data['title'][ movieTitlePos ], movieSimilarity ) )
```

(**d**) (**3 points**) Follow the guidelines on the first page of this assignment and incorporate the .py file that answers this topic into your submission.

Your answer:

The code in your .py file with name **Kritikos-Topic5** conforms to the module's directions, and opens and displays correctly in IDLE.

(**e**) (**3 points**) Follow the guidelines on the first page of this assignment and incorporate the .py file that answers this topic into your submission.

Your answer:

Your code executes correctly when run from your .py file with name **Kritikos-Topic5** using IDLE.