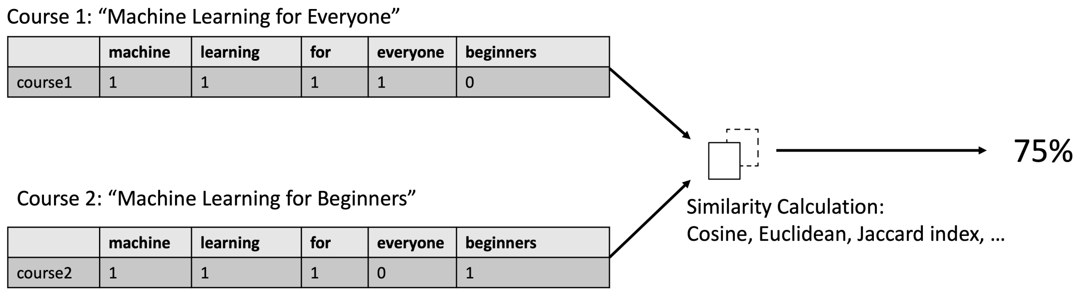
Skills Network

Content-based Course Recommender System using Course Similarities

Estimated time needed: **45** minutes

In one of the previous lab, you have learned and practiced how to calculate the similarity between two courses using Bag of Words (BoW) features. For example, the similarity between course1 `Machine Learning for Everyone` and course2 `Machine Learning for Beginners` are `75%` as shown below.



Course 1: “Machine Learning for Everyone”

	machine	learning	for	everyone	beginners
course1	1	1	1	1	0

Course 2: “Machine Learning for Beginners”

	machine	learning	for	everyone	beginners
course2	1	1	1	0	1

As we mentioned before, the content-based recommender system is highly based on the similarity calculation among items. The similarity or closeness of items is measured based on the similarity in the content or features of those items. The course genres are important features, and in addition to that, the BoW value is another important type of feature to represent course textual content.

In this lab, you will apply the course similarities metric to recommend new courses which are similar to a user's presently enrolled courses.

Objectives

After completing this lab you will be able to:

- Obtain the similarity between courses from a course similarity matrix
- Use the course similarity matrix to find and recommend new courses which are similar to enrolled courses

---

Prepare and setup lab environment

Let's first install and import the required libraries:

In [1]:

```
!pip install seaborn
!pip install numpy
!pip install pandas
!pip install matplotlib
```

```
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Collecting numpy!=1.24.0,>=1.20 (from seaborn)
  Downloading numpy-2.4.1-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_6
4.whl.metadata (6.6 kB)
Collecting pandas>=1.2 (from seaborn)
  Downloading pandas-3.0.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_6
4.whl.metadata (79 kB)
Collecting matplotlib!=3.6.1,>=3.4 (from seaborn)
  Downloading matplotlib-3.10.8-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x
86_64.whl.metadata (52 kB)
Collecting contourpy>=1.0.1 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading contourpy-1.3.3-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x8
6_64.whl.metadata (5.5 kB)
Collecting cycler>=0.10 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading fonttools-4.61.1-cp312-cp312-manylinux1_x86_64.manylinux2014_x86_6
4.manylinux_2_17_x86_64.manylinux_2_5_x86_64.whl.metadata (114 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading kiwisolver-1.4.9-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x8
6_64.whl.metadata (6.3 kB)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.12/site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.2)
Collecting pillow>=8 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading pillow-12.1.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_
64.whl.metadata (8.8 kB)
Collecting pyparsing>=3 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading pyparsing-3.3.2-py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.12/
site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-package
s (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
Downloading matplotlib-3.10.8-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86
_64.whl (8.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.7/8.7 MB 111.6 MB/s eta 0:00:00
Downloading numpy-2.4.1-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.w
hl (16.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.4/16.4 MB 139.9 MB/s eta 0:00:00
Downloading pandas-3.0.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.
whl (10.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10.9/10.9 MB 125.1 MB/s eta 0:00:00
Downloading contourpy-1.3.3-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_
64.whl (362 kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.61.1-cp312-cp312-manylinux1_x86_64.manylinux2014_x86_64.m
anylinux_2_17_x86_64.manylinux_2_5_x86_64.whl (5.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.0/5.0 MB 106.1 MB/s eta 0:00:00
Downloading kiwisolver-1.4.9-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_
64.whl (1.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 57.0 MB/s eta 0:00:00
Downloading pillow-12.1.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_6
4.whl (7.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.0/7.0 MB 102.2 MB/s eta 0:00:00
Downloading pyparsing-3.3.2-py3-none-any.whl (122 kB)
Installing collected packages: pyparsing, pillow, numpy, kiwisolver, fonttools, c
ycler, pandas, contourpy, matplotlib, seaborn
Successfully installed contourpy-1.3.3 cycler-0.12.1 fonttools-4.61.1 kiwisolver-
1.4.9 matplotlib-3.10.8 numpy-2.4.1 pandas-3.0.0 pillow-12.1.0 pyparsing-3.3.2 se
```

```
aborn-0.13.2
Requirement already satisfied: numpy in /opt/conda/lib/python3.12/site-packages
(2.4.1)
Requirement already satisfied: pandas in /opt/conda/lib/python3.12/site-packages
(3.0.0)
Requirement already satisfied: numpy>=1.26.0 in /opt/conda/lib/python3.12/site-pa
ckages (from pandas) (2.4.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.1
2/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-package
s (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.12/site-packa
ges (3.10.8)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.12/site
-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.12/site-pac
kages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.12/sit
e-packages (from matplotlib) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/conda/lib/python3.12/sit
e-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in /opt/conda/lib/python3.12/site-pack
ages (from matplotlib) (2.4.1)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.12/site-
packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.12/site-packag
es (from matplotlib) (12.1.0)
Requirement already satisfied: pyparsing>=3 in /opt/conda/lib/python3.12/site-pac
kages (from matplotlib) (3.3.2)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.12/
site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-package
s (from python-dateutil>=2.7->matplotlib) (1.17.0)
```

In [2]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

%matplotlib inline
```

In [3]:

```
# also set a random state
rs = 123
```

Next, let's load a pre-made course similarity matrix. If you are interested, you could easily calculate such a similarity matrix by iterating through all possible course pairs and calculating their similarities.

In [4]:

```
sim_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IB
```

In [5]:

```
sim_df = pd.read_csv(sim_url)
sim_df
```

Out[5]:

	0	1	2	3	4	5	6	7	
0	1.000000	0.088889	0.088475	0.065556	0.048810	0.104685	0.065202	0.143346	0.0
1	0.088889	1.000000	0.055202	0.057264	0.012182	0.078379	0.032545	0.119251	0.0
2	0.088475	0.055202	1.000000	0.026463	0.039406	0.000000	0.000000	0.154303	0.0
3	0.065556	0.057264	0.026463	1.000000	0.000000	0.250490	0.390038	0.000000	0.0
4	0.048810	0.012182	0.039406	0.000000	1.000000	0.000000	0.000000	0.085126	0.0
...	...	...	...	...	...	...	...	...	
302	0.033944	0.028239	0.018270	0.094759	0.060474	0.064851	0.053856	0.039467	0.0
303	0.076825	0.063911	0.082698	0.030638	0.030415	0.000000	0.000000	0.119098	0.0
304	0.072898	0.138270	0.133400	0.017443	0.129871	0.009285	0.000000	0.254274	0.0
305	0.039276	0.031367	0.012684	0.018796	0.000000	0.015008	0.024926	0.082199	0.0
306	0.121113	0.076940	0.000000	0.158073	0.000000	0.126211	0.157219	0.000000	0.1

307 rows × 307 columns

The similarity matrix is a real number, symmetric metric with each element representing the similarity value (ranged 0 to 1) between course index `i` and course index `j`.

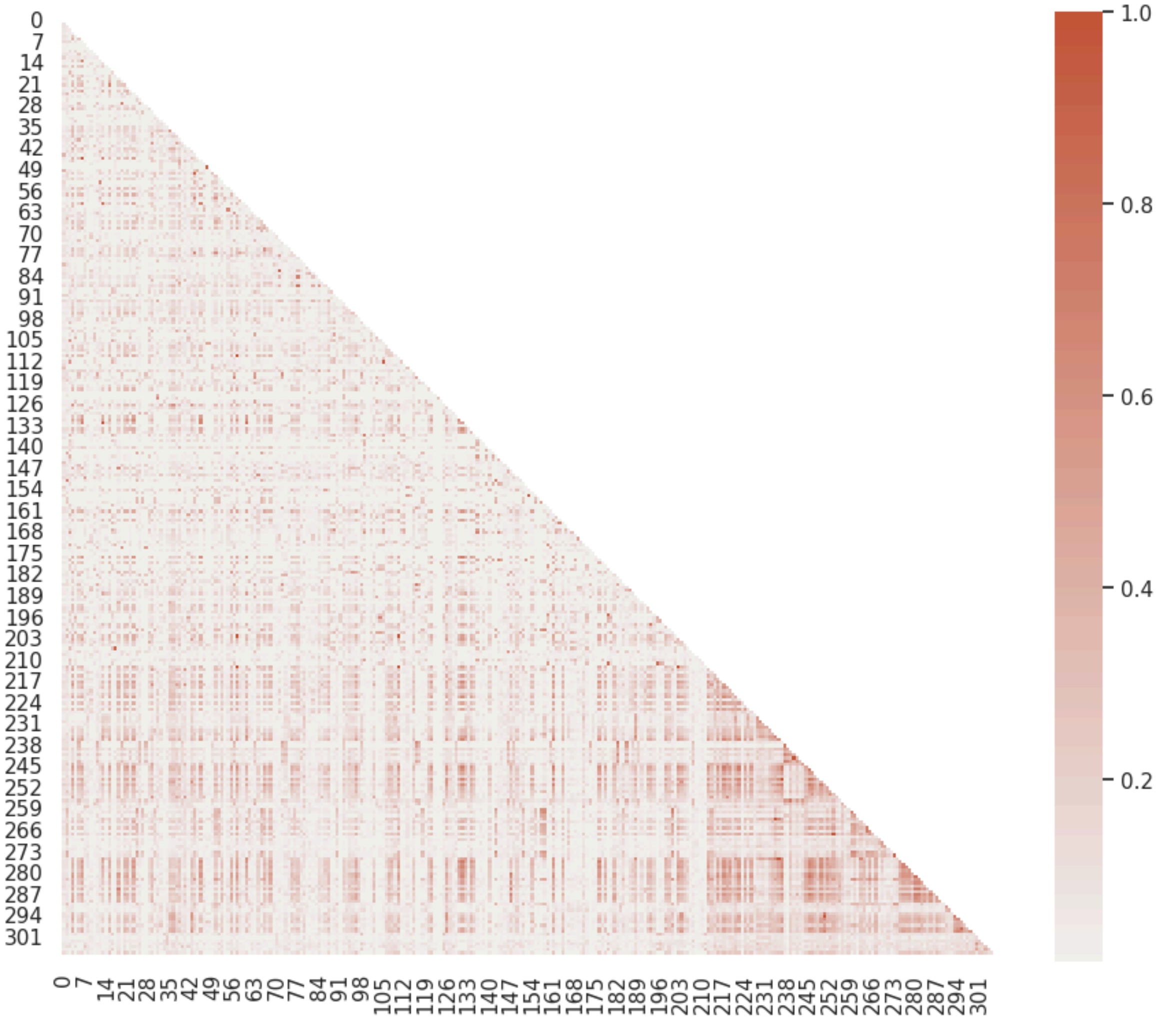
We could use `seaborn` to visualize the similarity metric, and since it is symmetric, we can just show the triangular matrix (lower left):

In [6]:

```
# Configure seaborn to set the plot style to 'white'
sns.set_theme(style="white")

# Create a mask for the upper triangle of the similarity matrix
mask = np.triu(np.ones_like(sim_df, dtype=bool))

# Create a new figure and axis for the heatmap
_, ax = plt.subplots(figsize=(11, 9))

# Create a diverging color palette for the heatmap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Plot a similarity heat map using seaborn's heatmap function
sns.heatmap(sim_df, mask=mask, cmap=cmap, vmin=0.01, vmax=1, center=0,
            square=True)
```

Out[6]:

```
<Axes: >
```



As we can see from the heatmap; there are many hot spots, which means many courses are similar to each other. Such patterns suggest that it is possible to build a recommender system based on course similarities.

Let's take a look at a quick example:

In [7]:
```
# Let's first load the course content and BoW dataset
course_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud
course_df = pd.read_csv(course_url)
bow_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IB
bow_df = pd.read_csv(bow_url)
```

In [8]:
```
bow_df.head()
```

Out[8]:

	doc_index	doc_id	token	bow
0	0	ML0201EN	ai	2
1	0	ML0201EN	apps	2
2	0	ML0201EN	build	2
3	0	ML0201EN	cloud	1
4	0	ML0201EN	coming	1

First, we want to mention that the matrix indices are course indices (such as `0, 1, 2, 3`). Very often we need to query the actual course ids (such as `ML0151EN` and `ML0101ENv3`) based on course indices and vice versa. We can save the course id's and indices into two dictionaries for late queries:

Then, based on the `doc_index` and `doc_id` columns, we create an index to id mapping and another id to index mapping in two Python dictionaries:

In [9]:
```
# Create course id to index and index to id mappings
def get_doc_dicts(bow_df):
    # Group the DataFrame by course index and ID, and get the maximum value for
    grouped_df = bow_df.groupby(['doc_index', 'doc_id']).max().reset_index(drop=
    # Create a dictionary mapping indices to course IDs
    idx_id_dict = grouped_df[['doc_id']].to_dict()['doc_id']
    # Create a dictionary mapping course IDs to indices
    id_idx_dict = {v: k for k, v in idx_id_dict.items()}
    # Clean up temporary DataFrame
    del grouped_df
    return idx_id_dict, id_idx_dict
```

Now suppose we have two example courses:

In [10]:
```
course1 = course_df[course_df['COURSE_ID'] == "ML0151EN"]
course1
```

Out[10]:

	COURSE_ID	TITLE	DESCRIPTION
200	ML0151EN	machine learning with r	this machine learning with r course dives into...

In [11]:
```
course2 = course_df[course_df['COURSE_ID'] == "ML0101ENv3"]
course2
```

Out[11]:

	COURSE_ID	TITLE	DESCRIPTION
158	ML0101ENv3	machine learning with python	machine learning can be an incredibly benefici...

From their titles we can see they are all about machine learning. As such, they should be very similar to each other. Let's try to find their similarity in the similarity matrix.

With their course ids, we can use the `id_idx_dict` dictionary to query their row and column index on the similarity matrix:

In [12]:
```
idx_id_dict, id_idx_dict = get_doc_dicts(bow_df)
idx1 = id_idx_dict["ML0151EN"]
idx2 = id_idx_dict["ML0101ENv3"]
print(f"Course 1's index is {idx1} and Course 2's index is {idx2}")
```

```
Course 1's index is 200 and Course 2's index is 158
```

Then we can locate their similarity value in row 200 and col 158, `sim_matrix[200][158]`:

In [13]:

```
sim_matrix = sim_df.to_numpy()
```

In [14]:

```
sim = sim_matrix[idx1][idx2]
sim
```

Out[14]:

```
np.float64(0.6626221399549089)
```

It's about 66% meaning these two courses are quite similar to each other.

TASK: Find courses which are similar enough to your enrolled courses.

Now you know how to easily use the pre-computed similarity matrix to query the similarity between any two courses. Do you want to make some course recommendations for yourself?

Let's assume you are an end-user of the online course platform and already audited or completed some courses previously. Next, you expect the system would recommend similar courses based on your enrollments history.

From the full course list, choose any courses that may interest you, such as those machine learning related courses:

In [15]:

```
pd.set_option('display.max_rows', None)
pd.set_option('max_colwidth', None)
course_df[['COURSE_ID', 'TITLE']]
```

Out[15]:

	COURSE_ID	TITLE
0	ML0201EN	robots are coming build iot apps with watson swift and node red
1	ML0122EN	accelerating deep learning with gpu
2	GPXX0ZG0EN	consuming restful services using the reactive jax rs client
3	RP0105EN	analyzing big data in r using apache spark
4	GPXX0Z2PEN	containerizing packaging and running a spring boot application
5	CNSC02EN	cloud native security conference data security
6	DX0106EN	data science bootcamp with r for university proffesors
7	GPXX0FTCEN	learn how to use docker containers for iterative development
8	RAVSCTEST1	scorm test 1
9	GPXX06RFEN	create your first mongodb database
10	GPXX0SDXEN	testing microservices with the arquillian managed container
11	CC0271EN	cloud pak for integration essentials
12	WA0103EN	watson analytics for social media
13	DX0108EN	data science bootcamp with python for university professors advance
14	GPXX0PICEN	create a cryptocurrency trading algorithm in python
15	DAI101EN	data ai essentials
16	GPXX0W7KEN	securing java microservices with eclipse microprofile json web token microprofile jwt
17	GPXX0QR3EN	enabling distributed tracing in microservices with zipkin
18	BD0145EN	sql access for hadoop
19	HCC105EN	hybrid cloud conference ai pipelines lab
20	DE0205EN	dataops methodology
21	DS0132EN	data ai jumpstart your journey
22	OS0101EN	introduction to open source
23	DS0201EN	end to end data science on cloudpak for data
24	BENTEST4	ai for everyone master the basics
25	CC0210EN	serverless computing using cloud functions developer i
26	PA0103EN	predicting customer satisfaction
27	HCC104EN	hybrid cloud conference serverless lab
28	GPXX0A1YEN	validating constraints for javabeans in java microservices
29	TMP0105EN	getting started with the data apache spark makers build
30	PA0107EN	predicting financial performance of a company
31	DB0113EN	db2 fundamentals i

	COURSE_ID	TITLE
32	PA0109EN	using clustering methods for investment portfolio analysis
33	PHPM002EN	php web application on a lamp stack
34	GPXX03HFEN	fundamentals of javascript through rock paper scissors
35	RP0103	using r with databases
36	RP0103EN	using r with databases
37	BD0212EN	spark fundamentals ii
38	GPXX0IBEN	data science in insurance basic statistical analysis
39	SECM03EN	apply end to end security to a cloud application
40	SC0103EN	spark overview for scala analytics
41	GPXX0YXHEN	testing a microprofile or jakarta ee application using microshed testing with an open liberty docker container
42	RP0151EN	r 101
43	TA0105	text analytics 101
44	SW0201EN	how to build watson ai and swift apis and make money
45	TMP0106	data science bootcamp
46	GPXX0BUBEN	insurance risk assessment with montecarlo method using apache spark
47	ST0201EN	statistics 201
48	ST0301EN	statistics 301
49	SW0101EN	build swift mobile apps with watson ai services
50	TMP0101EN	text analysis
51	DW0101EN	introduction to machine learning with sound
52	BD0143EN	using hbase for real time access to your big data
53	WA0101EN	watson analytics 101
54	GPXX04HEEN	insurance business modelling and basic actuarial calculations
55	BD0141EN	accessing hadoop data using hive
56	CO0401EN	beyond the basics istio and ibm cloud kubernetes service
57	ML0122ENv1	accelerating deep learning with gpu
58	BD0151EN	text analytics 101
59	TA0106EN	text analytics at scale
60	TMP107	data science bootcamp with python
61	ML0111EN	machine learning with apache systemml
62	GPXX048OEN	action classification task based on internet firewall logs
63	CO0201EN	container kubernetes essentials with ibm cloud

	COURSE_ID	TITLE
64	GPXX01DCEN	data science in health care advanced prognostication using neural networks
65	GPXX04XJEN	advanced machine deep learning for spam classification task
66	GPXX0JZ4EN	visual data analysis in banking
67	GPXX0ZYVEN	secure analysis of credit card dataset
68	GPXX0ZMZEN	data science in health care advanced machine learning classification
69	GPXX0742EN	network traffic anomaly detection intrusion detection task
70	GPXX0KV4EN	getting started with node js
71	GPXX01RYEN	getting started with mysql command line
72	CC0120EN	an introduction to ibm cloud satellite
73	QC0101EN	introduction to quantum computing
74	GPXX0YMEEN	launch an ai hotdog detector as a serverless python app
75	GPXX0Q8AEN	exploratory data analysis eda with pandas in banking
76	TA0105EN	text analytics 101
77	GPXX0XFQEN	create tables and load data in mysql using phpmyadmin
78	GPXX07REN	relational model concepts
79	PA0101EN	predictive modeling fundamentals i
80	BC0202EN	build an iot blockchain network for a supply chain
81	BC0101EN	blockchain essentials
82	GPXX0725EN	getting started with postgresql command line
83	BC0201EN	ibm blockchain foundation developer
84	GPXX0MIIEN	keys and constraints in mysql
85	BD0223EN	exploring spark s graphx
86	GPXX0435EN	data science in health care basic statistical analysis
87	GPXX06ZLEN	create tables and load data in postgresql using pgadmin
88	GPXX0QS6EN	monitoring the metrics of java microservices using eclipse microprofile metrics
89	GPXX07YGEN	configuring microservices running in kubernetes
90	AI0111EN	game playing ai with swift for tensorflow s4tf
91	GPXX0QJFEN	enabling cross origin resource sharing cors in a restful java microservice
92	GPXX0LLEEN	getting started with db2 on cloud
93	BD0121EN	apache pig 101
94	BD0135EN	developing distributed applications using zookeeper

	COURSE_ID	TITLE
95	BD0133EN	controlling hadoop jobs using oozie
96	BD0131EN	moving data into hadoop
97	BD0115EN	mapreduce and yarn
98	GPXX04V3EN	deploy a web server using python and ibm cloud engine
99	GPXX05P1EN	acknowledging messages using microprofile reactive messaging
100	GPXX0D14EN	build a personal movie recommender with django
101	GPXX0G3KEN	managing and injecting dependencies into java microservices using contexts and dependency injection cdi
102	BD0211EN	spark fundamentals i
103	GPXX0HZ2EN	deploying microservices to kubernetes
104	GPXX04TNEN	getting started with open liberty
105	GPXX0IHMEN	consuming restful java microservices asynchronously using eclipse microprofile rest client
106	ST0101EN	statistics 101
107	BD0153EN	analyzing big data with a spreadsheet ui
108	DP0101EN	openrefine 101
109	BD0137EN	solr 101
110	ML0120EN	deep learning with tensorflow
111	GPXX0E3QEN	building fault tolerant microservices with the fallback annotation
112	GPXX0XV3EN	consuming restful java microservices with template interfaces using eclipse microprofile rest client
113	CB0201EN	build chatbots with watson assistant
114	CB0105ENv1	node red basics to bots
115	GPXX0NHZEN	normalization keys constraints in relational database
116	CL0101EN	ibm cloud essentials
117	CO0301EN	getting started with microservices with istio and ibm cloud kubernetes service
118	GPXX0XENEN	playing tictactoe with reinforcement learning and openai gym
119	GPXX0HAAEN	deploying a microservice to openshift by using a kubernetes operator
120	DA0201EN	data analysis demos
121	DJ0101EN	data journalism first steps skills and tools
122	CP0101EN	mathematical optimization for business problems
123	DB0111EN	db2 academic training
124	DB0115EN	db2 fundamentals ii

	COURSE_ID	TITLE
125	COM001EN	scalable web applications on kubernetes
126	DA0151EN	data analysis using r 101
127	DB0151EN	nosql and dbaas 101
128	CO0302EN	kubernetes operators advanced
129	GPXX0T0FEN	project deploy a serverless app for image processing
130	DS0107	data science career talks
131	DS0110EN	data science with open data
132	DX0107EN	data science bootcamp with python for university professors
133	DS0321EN	bitcoin 101
134	DS0105EN	data science hands on with open source tools
135	DS0103EN	data science methodology
136	GPXX0I4FEN	creating asynchronous java microservices using microprofile reactive messaging
137	GPXX06KEEN	build a smart search form with algolia
138	GPXX0YBFEN	documenting restful apis using microprofile openapi
139	LB0109ENv1	reactive architecture distributed messaging patterns
140	GPXX0KHHEN	data science in agriculture land use classification
141	LB0111EN	reactive architecture cqrs event sourcing
142	ML0115EN	deep learning 101
143	LB0105ENv1	reactive architecture reactive microservices
144	LB0103ENv1	reactive architecture domain driven design
145	IT0101EN	building robots with tjbot
146	CC0250EN	building cloud native and multicloud applications
147	BD0101EN	big data 101
148	EE0101EN	modernizing java ee applications
149	GPXX04P5EN	data science in agriculture prognostication using by neural network
150	CO0101EN	docker essentials a developer introduction
151	ML0122ENv3	accelerating deep learning with gpus
152	LB0107ENv1	reactive architecture building scalable systems
153	CB0101EN	build your own chatbots
154	GPXX0PG8EN	data science in agriculture basic statistical analysis and geo visualisation
155	ML0101EN	machine learning with python
156	CB0103EN	build your own chatbot

	COURSE_ID	TITLE
157	ML0109EN	machine learning dimensionality reduction
158	ML0101ENv3	machine learning with python
159	GPXX0M7ZEN	consuming a restful java web service using json b and json p
160	CC0121EN	an introduction to ibm cloud for financial services
161	DA0101EN	data analysis with python
162	GPXX07UGEN	testing reactive java microservices using microshed testing framework
163	GPXX0QU9EN	checking the health of java microservices by using kubernetes readiness and liveness probes
164	SC0101EN	scala 101
165	GPXX0QTEEN	checking the health of java microservices by using eclipse microprofile health check
166	GPXX0WTIEN	train a hotdog image recognition model with python
167	GPXX08WYEN	externalizing configuration for java microservices using eclipse microprofile config
168	GPXX0UMSEN	integrating restful services with a reactive system
169	GPXX0M6UEN	using the cql shell to execute keyspace operations in cassandra
170	GPXX097UEN	performing table and crud operations with cassandra
171	SN0111EN	how to create and publish guided projects and hands on labs
172	GPXX0JGFEN	building a simple restful java microservice using jax rs and json b
173	CO0193EN	hybrid cloud conference backend services for containers
174	GPXX0HC7EN	transform photos to sketches and paintings with opencv
175	GPXX04MXEN	quick introduction to a b testing
176	DS0101EN	introduction to data science
177	LB0101ENv1	reactive architecture introduction to reactive systems
178	DS0301EN	data privacy fundamentals
179	GPXX0RL8EN	consuming a restful java web service with angularjs
180	GPXX05LMEN	implement consumer driven contract testing for java microservices using the pact framework
181	BD0111EN	hadoop 101
182	CC0101EN	introduction to cloud
183	GPXX0UN5EN	classification of yelp reviews using sentiment analysis
184	BD0221EN	spark mllib
185	CC0103EN	ibm cloud essentials v3
186	GPXX0TY1EN	performing database operations in the cloudant dashboard

	COURSE_ID	TITLE
187	GPXX0T3CEN	working with databases in ibm cloudant
188	PY0101EN	python for data science
189	CC0201EN	introduction to containers kubernetes and openshift v2
190	BD0123EN	simplifying data pipelines with apache kafka
191	DB0101EN	sql and relational databases 101
192	GPXX05RDEN	medical appointment data analysis
193	DV0151EN	data visualization with r
194	GPXX0BSAEN	implementing a graphql microservice using microprofile api to query and update data from multiple services
195	GPXX0G81EN	consuming a restful java web service with reactjs
196	GPXX0FFCEN	building and testing a java web application with maven and open liberty
197	GPXX0MP0EN	building a hypermedia driven restful java microservice using hypermedia as the engine of application state hateoas
198	DV0101EN	data visualization with python
199	ML0103EN	digital analytics regression
200	ML0151EN	machine learning with r
201	GPXX01AVEN	containerizing and running java microservices in docker containers
202	ML0120ENv2	deep learning with tensorflow
203	RP0101EN	r for data science
204	GPXX0G31EN	accessing and persisting data in microservices using java persistence api jpa
205	GPXX0KY1EN	data science in health care basic prognostication and geo visualization
206	GPXX0JLHEN	enabling distributed tracing in java microservices using eclipse microprofile opentracing and the jaeger tracing system
207	CC0150EN	building cloud native and multicloud applications
208	GPXX0QQ3EN	deploy an ai powered discord bot with a voice
209	GPXX0RQLEN	views in postgresql
210	GPXX0WRDEN	streaming updates from a microprofile reactive messaging microservice using server sent events sse
211	GPXX0ADEN	consuming a restful java web service with angular
212	ML0120ENv3	deep learning with tensorflow
213	SC0105EN	data science with scala
214	excourse01	relational database systems
215	excourse02	business intelligence and data warehousing

	COURSE_ID	TITLE
216	excourse03	nosql systems
217	excourse04	\nsql for data science
218	excourse05	\ndistributed computing with spark sql
219	excourse06	\nsql for data science capstone project
220	excourse07	database management essentials
221	excourse08	using databases with python
222	excourse09	process data from dirty to clean
223	excourse10	database architecture scale and nosql with elasticsearch
224	excourse11	the nature of data and relational database design
225	excourse12	python scripting files inheritance and databases
226	excourse13	relational database support for data warehouses
227	excourse14	introduction to structured query language sql
228	excourse15	crash course on python
229	excourse16	programming for everybody getting started with python
230	excourse17	python basics
231	excourse18	python programming a concise introduction
232	excourse19	introduction to python programming
233	excourse20	python and statistics for financial analysis
234	excourse21	applied machine learning in python
235	excourse22	introduction to data science in python
236	excourse23	data analysis using python
237	excourse24	introduction to cloud computing
238	excourse25	cloud computing basics cloud 101
239	excourse26	cloud computing foundations
240	excourse27	cloud computing concepts part 1
241	excourse28	fundamentals of cloud computing
242	excourse29	cloud computing concepts part 2
243	excourse30	cloud computing applications part 1 cloud systems and infrastructure
244	excourse31	cloud computing applications part 2 big data and applications in the cloud
245	excourse32	introduction to data analytics
246	excourse33	excel basics for data analysis
247	excourse34	introduction to data analysis

	COURSE_ID	TITLE
248	excourse35	introduction to predictive modeling
249	excourse36	data analysis using python
250	excourse37	data analysis with r programming
251	excourse38	data analysis with python
252	excourse39	excel fundamentals for data analysis
253	excourse40	exploratory data analysis for machine learning
254	excourse41	introduction to data analysis using excel
255	excourse42	big data analysis hive spark sql dataframes and graphframes
256	excourse43	cloud virtualization containers and apis
257	excourse44	alibaba cloud native solutions and container service
258	excourse45	docker basics
259	excourse46	machine learning
260	excourse47	machine learning for all
261	excourse48	introduction to machine learning language processing
262	excourse49	applied machine learning in python
263	excourse50	build train and deploy ml pipelines using bert
264	excourse51	introduction to machine learning in production
265	excourse52	machine learning data lifecycle in production
266	excourse53	deploying machine learning models in production
267	excourse54	exploratory data analysis for machine learning
268	excourse55	advanced computer vision with tensorflow
269	excourse56	deep learning applications for computer vision
270	excourse57	deep learning in computer vision
271	excourse58	computer vision basics
272	excourse59	fundamentals of digital image and video processing
273	excourse60	introduction to tensorflow for artificial intelligence machine learning and deep learning
274	excourse61	convolutional neural networks in tensorflow
275	excourse62	introduction to data science in python
276	excourse63	a crash course in data science
277	excourse64	data science in real life
278	excourse65	data science fundamentals for data analysts
279	excourse66	executive data science capstone

	COURSE_ID	TITLE
280	excourse67	introduction to big data
281	excourse68	big data modeling and management systems
282	excourse69	machine learning with big data
283	excourse70	big data capstone project
284	excourse71	big data essentials hdfs mapreduce and spark rdd
285	excourse72	foundations for big data analysis with sql
286	excourse73	analyzing big data with sql
287	excourse74	fundamentals of big data
288	excourse75	hadoop platform and application framework
289	excourse76	working with big data
290	excourse77	natural language processing with attention models
291	excourse78	natural language processing with sequence models
292	excourse79	natural language processing with probabilistic models
293	excourse80	r programming
294	excourse81	data analysis with r programming
295	excourse82	getting started with data visualization in r
296	excourse83	introduction to probability and data with r
297	excourse84	using r for regression and machine learning in investment
298	excourse85	data visualization in r with ggplot2
299	excourse86	the r programming environment
300	excourse87	html css and javascript for web developers
301	excourse88	javascript basics
302	excourse89	javascript jquery and json
303	excourse90	programming foundations with javascript html and css
304	excourse91	front end web development with react
305	excourse92	introduction to web development
306	excourse93	interactivity with javascript and jquery

In [16]:

```
# Reset pandas settings
pd.reset_option('display.max_rows')
pd.reset_option('max_colwidth')
```

TODO: Browse the course list and choose your interested courses

In [17]:

```
enrolled_course_ids = [ 'DX0106EN','DX0108EN'] # add your interested coures id t
```

In [18]:
```
enrolled_courses = course_df[course_df['COURSE_ID'].isin(enrolled_course_ids)]
enrolled_courses
```

Out[18]:

	COURSE_ID	TITLE	DESCRIPTION
6	DX0106EN	data science bootcamp with r for university pr...	a multi day intensive in person data science ...
13	DX0108EN	data science bootcamp with python for universi...	data science bootcamp with python for universi...

Given the full course list, we can find those unselected courses:

In [19]:
```
all_courses = set(course_df['COURSE_ID'])
```

In [20]:
```
unselected_course_ids = all_courses.difference(enrolled_course_ids)
unselected_course_ids
```

```
Out[20]: {'AI0111EN',
          'BC0101EN',
          'BC0201EN',
          'BC0202EN',
          'BD0101EN',
          'BD0111EN',
          'BD0115EN',
          'BD0121EN',
          'BD0123EN',
          'BD0131EN',
          'BD0133EN',
          'BD0135EN',
          'BD0137EN',
          'BD0141EN',
          'BD0143EN',
          'BD0145EN',
          'BD0151EN',
          'BD0153EN',
          'BD0211EN',
          'BD0212EN',
          'BD0221EN',
          'BD0223EN',
          'BENTEST4',
          'CB0101EN',
          'CB0103EN',
          'CB0105ENv1',
          'CB0201EN',
          'CC0101EN',
          'CC0103EN',
          'CC0120EN',
          'CC0121EN',
          'CC0150EN',
          'CC0201EN',
          'CC0210EN',
          'CC0250EN',
          'CC0271EN',
          'CL0101EN',
          'CNSC02EN',
          'CO0101EN',
          'CO0193EN',
          'CO0201EN',
          'CO0301EN',
          'CO0302EN',
          'CO0401EN',
          'COM001EN',
          'CP0101EN',
          'DA0101EN',
          'DA0151EN',
          'DA0201EN',
          'DAI101EN',
          'DB0101EN',
          'DB0111EN',
          'DB0113EN',
          'DB0115EN',
          'DB0151EN',
          'DE0205EN',
          'DJ0101EN',
          'DP0101EN',
          'DS0101EN',
          'DS0103EN',
          'DS0105EN',
          'DS0107',
          'DS0110EN',
          'DS0132EN',
          'DS0201EN',
          'DS0301EN',
          'DS0321EN',
          'DV0101EN',
          'DV0151EN',
          'DW0101EN',
          'DX0107EN',
          'EE0101EN',
          'GPXX01AVEN',
          'GPXX01DCEN',
          'GPXX01RYEN',
          'GPXX03HFEN',
          'GPXX0435EN',
          'GPXX048OEN',
          'GPXX04HEEN',
          'GPXX04MXEN',
          'GPXX04P5EN',
          'GPXX04TNEN',
          'GPXX04V3EN',
          'GPXX04XJEN',
          'GPXX05LMEN',
          'GPXX05P1EN',
          'GPXX05RDEN',
          'GPXX06KEEN',
          'GPXX06RFEN',
          'GPXX06ZLEN',
          'GPXX0725EN',
          'GPXX0742EN',
          'GPXX07REN',
          'GPXX07UGEN',
          'GPXX07YGEN',
          'GPXX08WYEN',
          'GPXX097UEN',
          'GPXX0A1YEN',
          'GPXX0ADEN',
          'GPXX0BSAEN',
          'GPXX0BUBEN',
          'GPXX0D14EN',
          'GPXX0E3QEN',
          'GPXX0FFCEN',
          'GPXX0FTCEN',
          'GPXX0G31EN',
          'GPXX0G3KEN',
          'GPXX0G81EN',
          'GPXX0HAAEN',
          'GPXX0HC7EN',
          'GPXX0HZ2EN',
          'GPXX0I4FEN',
          'GPXX0IBEN',
          'GPXX0IHMEN',
          'GPXX0JGFEN',
          'GPXX0JLHEN',
          'GPXX0JZ4EN',
          'GPXX0KHHEN',
          'GPXX0KV4EN',
          'GPXX0KY1EN',
          'GPXX0LLEEN',
          'GPXX0M6UEN',
          'GPXX0M7ZEN',
          'GPXX0MIIEN',
          'GPXX0MP0EN',
          'GPXX0NHZEN',
          'GPXX0PG8EN',
          'GPXX0PICEN',
          'GPXX0Q8AEN',
          'GPXX0QJFEN',
          'GPXX0QQ3EN',
          'GPXX0QR3EN',
          'GPXX0QS6EN',
          'GPXX0QTEEN',
          'GPXX0QU9EN',
          'GPXX0RL8EN',
          'GPXX0RQLEN',
          'GPXX0SDXEN',
          'GPXX0T0FEN',
          'GPXX0T3CEN',
          'GPXX0TY1EN',
          'GPXX0UMSEN',
          'GPXX0UN5EN',
          'GPXX0W7KEN',
          'GPXX0WRDEN',
          'GPXX0WTIEN',
          'GPXX0XENEN',
          'GPXX0XFQEN',
          'GPXX0XV3EN',
          'GPXX0YBFEN',
          'GPXX0YMEEN',
          'GPXX0YXHEN',
          'GPXX0Z2PEN',
          'GPXX0ZG0EN',
          'GPXX0ZMZEN',
          'GPXX0ZYVEN',
          'HCC104EN',
          'HCC105EN',
          'IT0101EN',
          'LB0101ENv1',
          'LB0103ENv1',
          'LB0105ENv1',
          'LB0107ENv1',
          'LB0109ENv1',
          'LB0111EN',
          'ML0101EN',
          'ML0101ENv3',
          'ML0103EN',
          'ML0109EN',
          'ML0111EN',
          'ML0115EN',
          'ML0120EN',
          'ML0120ENv2',
          'ML0120ENv3',
          'ML0122EN',
          'ML0122ENv1',
          'ML0122ENv3',
          'ML0151EN',
          'ML0201EN',
          'OS0101EN',
          'PA0101EN',
          'PA0103EN',
          'PA0107EN',
          'PA0109EN',
          'PHPM002EN',
          'PY0101EN',
          'QC0101EN',
          'RAVSCTEST1',
          'RP0101EN',
          'RP0103',
          'RP0103EN',
          'RP0105EN',
          'RP0151EN',
          'SC0101EN',
          'SC0103EN',
          'SC0105EN',
          'SECM03EN',
          'SN0111EN',
          'ST0101EN',
          'ST0201EN',
          'ST0301EN',
          'SW0101EN',
          'SW0201EN',
          'TA0105',
          'TA0105EN',
          'TA0106EN',
          'TMP0101EN',
          'TMP0105EN',
          'TMP0106',
          'TMP107',
          'WA0101EN',
          'WA0103EN',
          'excourse01',
          'excourse02',
          'excourse03',
          'excourse04',
          'excourse05',
          'excourse06',
          'excourse07',
          'excourse08',
          'excourse09',
          'excourse10',
          'excourse11',
          'excourse12',
          'excourse13',
          'excourse14',
          'excourse15',
          'excourse16',
          'excourse17',
          'excourse18',
          'excourse19',
          'excourse20',
          'excourse21',
          'excourse22',
          'excourse23',
          'excourse24',
          'excourse25',
          'excourse26',
          'excourse27',
          'excourse28',
          'excourse29',
          'excourse30',
          'excourse31',
          'excourse32',
          'excourse33',
          'excourse34',
          'excourse35',
          'excourse36',
          'excourse37',
          'excourse38',
          'excourse39',
          'excourse40',
          'excourse41',
          'excourse42',
          'excourse43',
          'excourse44',
          'excourse45',
          'excourse46',
          'excourse47',
          'excourse48',
          'excourse49',
          'excourse50',
          'excourse51',
          'excourse52',
          'excourse53',
          'excourse54',
          'excourse55',
          'excourse56',
          'excourse57',
          'excourse58',
          'excourse59',
          'excourse60',
          'excourse61',
          'excourse62',
          'excourse63',
          'excourse64',
          'excourse65',
          'excourse66',
          'excourse67',
          'excourse68',
          'excourse69',
          'excourse70',
          'excourse71',
          'excourse72',
          'excourse73',
          'excourse74',
          'excourse75',
          'excourse76',
          'excourse77',
          'excourse78',
          'excourse79',
          'excourse80',
          'excourse81',
          'excourse82',
          'excourse83',
          'excourse84',
          'excourse85',
          'excourse86',
          'excourse87',
          'excourse88',
          'excourse89',
          'excourse90',
          'excourse91',
          'excourse92',
          'excourse93'}
```

Now, you can iterate each unselect course and check if it is similar enough to any of your selected courses. If the similarity is larger than a threshold such as 0.5 or 0.6, then add it to your course recommendation list:

TODO: Complete the following method to recommend courses which are similar to your enrolled courses

In [34]:

```
def generate_recommendations_for_one_user(
    enrolled_course_ids,
    unselected_course_ids,
    id_idx_dict,
    sim_matrix
):
    res = {}
    threshold = 0.6

    for enrolled_course in enrolled_course_ids:
        for unselect_course in unselected_course_ids:

            if enrolled_course in id_idx_dict and unselect_course in id_idx_dict
                enrolled_idx = id_idx_dict[enrolled_course]
                unselect_idx = id_idx_dict[unselect_course]

                sim = sim_matrix[enrolled_idx][unselect_idx]

                if sim > threshold:
                    if unselect_course not in res or sim >= res[unselect_course]
                        res[unselect_course] = sim

    # Sort by similarity (descending)
    res = dict(sorted(res.items(), key=lambda x: x[1], reverse=True))

    return res
```

▶ Click here for Hints

The completed `generate_recommendations_user(...)` may ouput a dictionary like this:

{'ML0151EN': 0.6626221399549089, 'excourse47': 0.6347547807096177, 'excourse46': 0.6120541193300345}

TASK: Generate course recommendations based on course similarities for all test uesrs

In the previous task, you made some recommendations for yourself. Next, let's try to make recommendations for all the test users in the test dataset.

In [35]:
```
test_users_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.c
test_users_df = pd.read_csv(test_users_url)
```

Let's look at how many test users we have in the dataset.

In [36]:
```
test_users = test_users_df.groupby(['user']).max().reset_index(drop=False)
test_user_ids = test_users['user'].to_list()
print(f"Total numbers of test users {len(test_user_ids)}")
```

```
Total numbers of test users 33901
```

TODO: Complete the `generate_recommendations_for_all()` method to generate recommendations for all users. You may implement the task with different solutions

In [39]:
```
def generate_recommendations_for_all():
    users = []
    courses = []
    sim_scores = []

    # Load data
    sim_df = pd.read_csv(sim_url)
    course_df = pd.read_csv(course_url)

    # Build course-id → index mapping
    id_idx_dict = dict(
        zip(course_df['COURSE_ID'], course_df.index)
    )

    # Prepare test users
    test_users = test_users_df.groupby(['user']).max().reset_index(drop=False)
    test_user_ids = test_users['user'].to_list()

    all_courses = set(course_df['COURSE_ID'])

    for user_id in test_user_ids:
        users.append(user_id)

        enrolled_course_ids = test_users[
            test_users['user'] == user_id
        ]['item'].to_list()

        unselected_course_ids = list(
            all_courses.difference(enrolled_course_ids)
        )

        rec_dict = generate_recommendations_for_one_user(
            enrolled_course_ids,
            unselected_course_ids,
            id_idx_dict,
            sim_df.values
        )

        courses.append(list(rec_dict.keys()))
        sim_scores.append(list(rec_dict.values()))

    return users, courses, sim_scores
```

▶ Click here for Hints

After you completed the `generate_recommendations_for_all()` function, you can call it to save the results into a dataframe:

In [40]:
```
res_dict = {}
users, courses, sim_scores = generate_recommendations_for_all()
res_dict['USER'] = users
res_dict['COURSE_ID'] = courses
res_dict['SCORE'] = sim_scores
res_df = pd.DataFrame(res_dict, columns=['USER', 'COURSE_ID', 'SCORE'])
res_df
```

Out[40]:

	USER	COURSE_ID	SCORE
0	2	[WA0103EN]	[0.6311528416041716]
1	4	[WA0101EN]	[0.6311528416041716]
2	5	[WA0101EN]	[0.6311528416041716]
3	7	[]	[]
4	8	[]	[]
...	...	...	...
33896	2102054	[]	[]
33897	2102356	[]	[]
33898	2102680	[excourse22, excourse62]	[0.6475015976638527, 0.6475015976638527]
33899	2102983	[DAI101EN]	[0.6689936080056725]
33900	2103039	[DAI101EN]	[0.6689936080056725]

33901 rows × 3 columns

Similar to the previous user profile and course genre lab, with the recommendations generated for each user, you need to write some extra analytic code to answer the following questions:

- On average, how many new/unseen courses have been recommended to each user?
- What are the most frequently recommended courses? Return the top-10 commonly recommended courses across all users?

For example, suppose we have only 3 test users, each user receives the following recommendations:

- User1: ['course1', 'course2']
- User2: ['course3', 'course4']
- User3: ['course3', 'course4', 'course5']

Then, the average recommended courses per user is $(2 + 2 + 3) / 3 = 2.33$. The top-2 recommended courses are: `course3` : 2 times, and `course4` : 2 times.

Note that the answers may depend on your similarity threshold (default is 0.6). A lower similarity threshold yields more recommended courses but with smaller irrelevance.

Ideally, we should limit the maximum course recommendations for each user to be less than 20 courses per user.

Authors

Yan Luo

Other Contributors

```
toggle##


toggle|Date

toggle|-|-|-|-|

toggle|2021-10-25|1.0|Yan|Created
```

Copyright © 2021 IBM Corporation. All rights reserved.