NLP recommendation Engine for course material

1. Introduction

Natural Language Processing (NLP) is an area that explores how computers can be used to comprehend and manipulate natural language speech or text in a more beneficial manner. To develop the suitable tools and techniques to make computer systems comprehend and manipulate natural languages and perform desired tasks, it is important that information on how human beings understand and use language is gathered. NLP has been applied to many fields such as natural language text processing and summarizing, machine translation, multilingual and cross-language information retrieval (CLIR), user interfaces, speech recognition, artificial intelligence, and building of recommendation engines.

Recommendation Systems (RSs) use knowledge discovery and data mining techniques to predict items of interest to users and subsequently suggest these items to them as recommendations (Bobadilla et al 2013, Lu et al 2015). The exponential increase in the number of on-line information and on-line users resulted in a quick evolution in the techniques and applications of RS.

Methods for building RSs can be broadly grouped into three main categories: Content-Based (CB), Collaborative Filtering (CF), and hybrid techniques. CB techniques measure or predict the similarity between the profiles of items (descriptions or attributes) and users' profiles (attributes/descriptions of previously preferred items; Aggarwal 2016, Pazzani & Billsus 2007). The similarities are usually measured using appropriate functions such as cosine similarity or predicted using statistical or machine learning models such as regression. Therefore, CB systems are more suitable in areas where item descriptions can be easily obtained such as books, news articles, jobs and TV shows. They are also suitable for items with short lifetime and/or when the recommendation item pool changes frequently due to many new items entering the pool over short periods. However, these restrictions limit the number of ratings those items can get over their lifetime.

CF is a very prominent and successful techniques of modern recommendation systems. CF leverages the preferences of other similar users to make recommendations to the target user (Bobadilla et al 2013, Lu et al 2015). CF predicts good recommendations by analysing the correlations between all user behaviours rather than analysing correlations between items content. Therefore, CF is more suitable in domains where collecting meta-features or descriptions of items is illogical. CF is to provides more variety and providence to user's experience by taking advantage of the wisdom of the crowd. It recommends items that the target user may not have thought about but has been liked by other similar users. A major challenge of CF based recommendation systems is the cold-start problem which happens when there is no information linking new users or items. The RS is incapable of determining how those users or items are related and is therefore unable to offer useful recommendations.

Hybrid RSs achieve better performance by combining two or more techniques. Hybrid RS adopts the advantages of both CB and CF, and to lessen their inadequacies through hybridization.

The building of reading recommendation engines is an important tool used in many advanced university libraries to recommend the most suitable contents to students. RS can recommend reading materials down to a book or page number, which is a useful product for academics, publishers and other agencies. They support teaching and learning activities through enhanced information retrieval. Earlier recommendation systems focused on requirement fulfilment (Parameswaran et al 2011) and career-based relevancy recommendation (Farzan & Brusilovsky 2011). However, recently, recommender systems in higher education focuses on prediction of which courses a student will likely take (Pardos, Fan & Jiang 2019; Polyzou, Athanasios & Karypis 2019) or their grade if enrolled [Jiang & Pardos 2019; Jiang, Pardos & Wei 2019 ; Ren et al 2019]. A system at Stanford allows students to see course evaluations, grade distributions, and the most popular subjects taken before a subject of interest (Chaturapruek et al 2016). The recommender system at UC Berkeley recommends next-semester subject considerations based on their personal subject enrolment history (Pardos, Fan & Jiang 2019).

Therefore, this report is aimed at using the tern frequency and inverse document frequency (TFIDFI), K-nearest neighbour (KNN) and count vectorizer recommender engines to recommend existing course material to similar subjects. It also determines the quality of the natural language processors (NLP) using the test and training set and compare the two NLP recommenders.

2.0 Exploratory Data Analysis

Data exploration:

The data set consists of 18 features with 68,531 observations and six selected variables were used for the analysis. The six variables were selected because it was very challenging to fetch some of the supplementary data due to poor internet connections and the high volume of the dataset.

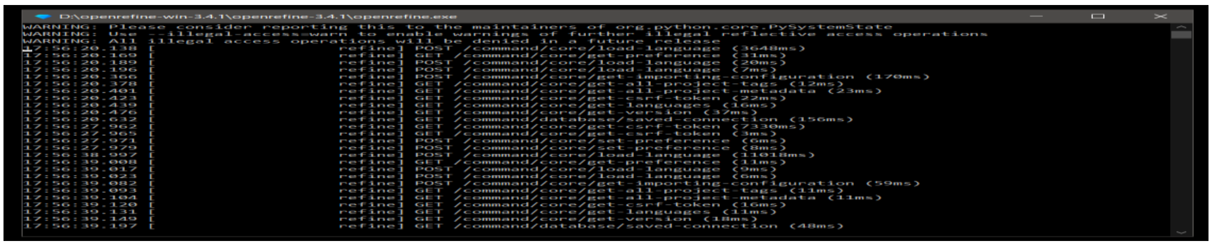
Variables names and description:

1. ID: University ID
2. Coursename: name of the course
3. ITEM_COUNT: number of items in the reading list
4. RESOURCE_TYPE: book journals
5. TITLE: major title (book, journal)
6. SUBTITLE: minor title (article)

The dataset was pre-processed using OpenRefine

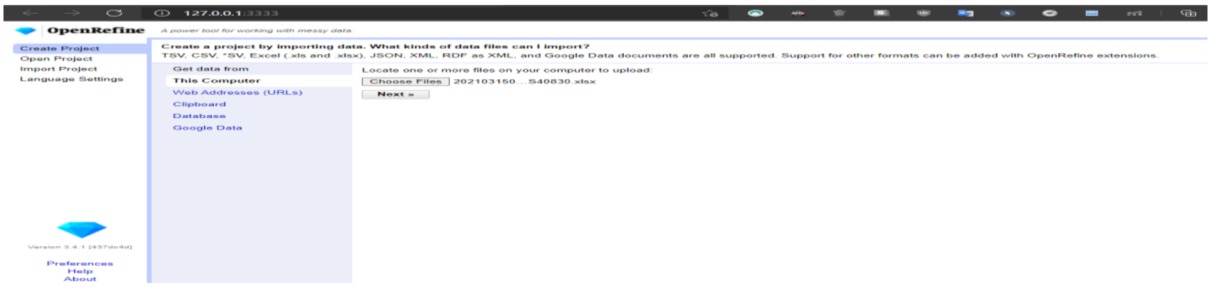
Step 1. Open refine was launch open refine and this step automatically opened a default browser, open refine was run locally using a computer web browser as its primary interface as shown in figure 1.0 and 1.1

Step 2: The dataset was loaded from the local computer into open refine and create project tab was selected (figure 1.1)



```
D:\openrefine-win-3.4.1\openrefine-3.4.1\openrefine.exe
WARNING: Please consider reporting this to the maintainers of org.python.core.PySystemState
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
17:56:20.138 [                          refine] POST /command/core/load-language (3648ms)
17:56:20.169 [                          refine] GET /command/core/get-preference (31ms)
17:56:20.189 [                          refine] POST /command/core/load-language (20ms)
17:56:20.196 [                          refine] POST /command/core/load-language (7ms)
17:56:20.366 [                          refine] POST /command/core/get-importing-configuration (170ms)
17:56:20.378 [                          refine] GET /command/core/get-all-project-tags (12ms)
17:56:20.401 [                          refine] GET /command/core/get-all-project-metadata (23ms)
17:56:20.423 [                          refine] GET /command/core/get-csrf-token (22ms)
17:56:20.439 [                          refine] GET /command/core/get-languages (16ms)
17:56:20.476 [                          refine] GET /command/core/get-version (37ms)
17:56:20.632 [                          refine] GET /command/database/saved-connection (156ms)
17:56:27.962 [                          refine] GET /command/core/get-csrf-token (7330ms)
17:56:27.965 [                          refine] GET /command/core/get-csrf-token (3ms)
17:56:27.971 [                          refine] POST /command/core/set-preference (6ms)
17:56:27.979 [                          refine] POST /command/core/set-preference (8ms)
17:56:38.997 [                          refine] POST /command/core/load-language (11018ms)
17:56:39.008 [                          refine] GET /command/core/get-preference (11ms)
17:56:39.017 [                          refine] POST /command/core/load-language (9ms)
17:56:39.023 [                          refine] POST /command/core/load-language (6ms)
17:56:39.082 [                          refine] POST /command/core/get-importing-configuration (59ms)
17:56:39.093 [                          refine] GET /command/core/get-all-project-tags (11ms)
17:56:39.104 [                          refine] GET /command/core/get-all-project-metadata (11ms)
17:56:39.120 [                          refine] GET /command/core/get-csrf-token (16ms)
17:56:39.131 [                          refine] GET /command/core/get-languages (11ms)
17:56:39.149 [                          refine] GET /command/core/get-version (18ms)
17:56:39.197 [                          refine] GET /command/database/saved-connection (48ms)
```

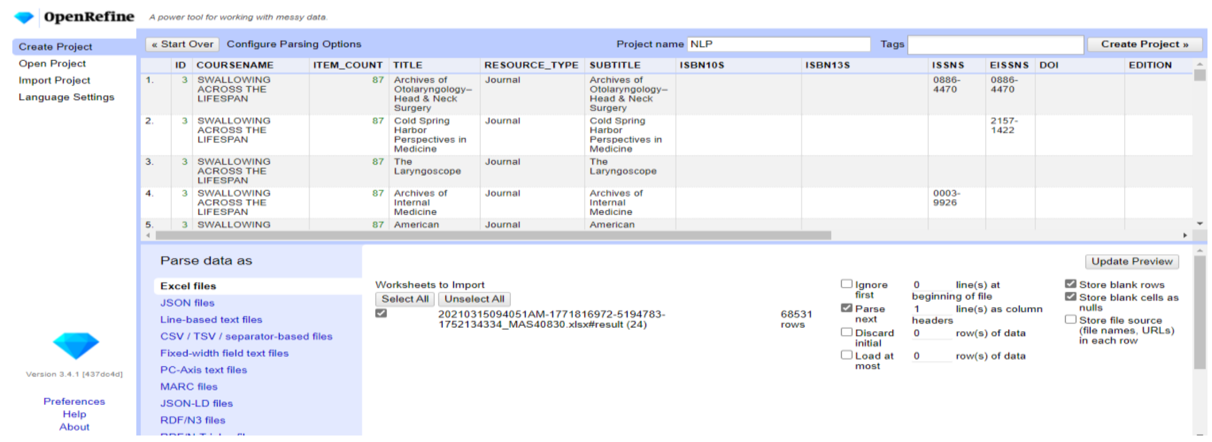
Figure 1.0 Lunch screen on open refine.



Create a project by importing data. What kinds of data files can I import?
TSV, CSV, *SV, Excel (.xls and .xlsx), JSON, XML, RDF as XML, and Google Data documents are all supported. Support for other formats can be added with OpenRefine extensions.

Locate one or more files on your computer to upload:
202103150....540830.xlsx

Figure 1.1 loading datasets on open refine



Project name NLP Tags

	ID	COURSENAME	ITEM_COUNT	TITLE	RESOURCE_TYPE	SUBTITLE	ISBN10S	ISBN13S	ISSNS	EISSNS	DOI	EDITION
1.	3	SWALLOWING ACROSS THE LIFESPAN	87	Archives of Otolaryngology–Head & Neck Surgery	Journal	Archives of Otolaryngology–Head & Neck Surgery			0886-4470	0886-4470		
2.	3	SWALLOWING ACROSS THE LIFESPAN	87	Cold Spring Harbor Perspectives in Medicine	Journal	Cold Spring Harbor Perspectives in Medicine				2157-1422		
3.	3	SWALLOWING ACROSS THE LIFESPAN	87	The Laryngoscope	Journal	The Laryngoscope						
4.	3	SWALLOWING ACROSS THE LIFESPAN	87	Archives of Internal Medicine	Journal	Archives of Internal Medicine			0003-9926			
5.	3	SWALLOWING	87	American	Journal	American						

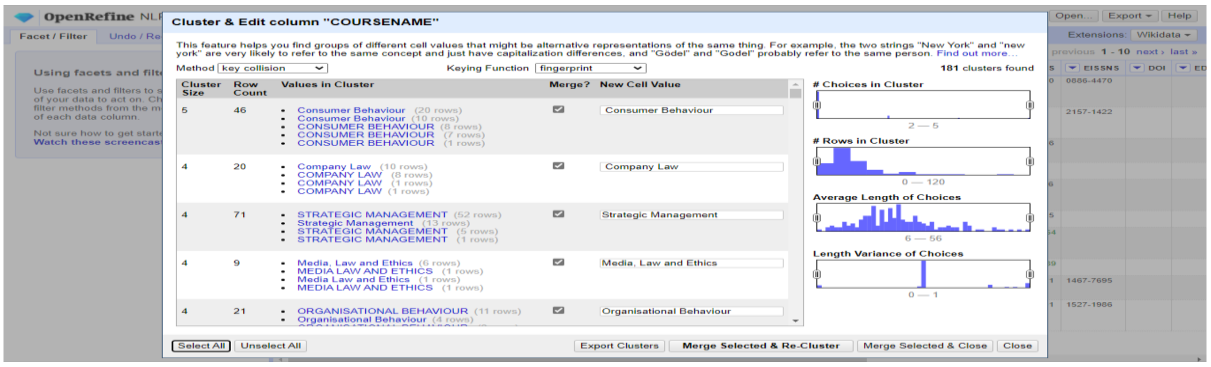
Worksheets to Import
☑ 20210315094051AM-1771816972-5194783-1752134334_MAS40830.xlsx#result (24) 68531 rows

☐ Ignore first 0 line(s) at beginning of file
☑ Parse next 1 line(s) as column headers
☐ Discard initial 0 row(s) of data
☐ Load at most 0 row(s) of data

☑ Store blank rows
☑ Store blank cells as nulls
☐ Store file source (file names, URLs) in each row

Figure 1.2

Step 7. Performing some basic cleaning operation of COURSE NAME by selecting facet, select text facet which shows 3657 choices select count to order to a list the most frequently occurring course name first, the cluster was selected to show some course name consistency problem within the dataset. select method key collision. Figure 1.4 shows the cluster > edit column "COURSE NAME


Cluster & Edit column "COURSENAME"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. Find out more...

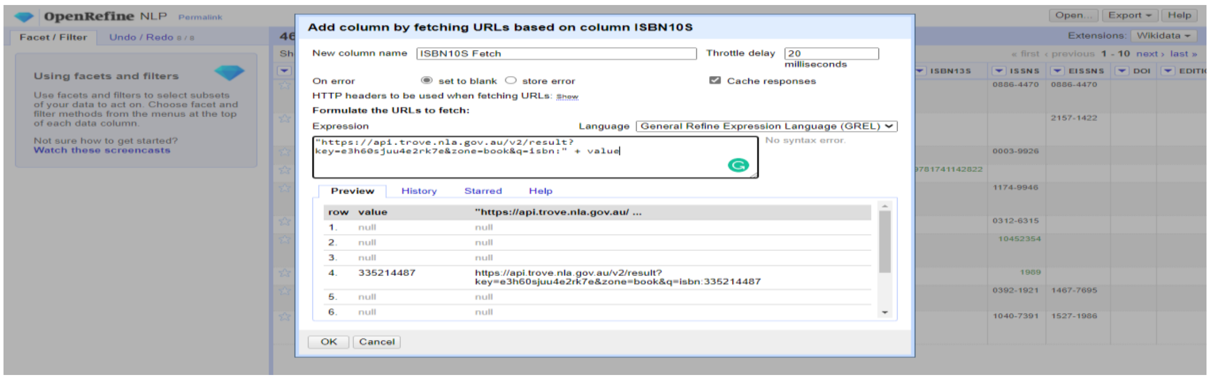
Method: key collision   Keying Function: fingerprint   181 clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
5	46	Consumer Behaviour (20 rows); Consumer Behaviour (10 rows); CONSUMER BEHAVIOUR (8 rows); CONSUMER BEHAVIOUR (7 rows); CONSUMER BEHAVIOUR (1 rows)	✓	Consumer Behaviour
4	20	Company Law (10 rows); COMPANY LAW (8 rows); COMPANY LAW (1 rows); COMPANY LAW (1 rows)	✓	Company Law
4	71	STRATEGIC MANAGEMENT (52 rows); Strategic Management (13 rows); STRATEGIC MANAGEMENT (5 rows); STRATEGIC MANAGEMENT (1 rows)	✓	Strategic Management
4	9	Media, Law and Ethics (6 rows); MEDIA LAW AND ETHICS (1 rows); Media Law and Ethics (1 rows); MEDIA LAW AND ETHICS (1 rows)	✓	Media, Law and Ethics
4	21	ORGANISATIONAL BEHAVIOUR (11 rows); Organisational Behaviour (4 rows)	✓	Organisational Behaviour

Figure 1.3

step 8. Doing the title case transformation of the course name field, title, subtitle, and the rest of the selected data for the analysis.

step 9. performing adding columns by fetching URLs based on column title for ISBN and book description using trove api.
But unfortunately, most attempts came with an error message after several attempts and hours spent shown in figure 1.4 and figure 1.5.


Add column by fetching URLs based on column ISBN10S

New column name: ISBN10S Fetch   Throttle delay: 20 milliseconds

On error: (●) set to blank ( ) store error   [✓] Cache responses

HTTP headers to be used when fetching URLs: Show

Formulate the URLs to fetch:

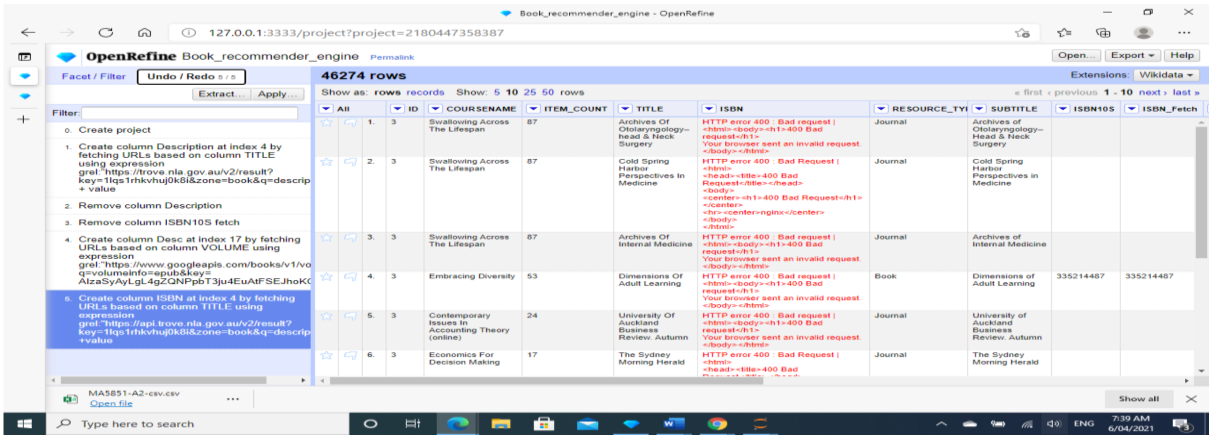
Expression   Language: General Refine Expression Language (GREL)

```
"https://api.trove.nla.gov.au/v2/result?
key=e3h60sjuu4e2rk7e&zone=book&q=isbn:" + value
```

No syntax error.

row	value	"https://api.trove.nla.gov.au/ ...
1.	null	null
2.	null	null
3.	null	null
4.	335214487	https://api.trove.nla.gov.au/v2/result?key=e3h60sjuu4e2rk7e&zone=book&q=isbn:335214487
5.	null	null
6.	null	null

Figure 1.4


Book_recommender_engine - OpenRefine

127.0.0.1:3333/project?project=2180447358387

OpenRefine Book_recommender_engine   46274 rows

Undo / Redo history:

0. Create project
1. Create column Description at index 4 by fetching URLs based on column TITLE using expression grel:"https://trove.nla.gov.au/v2/result?key=1lqs1rhkvhuj0k8i&zone=book&q=descrip + value
2. Remove column Description
3. Remove column ISBN10S fetch
4. Create column Desc at index 17 by fetching URLs based on column VOLUME using expression grel:"https://www.googleapis.com/books/v1/vo q=volumeinfo=epub&key=AlzaSyAyLgL4gZQNPpbT3ju4EuAtFSEJhoK
5. Create column ISBN at index 4 by fetching URLs based on column TITLE using expression grel:"https://api.trove.nla.gov.au/v2/result?key=1lqs1rhkvhuj0k8i&zone=book&q=descrip +value

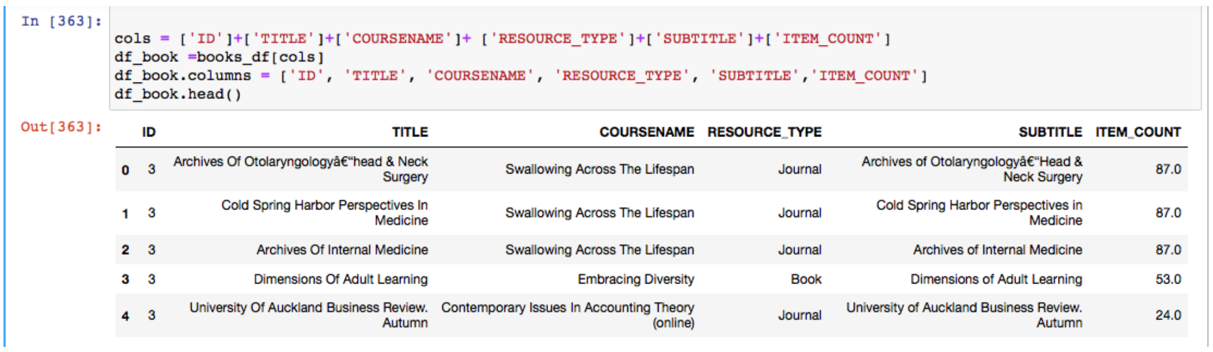
	ID	COURSENAME	ITEM_COUNT	TITLE	ISBN	RESOURCE_TY	SUBTITLE	ISBN10S	ISBN_Fetch
1.	3	Swallowing Across The Lifespan	87	Archives Of Otolaryngology–head & Neck Surgery	HTTP error 400 : Bad request | <html><body><h1>400 Bad request</h1> Your browser sent an invalid request. </body></html>	Journal	Archives of Otolaryngology–Head & Neck Surgery		
2.	3	Swallowing Across The Lifespan	87	Cold Spring Harbor Perspectives In Medicine	HTTP error 400 : Bad Request | <html> <head><title>400 Bad Request</title></head> <body> <center><h1>400 Bad Request</h1> </center> <hr><center>nginx</center> </body> </html>	Journal	Cold Spring Harbor Perspectives in Medicine		
3.	3	Swallowing Across The Lifespan	87	Archives Of Internal Medicine	HTTP error 400 : Bad request | <html><body><h1>400 Bad request</h1> Your browser sent an invalid request. </body></html>	Journal	Archives of Internal Medicine		
4.	3	Embracing Diversity	53	Dimensions Of Adult Learning	HTTP error 400 : Bad request | <html><body><h1>400 Bad request</h1> Your browser sent an invalid request. </body></html>	Book	Dimensions of Adult Learning	335214487	335214487
5.	3	Contemporary Issues In Accounting Theory (online)	24	University Of Auckland Business Review. Autumn	HTTP error 400 : Bad request | <html><body><h1>400 Bad request</h1> Your browser sent an invalid request. </body></html>	Journal	University of Auckland Business Review. Autumn		
6.	3	Economics For Decision Making	17	The Sydney Morning Herald	HTTP error 400 : Bad Request | <html> <head><title>400 Bad	Journal	The Sydney Morning Herald		

Figure 1.5.

Step 10. Exporting the dataset by clicking on export in the top right-hand corner and select.

Step 11. Import the dataset into python jupyter notebook IDE, all the required python libraries imported for the analysis like pandas, Numpy, Seaborn, Matplotlib, textwrap, and sklearn(metric, prepressing. The csv data file is then imported using read_csv() function predefined in pandas. The open refine data file contain 19 variables and 46,274 observation.

Six variables were selected for the analysis shown in the figure 1.6 below



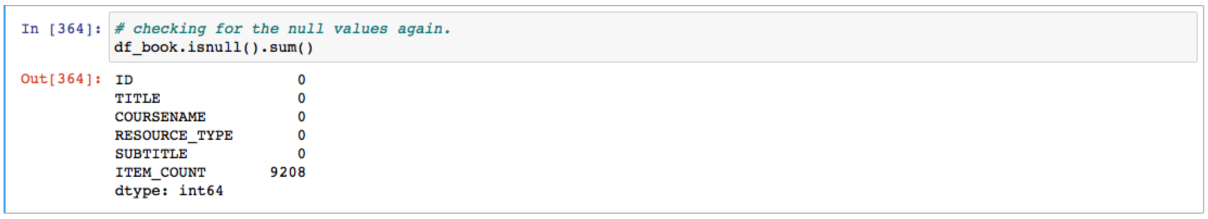
```
In [363]:
cols = ['ID']+['TITLE']+['COURSENAME']+ ['RESOURCE_TYPE']+['SUBTITLE']+['ITEM_COUNT']
df_book =books_df[cols]
df_book.columns = ['ID', 'TITLE', 'COURSENAME', 'RESOURCE_TYPE', 'SUBTITLE','ITEM_COUNT']
df_book.head()
```

Out[363]:

	ID	TITLE	COURSENAME	RESOURCE_TYPE	SUBTITLE	ITEM_COUNT
0	3	Archives Of Otolaryngologyâ€"head & Neck Surgery	Swallowing Across The Lifespan	Journal	Archives of Otolaryngologyâ€"Head & Neck Surgery	87.0
1	3	Cold Spring Harbor Perspectives In Medicine	Swallowing Across The Lifespan	Journal	Cold Spring Harbor Perspectives in Medicine	87.0
2	3	Archives Of Internal Medicine	Swallowing Across The Lifespan	Journal	Archives of Internal Medicine	87.0
3	3	Dimensions Of Adult Learning	Embracing Diversity	Book	Dimensions of Adult Learning	53.0
4	3	University Of Auckland Business Review. Autumn	Contemporary Issues In Accounting Theory (online)	Journal	University of Auckland Business Review. Autumn	24.0

Figure 1.6

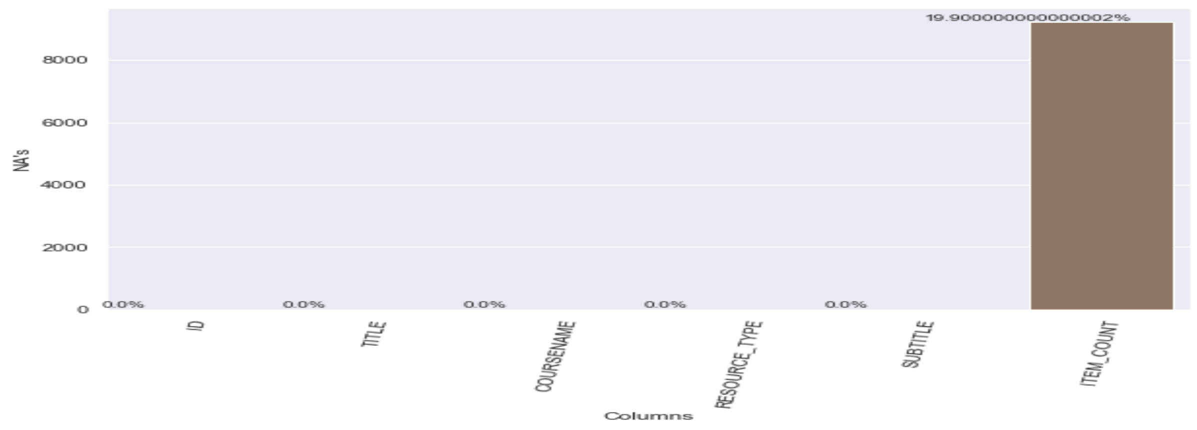
Check for the num value of each of the variable by using function isnull.sum() to sum all the nan on each variable.



```
In [364]: # checking for the null values again.
          df_book.isnull().sum()

Out[364]: ID                  0
          TITLE               0
          COURSENAME          0
          RESOURCE_TYPE       0
          SUBTITLE            0
          ITEM_COUNT       9208
          dtype: int64
```

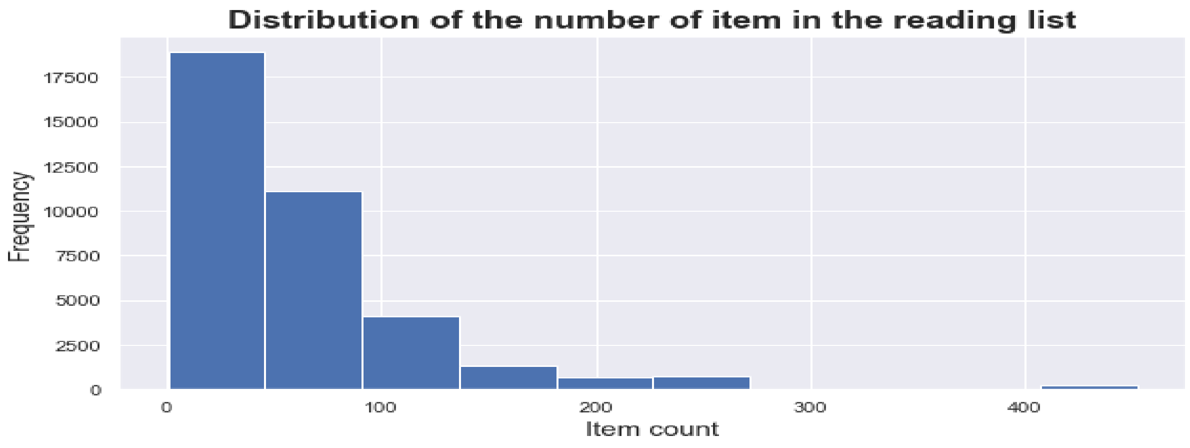
Figure 1.7

Then plotting each variable with the null values shown in figure 1.8

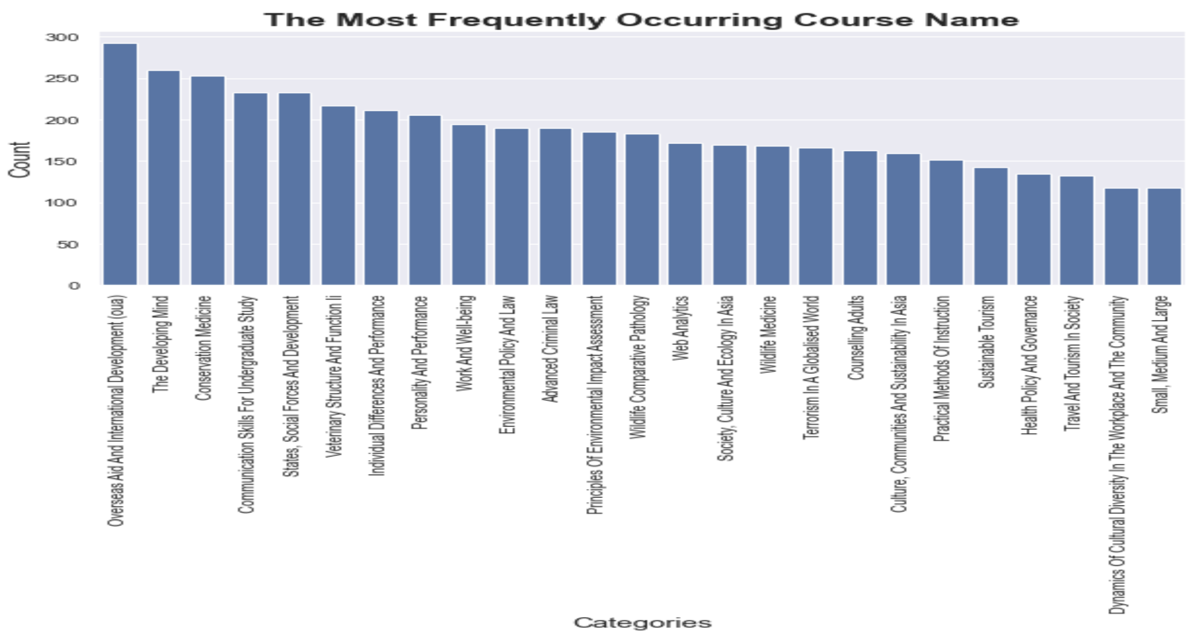


Which shown ITEM_COUNT having the highest null values of about 19.99 percent with the remaining variable having 0 percent.

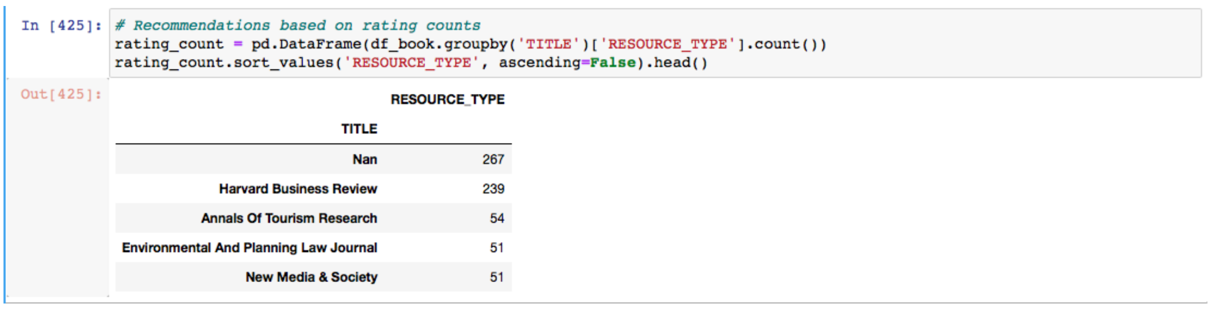
The distribution of the number of item in the reading list by plotting the ITEM_COUNT frequency shown in the figure 1.8



Let’s plot the course name in terms of their occurrence to get an insight into title terms of appearance. Figure 1.9



Recommendation based on rating count by grouping by title and resource type count using groupby() and count() function as illustrated on the table 1.0



```
In [425]: # Recommendations based on rating counts
          rating_count = pd.DataFrame(df_book.groupby('TITLE')['RESOURCE_TYPE'].count())
          rating_count.sort_values('RESOURCE_TYPE', ascending=False).head()
Out[425]:
```

	RESOURCE_TYPE
TITLE	
Nan	267
Harvard Business Review	239
Annals Of Tourism Research	54
Environmental And Planning Law Journal	51
New Media & Society	51

Resource type value count was plotted figure 2.0

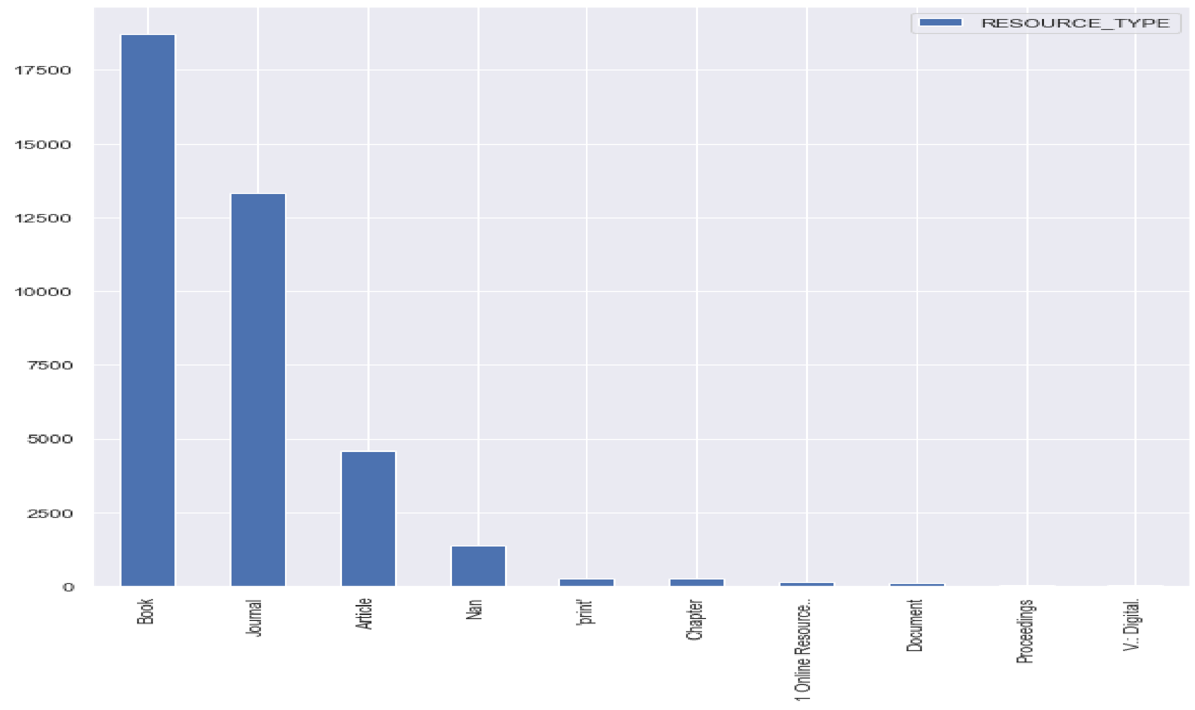
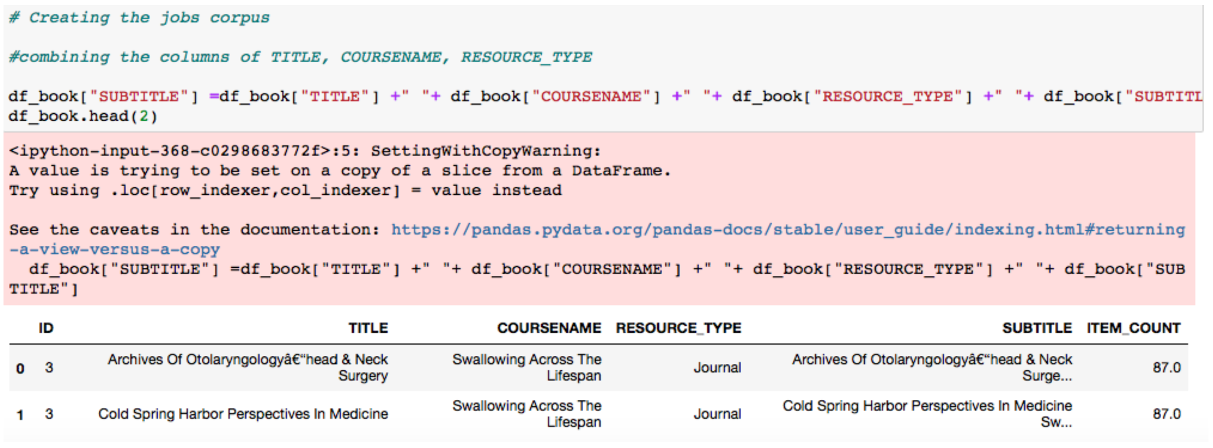


figure 2.0

The figure 2.0 shows that book have the highest frequency rate, journal, article.

Next step was Then create title corpus by combining TITLE, COURSENAME, RESOURCE_TYPE, ID, SUBTITLE, ITEM_COUNT.



```
# Creating the jobs corpus

#combining the columns of TITLE, COURSENAME, RESOURCE_TYPE

df_book["SUBTITLE"] =df_book["TITLE"] +" "+ df_book["COURSENAME"] +" "+ df_book["RESOURCE_TYPE"] +" "+ df_book["SUBTITL
df_book.head(2)
```

```
<ipython-input-368-c0298683772f>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  df_book["SUBTITLE"] =df_book["TITLE"] +" "+ df_book["COURSENAME"] +" "+ df_book["RESOURCE_TYPE"] +" "+ df_book["SUB
TITLE"]
```

	ID	TITLE	COURSENAME	RESOURCE_TYPE	SUBTITLE	ITEM_COUNT
0	3	Archives Of Otolaryngologyâ€"head & Neck Surgery	Swallowing Across The Lifespan	Journal	Archives Of Otolaryngologyâ€"head & Neck Surge...	87.0
1	3	Cold Spring Harbor Perspectives In Medicine	Swallowing Across The Lifespan	Journal	Cold Spring Harbor Perspectives In Medicine Sw...	87.0

Figure 2.1

3.0 Recommender Systems

As this computing has more textual data and there are no ratings available for any book**,** other matrix decomposition methods, such as correlation coefficient-based or Pearson's correlation could not be applied. Therefore, content-based filtering was applied. This approach showed how to recommend books to people based on the attributes of the books themselves. Four recommender systems were built using the following recommender engines:

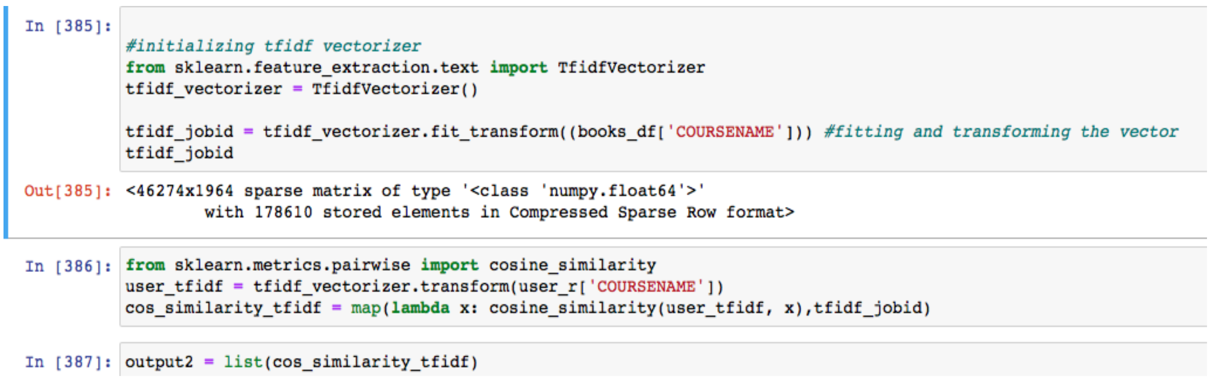
1. Content based Recommender with Term Frequency and Inverse Document frequency (TF -IDF)
2. Content Based Recommender with Count Vectorizer
3. Content Based Recommender with K- nearest neighbour (KNN)

This was achieved by measuring the similarity between the two-course names, because it must measure the **similarity** between two vectors. Its calculation is very efficient, especially for sparse vectors, as only the non-zero dimensions need to be considered. (Li 2013)

5.1 Cosine Similarity
This is the most widely used metric to measure the similarity between two vectors. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space.Its calculation is very efficient, particularly for sparse vectors, as only the non-zero sizes need to be considered. As a vital component, cosine similarity has been applied in solving different text mining problems, such as text classification, text summarization, information recovery, question answering, and many more. (Li 2013)
The general concept of the cosine similarity is if the cosine is close to 1 the items are similar, and if is close to 0 not similar, there is another scenario where cosine equal to -1, which means they are similar but opposite items.

4.0 Method:

Content based Recommender with Term Frequency and Inverse Document frequency (TF - IDF)
It calculate the cosine similarity of the two vectors by using python package from **sklearn**, the following code for a given user's course name illustrated in the figure 2.2

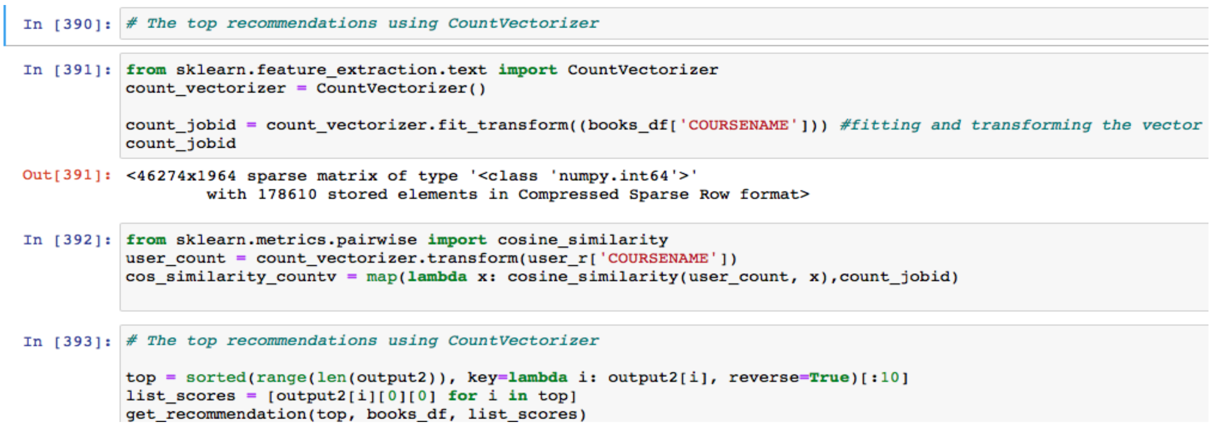


```
In [385]:
#initializing tfidf vectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()

tfidf_jobid = tfidf_vectorizer.fit_transform((books_df['COURSENAME'])) #fitting and transforming the vector
tfidf_jobid

Out[385]: <46274x1964 sparse matrix of type '<class 'numpy.float64'>'
        with 178610 stored elements in Compressed Sparse Row format>

In [386]: from sklearn.metrics.pairwise import cosine_similarity
user_tfidf = tfidf_vectorizer.transform(user_r['COURSENAME'])
cos_similarity_tfidf = map(lambda x: cosine_similarity(user_tfidf, x),tfidf_jobid)

In [387]: output2 = list(cos_similarity_tfidf)
```

Figure 2.2

Content Based Recommender with Count Vectorize

It provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

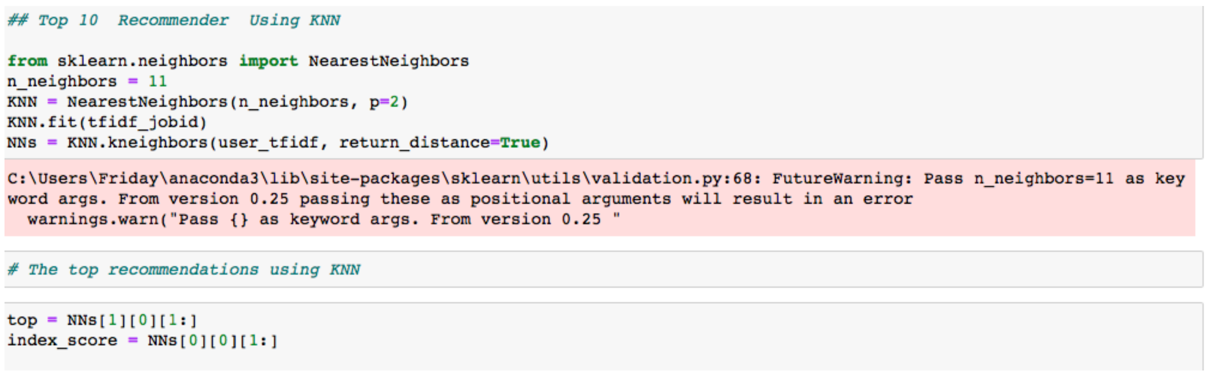


```
In [390]: # The top recommendations using CountVectorizer

In [391]: from sklearn.feature_extraction.text import CountVectorizer
count_vectorizer = CountVectorizer()

count_jobid = count_vectorizer.fit_transform((books_df['COURSENAME'])) #fitting and transforming the vector
count_jobid

Out[391]: <46274x1964 sparse matrix of type '<class 'numpy.int64'>'
        with 178610 stored elements in Compressed Sparse Row format>

In [392]: from sklearn.metrics.pairwise import cosine_similarity
user_count = count_vectorizer.transform(user_r['COURSENAME'])
cos_similarity_countv = map(lambda x: cosine_similarity(user_count, x),count_jobid)

In [393]: # The top recommendations using CountVectorizer

top = sorted(range(len(output2)), key=lambda i: output2[i], reverse=True)[:10]
list_scores = [output2[i][0][0] for i in top]
get_recommendation(top, books_df, list_scores)
```

4.1 KNN Recommender System

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

The code below, compute the 10 nearest neighbours for a given course name, features figure 2.4



```
## Top 10  Recommender  Using KNN

from sklearn.neighbors import NearestNeighbors
n_neighbors = 11
KNN = NearestNeighbors(n_neighbors, p=2)
KNN.fit(tfidf_jobid)
NNs = KNN.kneighbors(user_tfidf, return_distance=True)
```

```
C:\Users\Friday\anaconda3\lib\site-packages\sklearn\utils\validation.py:68: FutureWarning: Pass n_neighbors=11 as key
word args. From version 0.25 passing these as positional arguments will result in an error
  warnings.warn("Pass {} as keyword args. From version 0.25 "
```

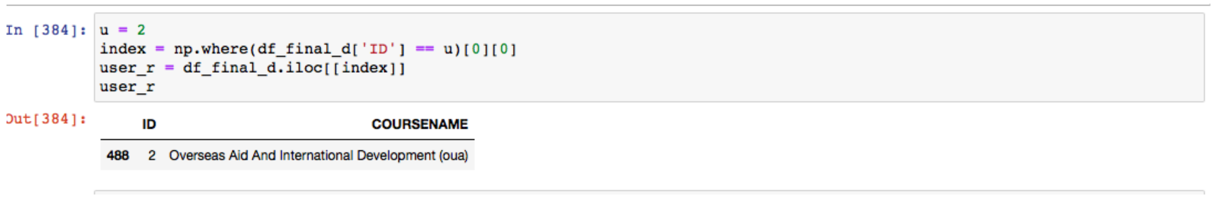
```
# The top recommendations using KNN
```

```
top = NNs[1][0][1:]
index_score = NNs[0][0][1:]
```

figure 2.4

4 To evaluate the recommendations.

Building recommendations systems using TF-IDF, count vectorizer, and cosine similarity using mainly text data and because there is not predefined testing matrix available for getting the accuracy score thus, the recommendations relevance has been checked manually. A random course name was selected randomly from the dataset.



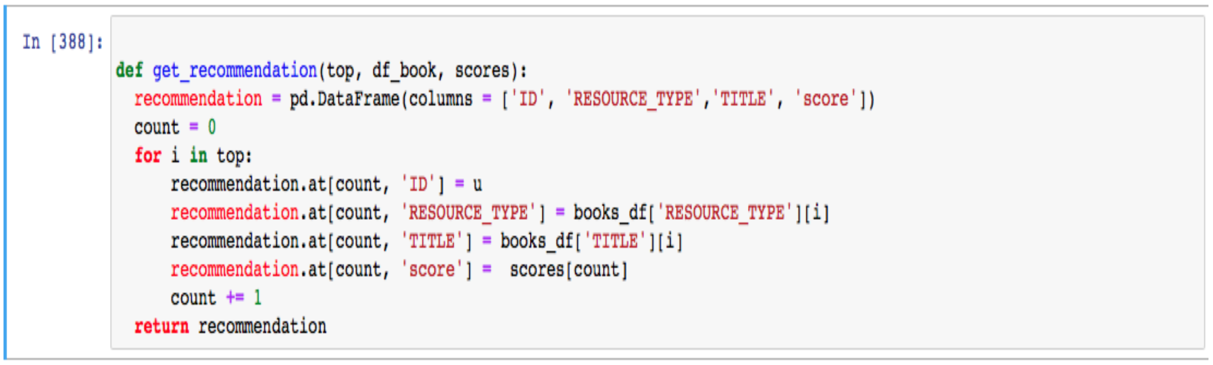
```
In [384]: u = 2
          index = np.where(df_final_d['ID'] == u)[0][0]
          user_r = df_final_d.iloc[[index]]
          user_r
```

Out[384]:

	ID	COURSENAME
488	2	Overseas Aid And International Development (oua)

Creating a function:

A function is created with a recommendation engine to called the first top 10, dataset, and scores and return recommendation.



```
In [388]:
def get_recommendation(top, df_book, scores):
  recommendation = pd.DataFrame(columns = ['ID', 'RESOURCE_TYPE','TITLE', 'score'])
  count = 0
  for i in top:
      recommendation.at[count, 'ID'] = u
      recommendation.at[count, 'RESOURCE_TYPE'] = books_df['RESOURCE_TYPE'][i]
      recommendation.at[count, 'TITLE'] = books_df['TITLE'][i]
      recommendation.at[count, 'score'] =  scores[count]
      count += 1
  return recommendation
```

Figure 2.5

Section two

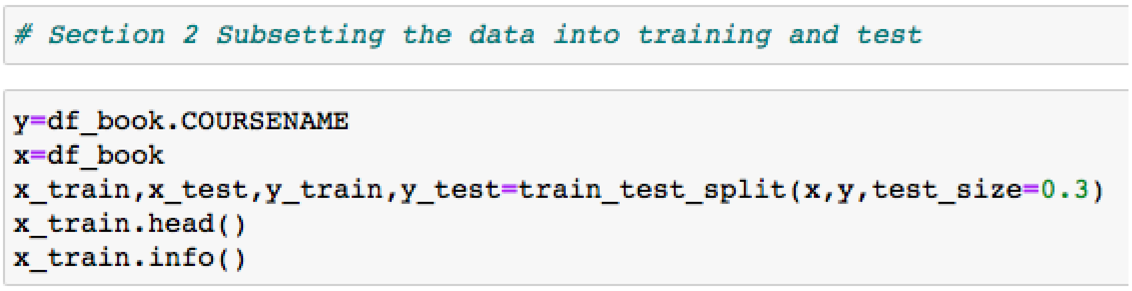
Sub-setting dataset using a train and test data to

Train-Test Split Estimation

The train-test split is a method for estimating the performance of a machine learning algorithm.

It can be used for regression or classification problems and it is mostly used in supervised machine learning

The technique entails taking a dataset and dividing it into two subsets called train and test. The first subset which is the train subset is used to fit the model. The second test subset is not used to train the model; as an alternative, the input part of the dataset is provided to the model, then predictions are produced and compared to the anticipated values. This second dataset The aim is to estimate the performance of the machine learning model on test dataset that is not used to train the data: data not used to train the data. With the dataset it is split into 70 person train and 30 test



```
# Section 2 Subsetting the data into training and test
```

```
y=df_book.COURSENAME
x=df_book
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
x_train.head()
x_train.info()
```

Figure 2.6

The original dataset is been split into input (*X*) and output (*y*) columns, then call the function split() in python by passing both arrays and have them split properly into train and test subsets.

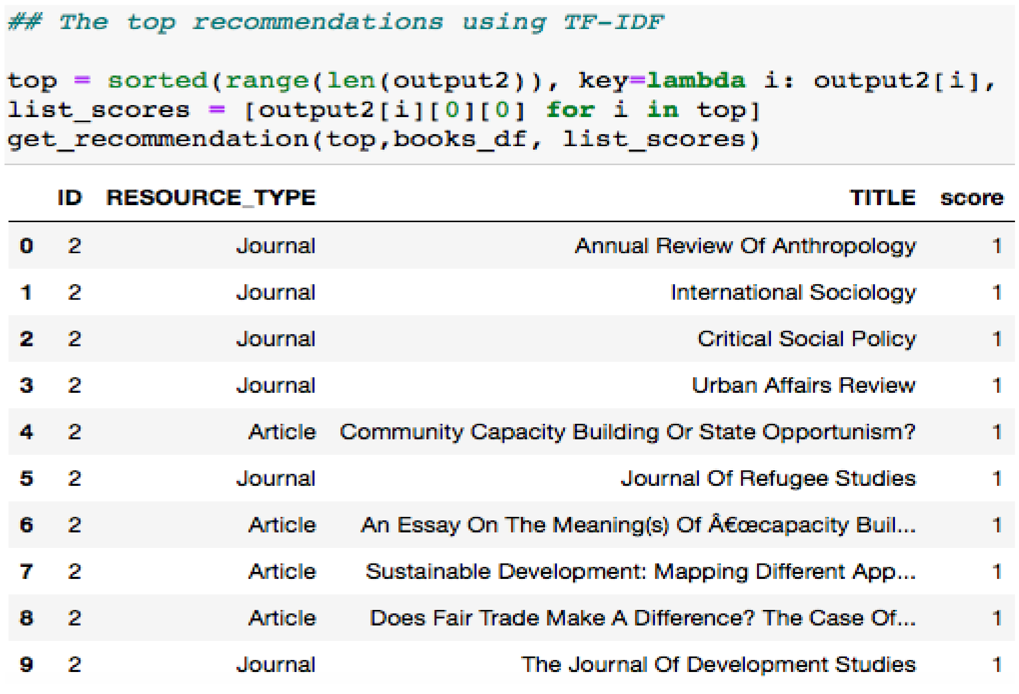
The train and test data is passed through the function called get_recommend()

To get the recommend for the top 10.

5.0 Result and Discussion

5.1 The Content based Recommender with Term Frequency and Inverse Document frequency (TF -IDF).

The Content based Recommender with Term Frequency and Inverse Document frequency (TF -IDF) result shows the top 10 recommended resources. The first item on the list is the most recommended while those at the bottom are the least recommended resources. The most recommended resource types are journals and articles (Figure 1.1), this may be because journals are reliable sources of information due to the vigorous peer review process associated with publishing in journals. Annual review of Anthropology was the most recommended resource based on the TF -IDF model, while the journal of development studies was the least recommended (Figure 1.1).



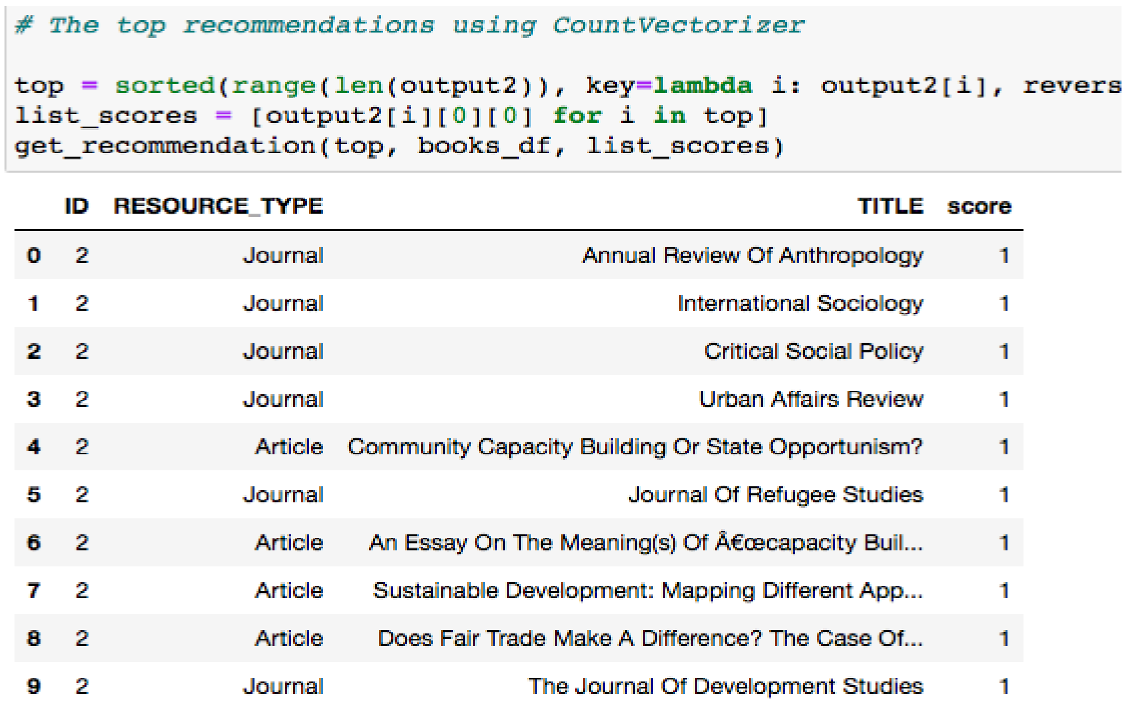
```
## The top recommendations using TF-IDF

top = sorted(range(len(output2)), key=lambda i: output2[i],
list_scores = [output2[i][0][0] for i in top]
get_recommendation(top,books_df, list_scores)
```

	ID	RESOURCE_TYPE	TITLE	score
0	2	Journal	Annual Review Of Anthropology	1
1	2	Journal	International Sociology	1
2	2	Journal	Critical Social Policy	1
3	2	Journal	Urban Affairs Review	1
4	2	Article	Community Capacity Building Or State Opportunism?	1
5	2	Journal	Journal Of Refugee Studies	1
6	2	Article	An Essay On The Meaning(s) Of Â€œcapacity Buil...	1
7	2	Article	Sustainable Development: Mapping Different App...	1
8	2	Article	Does Fair Trade Make A Difference? The Case Of...	1
9	2	Journal	The Journal Of Development Studies	1

Figure 1.1 Term Frequency and Inverse Document frequency (TF -IDF)

5.2 Content Based Recommender with Count Vectorize

Like the TF -IDF, the Count Vectorize mostly recommended journals (Figure 1.2). It also recommended the Annual review of Anthropology as the best resource and least recommended the journal of development studies (Figure 1.2).



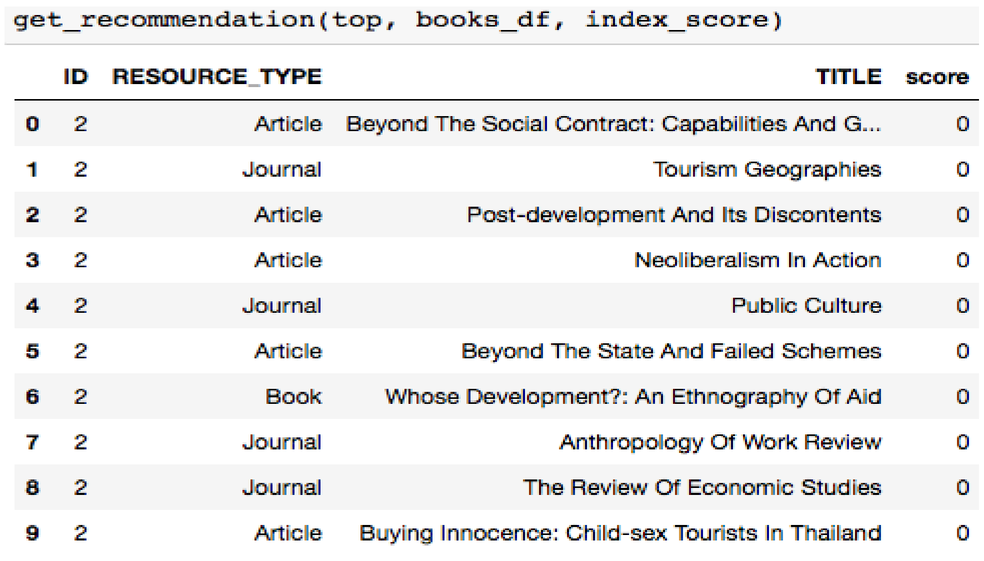
```
# The top recommendations using CountVectorizer

top = sorted(range(len(output2)), key=lambda i: output2[i], revers
list_scores = [output2[i][0][0] for i in top]
get_recommendation(top, books_df, list_scores)
```

	ID	RESOURCE_TYPE	TITLE	score
0	2	Journal	Annual Review Of Anthropology	1
1	2	Journal	International Sociology	1
2	2	Journal	Critical Social Policy	1
3	2	Journal	Urban Affairs Review	1
4	2	Article	Community Capacity Building Or State Opportunism?	1
5	2	Journal	Journal Of Refugee Studies	1
6	2	Article	An Essay On The Meaning(s) Of Â€œcapacity Buil...	1
7	2	Article	Sustainable Development: Mapping Different App...	1
8	2	Article	Does Fair Trade Make A Difference? The Case Of...	1
9	2	Journal	The Journal Of Development Studies	1

Figure 1.2 Content Based Recommender with Count Vectorize

5.3 Content Based Recommender with K- nearest neighbour (KNN)

The KNN model recommended mostly articles, then journals and one book. It mostly recommended the article with the title “Beyond the social contract: Capabilities , and recommended “Buying innocence: Child-sex tourist in Thailand. It also recommended one book “ Whose development?:An ethnography of aid”.



```
get_recommendation(top, books_df, index_score)
```

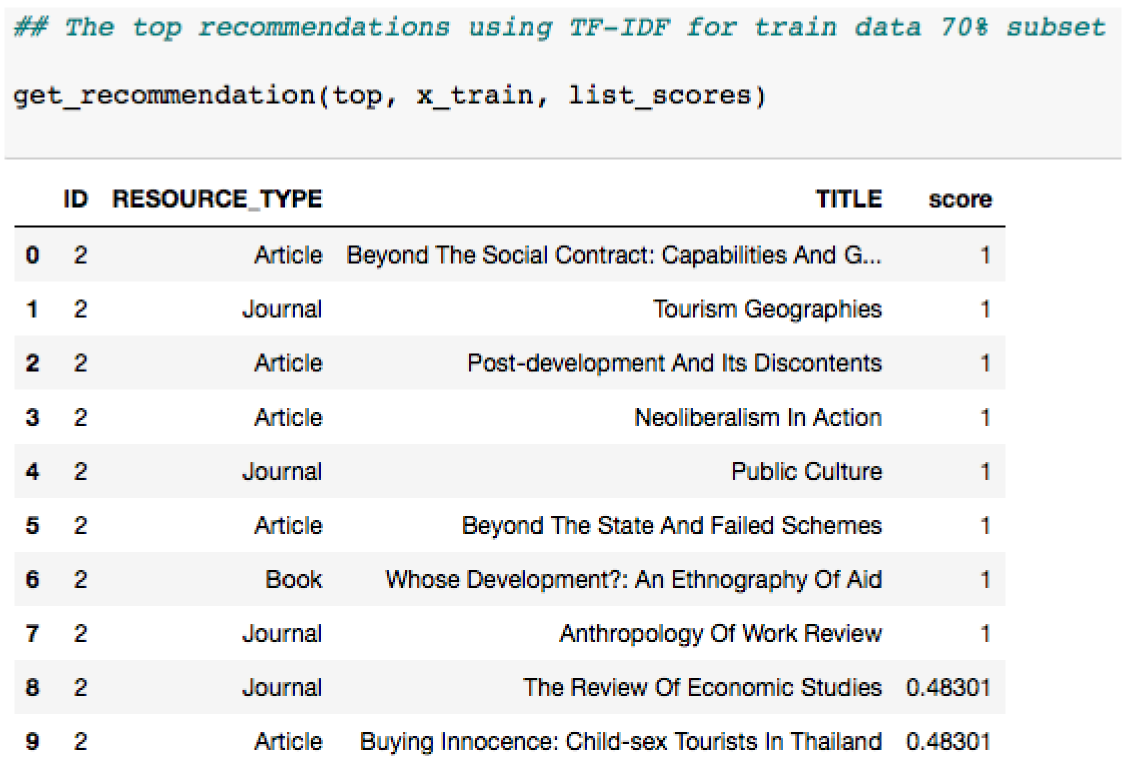
	ID	RESOURCE_TYPE	TITLE	score
0	2	Article	Beyond The Social Contract: Capabilities And G...	0
1	2	Journal	Tourism Geographies	0
2	2	Article	Post-development And Its Discontents	0
3	2	Article	Neoliberalism In Action	0
4	2	Journal	Public Culture	0
5	2	Article	Beyond The State And Failed Schemes	0
6	2	Book	Whose Development?: An Ethnography Of Aid	0
7	2	Journal	Anthropology Of Work Review	0
8	2	Journal	The Review Of Economic Studies	0
9	2	Article	Buying Innocence: Child-sex Tourists In Thailand	0

Figure 2.3 Content Based Recommender with K- nearest neighbour (KNN)

5.4 Differences and similarities between the Content Based Recommenders.

The result of the counter vectorizer was similar to the TF-IDF , they both recommended the same book title for the course name. However, the Content Based Recommender with K-nearest neighbour (KNN) recommended a different set of reading resources from the TF-IDF and countervectorizer recommender engine.

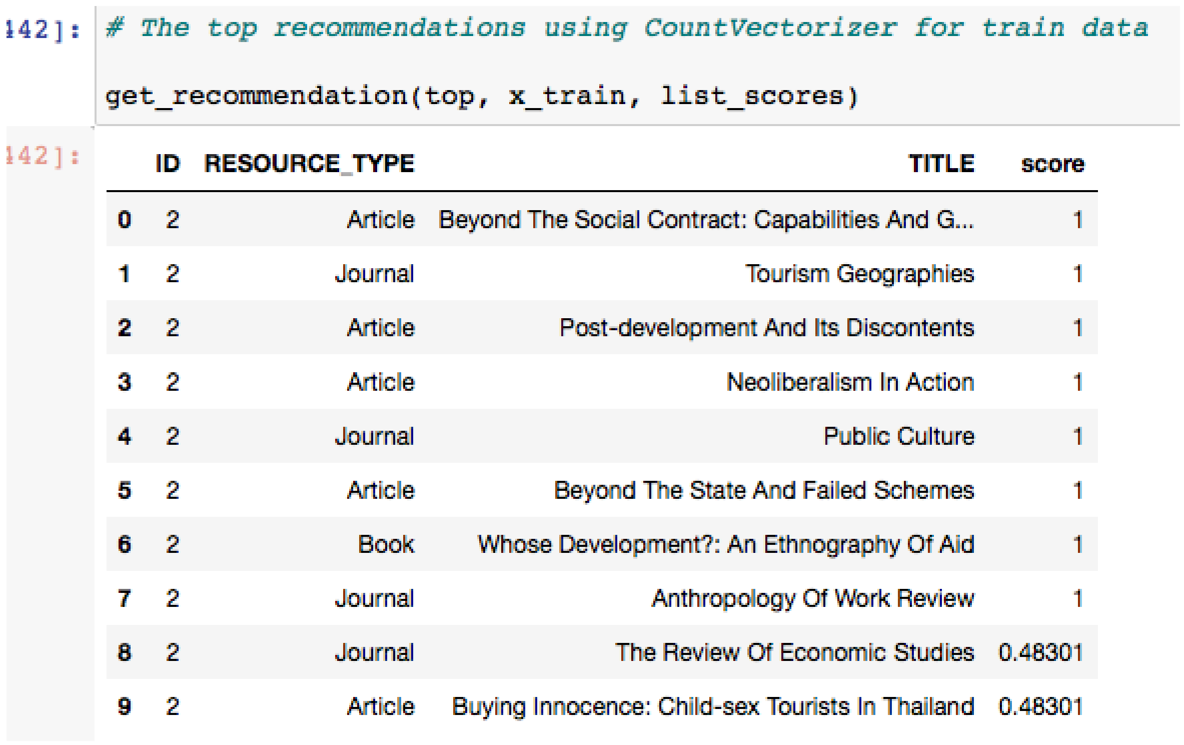
5.5 Train and test subset result:



```
## The top recommendations using TF-IDF for train data 70% subset

get_recommendation(top, x_train, list_scores)
```

	ID	RESOURCE_TYPE	TITLE	score
0	2	Article	Beyond The Social Contract: Capabilities And G...	1
1	2	Journal	Tourism Geographies	1
2	2	Article	Post-development And Its Discontents	1
3	2	Article	Neoliberalism In Action	1
4	2	Journal	Public Culture	1
5	2	Article	Beyond The State And Failed Schemes	1
6	2	Book	Whose Development?: An Ethnography Of Aid	1
7	2	Journal	Anthropology Of Work Review	1
8	2	Journal	The Review Of Economic Studies	0.48301
9	2	Article	Buying Innocence: Child-sex Tourists In Thailand	0.48301

The result is like recommendation using TF-IDF using the full dataset the difference is in the score. Which shows that the recommendation is almost 100 percent efficient.

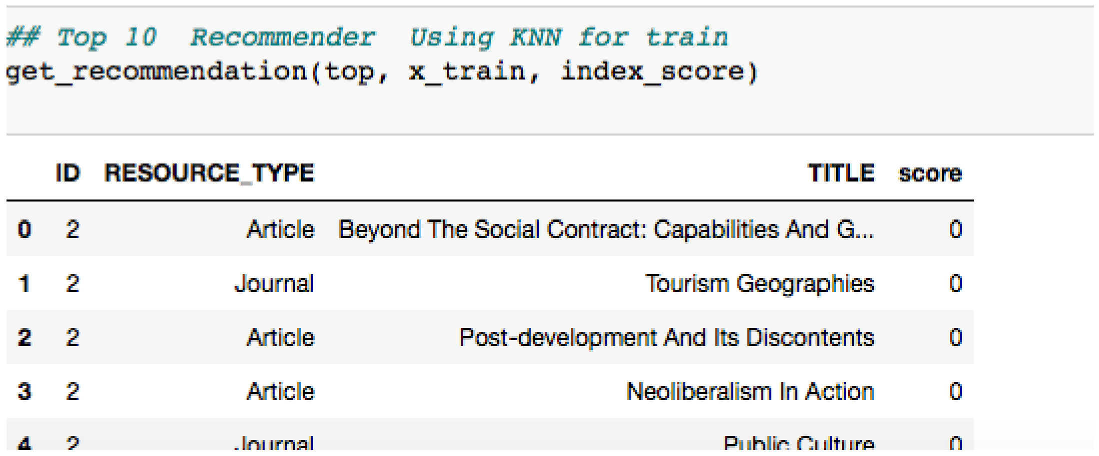


```
142]: # The top recommendations using CountVectorizer for train data

      get_recommendation(top, x_train, list_scores)
```

142]:

	ID	RESOURCE_TYPE	TITLE	score
0	2	Article	Beyond The Social Contract: Capabilities And G...	1
1	2	Journal	Tourism Geographies	1
2	2	Article	Post-development And Its Discontents	1
3	2	Article	Neoliberalism In Action	1
4	2	Journal	Public Culture	1
5	2	Article	Beyond The State And Failed Schemes	1
6	2	Book	Whose Development?: An Ethnography Of Aid	1
7	2	Journal	Anthropology Of Work Review	1
8	2	Journal	The Review Of Economic Studies	0.48301
9	2	Article	Buying Innocence: Child-sex Tourists In Thailand	0.48301

The countervectorizer is same as the TF-IDF recommendation on both the main dataset and the train dataset.



```
## Top 10  Recommender  Using KNN for train
get_recommendation(top, x_train, index_score)
```

	ID	RESOURCE_TYPE	TITLE	score
0	2	Article	Beyond The Social Contract: Capabilities And G...	0
1	2	Journal	Tourism Geographies	0
2	2	Article	Post-development And Its Discontents	0
3	2	Article	Neoliberalism In Action	0
4	2	Journal	Public Culture	0

The KNN recommendation is different from the other two recommender

kNN is a machine learning method to find clusters of similar course name based on common book title , and make predictions using the average title of top-k nearest neighbours.

Conclusion:

The counter vectorizer model recommended the same reading resources based on the course name. The K- nearest neighbour (KNN) recommended a different set of reading resources from the TF-IDF and counter vectorizer recommender engine. With the training dataset, all three models recommended the same resource.

References

Aditya Parameswaran, Petros Venetis, and Hector Garcia-Molina. 2011. Recommendation systems with complex constraints: A course recommendation perspective. ACM Transactions on Information Systems (TOIS) 29, 4 (2011), 20.

Agoritsa Polyzou, N Athanasios, and George Karypis. 2019. Scholars Walk: A Markov Chain Framework for Course Recommendation. In Proceedings of the 12th International Conference on Educational Data Mining. 396–401.

Britz, D. (2015). Understanding convolutional neural networks for NLP. *URL: http://www. wildml. com/2015/11/understanding-convolutional-neuralnetworks-for-nlp/(visited on 11/07/2015)*.

Charu C Aggarwal, "Content-based recommender systems" in Recommender Systems, Springer, pp. 139-166, 2016.

Jesus Bobadilla, Fernando Ortega, Antonio Hernando and Abraham Gutierrez, "Recommender systems survey", *Knowledge-Based Systems*, vol. 46, pp. 109-132, 2013.

Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang and Guangquan Zhang, "Recommender system application developments: a survey", *Decision Support Systems*, vol. 74, pp. 12-32, 2015.

Li, B., & Han, L. (2013, October). Distance weighted cosine similarity measure for text classification. In *International conference on intelligent data engineering and automated learning* (pp. 611-618). Springer, Berlin, Heidelberg.

Michael J Pazzani and Daniel Billsus, "Content-based recommendation systems" in The adaptive web, Springer, pp. 325-341, 2007.

Rosta Farzan and Peter Brusilovsky. 2011. Encouraging user participation in a course recommender system: An impact on user behavior. Computers in Human Behavior 27, 1 (2011), 276–284.

Sorathan Chaturapruek, Thomas Dee, Ramesh Johari, René Kizilcec, and Mitchell Stevens. 2018. How a data-driven course planning tool affects college students' GPA: evidence from two field experiments. (2018)

Weijie Jiang and Zachary A Pardos. 2019. Time slice imputation for personalized goal-based recommendation in higher education. In Proceedings of the 13th ACM Conference on Recommender Systems. ACM, 506–510

Weijie Jiang, Zachary A Pardos, and Qiang Wei. 2019. Goal-based course recommendation. In Proceedings of the 9th International Conference on Learning Analytics & Knowledge. ACM, 36–45

Zachary A Pardos, Zihao Fan, and Weijie Jiang. 2019. Connectionist recommendation in the wild: on the utility and scrutability of neural networks for personalized course guidance. User Model User-Adap Inter 29, 2 (2019), 487–525.

Zhiyun Ren, Xia Ning, Andrew S Lan, and Huzefa Rangwala. 2019. Grade Prediction Based on Cumulative Knowledge and Co-taken Courses. In Proceedings of the 12th International Conference on Educational Data Mining. 158–167