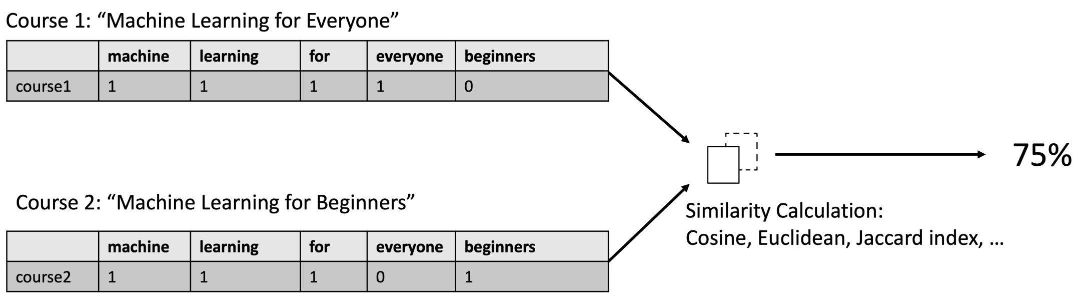
Calculate Course Similarity using BoW Features

Estimated time needed: **45** minutes

Similarity measurement between items is the foundation of many recommendation algorithms, especially for content-based recommendation algorithms. For example, if a new course is similar to user's enrolled courses, we could recommend that new similar course to the user. Or If user A is similar to user B, then we can recommend some of user B's courses to user A (the unseen courses) because user A and user B may have similar interests.

In a previous course, you learned many similarity measurements such as `consine`, `jaccard index`, or `euclidean distance`, and these methods need to work on either two vectors or two sets (sometimes even matrices or tensors).

In previous labs, we extracted the BoW features from course textual content. Given the course BoW feature vectors, we can easily apply similarity measurement to calculate the course similarity as shown in the below figure.


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

Similarity Calculation:
Cosine, Euclidean, Jaccard index, ...
75%

Objectives

After completing this lab you will be able to:

- Calculate the similarity between any two courses using BoW feature vectors

---

Prepare and setup lab environment

First let's install and import required libraries:

In [1]:
```
!pip install nltk
!pip install gensim
!pip install scipy==1.10
!pip install pandas
!pip install matplotlib
!pip install seaborn
```

```
Collecting nltk
  Downloading nltk-3.9.2-py3-none-any.whl.metadata (3.2 kB)
Collecting click (from nltk)
  Downloading click-8.3.1-py3-none-any.whl.metadata (2.6 kB)
Collecting joblib (from nltk)
  Downloading joblib-1.5.3-py3-none-any.whl.metadata (5.5 kB)
Collecting regex>=2021.8.3 (from nltk)
  Downloading regex-2025.11.3-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86
_64.manylinux_2_28_x86_64.whl.metadata (40 kB)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.12/site-packages (f
rom nltk) (4.67.1)
Downloading nltk-3.9.2-py3-none-any.whl (1.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 95.1 MB/s eta 0:00:00
Downloading regex-2025.11.3-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_6
4.manylinux_2_28_x86_64.whl (803 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 803.5/803.5 kB 60.0 MB/s eta 0:00:00
Downloading click-8.3.1-py3-none-any.whl (108 kB)
Downloading joblib-1.5.3-py3-none-any.whl (309 kB)
Installing collected packages: regex, joblib, click, nltk
Successfully installed click-8.3.1 joblib-1.5.3 nltk-3.9.2 regex-2025.11.3
Collecting gensim
  Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_6
4.whl.metadata (8.4 kB)
Collecting numpy>=1.18.5 (from gensim)
  Downloading numpy-2.4.1-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_6
4.whl.metadata (6.6 kB)
Collecting scipy>=1.7.0 (from gensim)
  Downloading scipy-1.17.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_6
4.whl.metadata (62 kB)
Collecting smart_open>=1.8.1 (from gensim)
  Downloading smart_open-7.5.0-py3-none-any.whl.metadata (24 kB)
Collecting wrapt (from smart_open>=1.8.1->gensim)
  Downloading wrapt-2.0.1-cp312-cp312-manylinux1_x86_64.manylinux_2_28_x86_64.man
ylinux_2_5_x86_64.whl.metadata (9.0 kB)
Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.
whl (27.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 27.9/27.9 MB 149.2 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 179.5 MB/s eta 0:00:00
Downloading scipy-1.17.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.
whl (35.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 35.0/35.0 MB 70.0 MB/s eta 0:00:00:0
0:01
Downloading smart_open-7.5.0-py3-none-any.whl (63 kB)
Downloading wrapt-2.0.1-cp312-cp312-manylinux1_x86_64.manylinux_2_28_x86_64.manyl
inux_2_5_x86_64.whl (121 kB)
Installing collected packages: wrapt, numpy, smart_open, scipy, gensim
Successfully installed gensim-4.4.0 numpy-2.4.1 scipy-1.17.0 smart_open-7.5.0 wra
pt-2.0.1
ERROR: Ignored the following yanked versions: 1.11.0, 1.14.0rc1
ERROR: Could not find a version that satisfies the requirement scipy==1.10 (from
versions: 0.8.0, 0.9.0, 0.10.0, 0.10.1, 0.11.0, 0.12.0, 0.12.1, 0.13.0, 0.13.1,
0.13.2, 0.13.3, 0.14.0, 0.14.1, 0.15.0, 0.15.1, 0.16.0, 0.16.1, 0.17.0, 0.17.1,
0.18.0, 0.18.1, 0.19.0, 0.19.1, 1.0.0, 1.0.1, 1.1.0, 1.2.0, 1.2.1, 1.2.2, 1.2.3,
1.3.0, 1.3.1, 1.3.2, 1.3.3, 1.4.0, 1.4.1, 1.5.0, 1.5.1, 1.5.2, 1.5.3, 1.5.4, 1.6.
0, 1.6.1, 1.9.2, 1.9.3, 1.11.0rc1, 1.11.0rc2, 1.11.1, 1.11.2, 1.11.3, 1.11.4, 1.1
2.0rc1, 1.12.0rc2, 1.12.0, 1.13.0rc1, 1.13.0, 1.13.1, 1.14.0rc2, 1.14.0, 1.14.1,
1.15.0rc1, 1.15.0rc2, 1.15.0, 1.15.1, 1.15.2, 1.15.3, 1.16.0rc1, 1.16.0rc2, 1.16.
0, 1.16.1, 1.16.2, 1.16.3, 1.17.0rc1, 1.17.0rc2, 1.17.0)
```

```
ERROR: No matching distribution found for scipy==1.10
Collecting pandas
  Downloading pandas-2.3.3-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_6
4.whl.metadata (91 kB)
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: pytz>=2020.1 in /opt/conda/lib/python3.12/site-pac
kages (from pandas) (2024.2)
Collecting tzdata>=2022.7 (from pandas)
  Downloading tzdata-2025.3-py2.py3-none-any.whl.metadata (1.4 kB)
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)
Downloading pandas-2.3.3-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.
whl (12.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.4/12.4 MB 132.7 MB/s eta 0:00:00
Downloading tzdata-2025.3-py2.py3-none-any.whl (348 kB)
Installing collected packages: tzdata, pandas
Successfully installed pandas-2.3.3 tzdata-2025.3
Collecting matplotlib
  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)
  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)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  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)
  Downloading kiwisolver-1.4.9-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x8
6_64.whl.metadata (6.3 kB)
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)
Collecting pillow>=8 (from matplotlib)
  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)
  Downloading pyparsing-3.3.1-py3-none-any.whl.metadata (5.6 kB)
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)
Downloading matplotlib-3.10.8-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86
_64.whl (8.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.7/8.7 MB 164.8 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 164.2 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 106.0 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 171.6 MB/s eta 0:00:00
Downloading pyparsing-3.3.1-py3-none-any.whl (121 kB)
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler,
contourpy, matplotlib
Successfully installed contourpy-1.3.3 cycler-0.12.1 fonttools-4.61.1 kiwisolver-
1.4.9 matplotlib-3.10.8 pillow-12.1.0 pyparsing-3.3.1
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /opt/conda/lib/python3.12/
site-packages (from seaborn) (2.4.1)
Requirement already satisfied: pandas>=1.2 in /opt/conda/lib/python3.12/site-pack
ages (from seaborn) (2.3.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /opt/conda/lib/python3.
12/site-packages (from seaborn) (3.10.8)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.12/site
-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.12/site-pac
kages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.12/sit
e-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/conda/lib/python3.12/sit
e-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.12/site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.12/site-packag
es (from matplotlib!=3.6.1,>=3.4->seaborn) (12.1.0)
Requirement already satisfied: pyparsing>=3 in /opt/conda/lib/python3.12/site-pac
kages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.3.1)
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: pytz>=2020.1 in /opt/conda/lib/python3.12/site-pac
kages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.12/site-p
ackages (from pandas>=1.2->seaborn) (2025.3)
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)
Installing collected packages: seaborn
Successfully installed seaborn-0.13.2
```

In [17]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import gensim
import pandas as pd
import nltk as nltk
import scipy
from scipy.spatial.distance import cosine,euclidean
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk import ngrams
from gensim import corpora

%matplotlib inline
```

In [3]:
```python
# also set a random state
rs = 123
```

Calculate the consine similarity between two example courses

Suppose we have two simple example courses:

In [4]:
```
course1 = "machine learning for everyone"
```

In [5]:
```
course2 = "machine learning for beginners"
```

Next we can quickly tokenize them using the split() method (or using `word_tokenize()` method provided in `nltk` as we did in the previous lab).

In [6]:
```
tokens = set(course1.split() + course2.split())
```

In [7]:
```
tokens = list(tokens)
tokens
```

Out[7]:
```
['machine', 'for', 'learning', 'beginners', 'everyone']
```

then generate BoW features (token counts) for these two courses (or using `tokens_dict.doc2bow()` method provided in `nltk`, similar to what we did in the previous lab).

In [8]:
```
def generate_sparse_bow(course):
    """
    Generate a sparse bag-of-words (BoW) representation for a given course.

    Parameters:
    course (str): The input course text to generate the BoW representation for.

    Returns:
    list: A sparse BoW representation where each element corresponds to the pres
    of a word in the input course text.
    """

    # Initialize an empty list to store the BoW vector
    bow_vector = []

    # Tokenize the course text by splitting it into words
    words = course.split()

    # Iterate through all unique words (tokens) in the course
    for token in set(words):
        # Check if the token is present in the course text
        if token in words:
            # If the token is present, append 1 to the BoW vector
            bow_vector.append(1)
        else:
            # If the token is not present, append 0 to the BoW vector
            bow_vector.append(0)

    # Return the sparse BoW vector
    return bow_vector
```

In [15]:
```
bow1 = generate_sparse_bow(course1)
bow1
```

Out[15]:
```
[1, 1, 1, 1]
```

In [10]:
```
bow2 = generate_sparse_bow(course2)
bow2
```

Out[10]:
```
[1, 1, 1, 1]
```

From the above cell outputs, we can see the two vectors are very similar. Only two dimensions are different.

Now we can quickly apply the cosine similarity measurement on the two vectors:

In [11]:
```
cos_sim = 1 - cosine(bow1, bow2)
```

In [12]:
```
print(f"The cosine similarity between course `{course1}` and course `{course2}`
```

```
The cosine similarity between course `machine learning for everyone` and course `
machine learning for beginners` is 100.0%
```

Practice: Try other similarity measurements such as Euclidean Distance or Jaccard index.

In [18]:
```
# WRITE YOUR CODE HERE
ed = euclidean(bow1,bow2)
print(ed)
```

```
0.0
```

For Example: Euclidean distance between 2 points $p$ and $q$ can be summarized by this equation: $d(p,q)={\sqrt {(p_{1}-q_{1})^{2}+(p_{2}-q_{2})^{2}+(p_{3}-q_{3})^{2}}}$. You can use `euclidean(p,q)` function from `scipy` package to calculate it.

TASK: Find similar courses to the course `Machine Learning with Python`

Now you have learned how to calculate cosine similarity between two sample BoW feature vectors. Let's work on some real course BoW feature vectors.

In [19]:
```
# Load the BoW features as Pandas dataframe
bows_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/I
bows_df = pd.read_csv(bows_url)
bows_df = bows_df[['doc_id', 'token', 'bow']]
```

In [20]:
```
bows_df.head(10)
```

Out[20]:

	doc_id	token	bow
0	ML0201EN	ai	2
1	ML0201EN	apps	2
2	ML0201EN	build	2
3	ML0201EN	cloud	1
4	ML0201EN	coming	1
5	ML0201EN	create	1
6	ML0201EN	data	1
7	ML0201EN	developer	1
8	ML0201EN	found	1
9	ML0201EN	fun	1

The `bows_df` dataframe contains the BoW features vectors for each course, in a vertical and dense format. It has three columns `doc_id` represents the course id, `token` represents the token value, and `bow` represents the BoW value (token count).

Then, let's load another course content dataset which contains the course title and description:

In [21]:
```
# Load the course dataframe
course_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud
course_df = pd.read_csv(course_url)
```

In [22]:
```
course_df.head(10)
```

Out[22]:

	COURSE_ID	TITLE	DESCRIPTION
0	ML0201EN	robots are coming build iot apps with watson ...	have fun with iot and learn along the way if ...
1	ML0122EN	accelerating deep learning with gpu	training complex deep learning models with lar...
2	GPXX0ZG0EN	consuming restful services using the reactive ...	learn how to use a reactive jax rs client to a...
3	RP0105EN	analyzing big data in r using apache spark	apache spark is a popular cluster computing fr...
4	GPXX0Z2PEN	containerizing packaging and running a sprin...	learn how to containerize package and run a ...
5	CNSC02EN	cloud native security conference data security	introduction to data security on cloud
6	DX0106EN	data science bootcamp with r for university pr...	a multi day intensive in person data science ...
7	GPXX0FTCEN	learn how to use docker containers for iterati...	learn how to use docker containers for iterati...
8	RAVSCTEST1	scorm test 1	scron test course
9	GPXX06RFEN	create your first mongodb database	in this guided project you will get started w...

Given course ID `ML0101ENv3`, let's find out its title and description:

In [23]:
```
course_df[course_df['COURSE_ID'] == 'ML0101ENv3']
```

Out[23]:

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

We can see it is a machine learning with Python course so we can expect any machine learning or Python related courses would be similar.

Then, let's print its associated BoW features:

In [24]:
```
ml_course = bows_df[bows_df['doc_id'] == 'ML0101ENv3']
ml_course
```

Out[24]:

	doc_id	token	bow
2747	ML0101ENv3	course	1
2748	ML0101ENv3	learning	4
2749	ML0101ENv3	machine	3
2750	ML0101ENv3	need	1
2751	ML0101ENv3	get	1
2752	ML0101ENv3	started	1
2753	ML0101ENv3	python	2
2754	ML0101ENv3	tool	1
2755	ML0101ENv3	tools	1
2756	ML0101ENv3	predict	1
2757	ML0101ENv3	free	1
2758	ML0101ENv3	hidden	1
2759	ML0101ENv3	insights	1
2760	ML0101ENv3	beneficial	1
2761	ML0101ENv3	future	1
2762	ML0101ENv3	trends	1
2763	ML0101ENv3	give	1
2764	ML0101ENv3	supervised	1
2765	ML0101ENv3	unsupervised	1

We can see the BoW feature vector is in vertical format but normally feature vectors are in horizontal format. One way to transpose the feature vector from vertical to horizontal is to use the Pandas `pivot()` method:

In [25]:
```
ml_courseT = ml_course.pivot(index=['doc_id'], columns='token').reset_index(leve
ml_courseT
```

Out[25]:

	doc_id								
token		**beneficial**	**course**	**free**	**future**	**get**	**give**	**hidden**	**insights**	**learni**
0	ML0101ENv3	1	1	1	1	1	1	1	1	

To compare the BoWs of any two courses, which normally have a different set of tokens, we need to create a union token set and then transpose them. We have provided a method called `pivot_two_bows` as follows:

In [56]:
```
def pivot_two_bows(basedoc, comparedoc):
    """
    Pivot two bag-of-words (BoW) representations for comparison.

    Parameters:
    basedoc (DataFrame): DataFrame containing the bag-of-words representation fo
    comparedoc (DataFrame): DataFrame containing the bag-of-words representation

    Returns:
    DataFrame: A DataFrame with pivoted BoW representations for the base and com
    facilitating direct comparison of word occurrences between the two documents
    """

    # Create copies of the input DataFrames to avoid modifying the originals
    base = basedoc.copy()
    base['type'] = 'base'  # Add a 'type' column indicating base document
    compare = comparedoc.copy()
    compare['type'] = 'compare'  # Add a 'type' column indicating compared docum

    # Concatenate the two DataFrames vertically
    join = pd.concat([base, compare])

    # Pivot the concatenated DataFrame based on 'doc_id' and 'type', with words
    joinT = join.pivot(index=['doc_id', 'type'], columns='token').fillna(0).rese

    # Assign meaningful column names to the pivoted DataFrame
    joinT.columns = ['doc_id', 'type'] + [t[1] for t in joinT.columns][2:]

    # Return the pivoted DataFrame for comparison
    return joinT
```

In [27]:
```
course1 = bows_df[bows_df['doc_id'] == 'ML0151EN']
course2 = bows_df[bows_df['doc_id'] == 'ML0101ENv3']
```

In [28]:
```
bow_vectors = pivot_two_bows(course1, course2)
bow_vectors
```

Out[28]:

	doc_id	type	approachable	basics	beneficial	comparison	course	dives
0	ML0101ENv3	compare	0.0	0.0	1.0	0.0	1.0	0.0
1	ML0151EN	base	1.0	1.0	0.0	1.0	1.0	1.0

2 rows × 36 columns

Similarly, we can use the cosine method to calculate their similarity:

In [29]:
```
similarity = 1 - cosine(bow_vectors.iloc[0, 2:], bow_vectors.iloc[1, 2:])
similarity
```

Out[29]:
```
np.float64(0.662622139954909)
```

Now it's your turn to perform a task of finding all courses similar to the course `Machine Learning with Python`:

In [30]:
```
course_df[course_df['COURSE_ID'] == 'ML0101ENv3']
```

Out[30]:

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

You can set a similarity threshold such as 0.5 to determine if two courses are similar enough.

TODO: Find courses which are similar to course `Machine Learning with Python (ML0101ENv3)` *, you also need to show the title and descriptions of those courses.*

In [60]:
```python
# WRITE YOUR CODE HERE
similar_courses = []
course = bows_df[bows_df['doc_id'] == 'ML0101ENv3']
other_courses = bows_df[bows_df['doc_id'] != 'ML0101ENv3']['doc_id'].unique()
for c in other_courses:
    current_course = bows_df[bows_df['doc_id'] == c]
    pivot_df = pivot_two_bows(course, current_course)
    pivot_df = pivot_df.loc[:, ~pivot_df.columns.duplicated()]
    bow_base = pivot_df.loc[pivot_df['type'] == 'base'].drop(columns=['doc_id',

    bow_compare = pivot_df.loc[pivot_df['type'] == 'compare'].drop(columns=['doc

    sim = 1 - cosine(bow_base, bow_compare)

    if sim > 0.5:
        similar_courses.append((c, sim))

print(similar_courses)
## For each course other than ML0101ENv3, use pivot_course_rows to convert it wi
## Then use the cosine method to calculate the similarity
## Report all courses with similarities larger than a specific threshold (such a
```

```
[('ML0109EN', np.float64(0.5217491947499509)), ('ML0151EN', np.float64(0.66262213
9954909)), ('excourse46', np.float64(0.6120541193300345)), ('excourse47', np.floa
t64(0.6347547807096177)), ('excourse60', np.float64(0.5490400192158565))]
```

▶ Click here for Hints

Summary

Congratulations, you have finished the course similarity lab. In this lab, you used cosine and course BoW features to calculate the similarities among courses. Such similarity measurement is the core of many content-based recommender systems, which you will learn and practice in the later labs.

Authors

Yan Luo

Other Contributors

```
toggle##

toggle|Date

toggle|-|-|-|-|

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