PART 2: PRODUCT MATCHING

In [1]:
```
# Importing libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from nltk.stem import PorterStemmer
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pickle
```

In [2]:
```
flipkart = pd.read_csv('flipkart.csv')
flipkart
```

Out[2]:

	uniq_id	crawl_timestamp	product_url	pr
0	c2d766ca982eca8304150849735ffef9	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	C
1	7f7036a6d550aaa89d34c77bd39a5e48	2016-03-25 22:59:23 +0000	http://www.flipkart.com/fabhomedecor-fabric-do...	Fal F
2	f449ec65dcbc041b6ae5e6a32717d01b	2016-03-25 22:59:23 +0000	http://www.flipkart.com/aw-bellies/p/itmeh4grg...	
3	0973b37acd0c664e3de26e97e5571454	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	C
4	bc940ea42ee6bef5ac7cea3fb5cfbee7	2016-03-25 22:59:23 +0000	http://www.flipkart.com/sicons-all-purpose-arn...	Pu D
...	...	...	...	
19995	7179d2f6c4ad50a17d014ca1d2815156	2015-12-01 10:15:43 +0000	http://www.flipkart.com/walldesign-small-vinyl...	
19996	71ac419198359d37b8fe5e3fffdfee09	2015-12-01 10:15:43 +0000	http://www.flipkart.com/wallmantra-large-vinyl...	Sti
19997	93e9d343837400ce0d7980874ece471c	2015-12-01 10:15:43 +0000	http://www.flipkart.com/elite-collection-mediu...	El Me
19998	669e79b8fa5d9ae020841c0c97d5e935	2015-12-01 10:15:43 +0000	http://www.flipkart.com/elite-collection-mediu...	El Me
19999	cb4fa87a874f715fff567f7b7b3be79c	2015-12-01 10:15:43 +0000	http://www.flipkart.com/elite-collection-mediu...	El Me

20000 rows × 15 columns

In [3]:
```
flipkart.shape
```

Out[3]:
```
(20000, 15)
```

In [4]:
```
flipkart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   uniq_id                  20000 non-null  object 
 1   crawl_timestamp          20000 non-null  object 
 2   product_url              20000 non-null  object 
 3   product_name             20000 non-null  object 
 4   product_category_tree    20000 non-null  object 
 5   pid                      20000 non-null  object 
 6   retail_price             19922 non-null  float64
 7   discounted_price         19922 non-null  float64
 8   image                    19997 non-null  object 
 9   is_FK_Advantage_product  20000 non-null  bool   
 10  description              19998 non-null  object 
 11  product_rating           20000 non-null  object 
 12  overall_rating           20000 non-null  object 
 13  brand                    14136 non-null  object 
 14  product_specifications   19986 non-null  object 
dtypes: bool(1), float64(2), object(12)
memory usage: 2.2+ MB
```

In [5]:
```
flipkart.describe()
```

Out[5]:

	retail_price	discounted_price
count	19922.000000	19922.000000
mean	2979.206104	1973.401767
std	9009.639341	7333.586040
min	35.000000	35.000000
25%	666.000000	350.000000
50%	1040.000000	550.000000
75%	1999.000000	999.000000
max	571230.000000	571230.000000

In [6]:
```
flipkart.isnull()
```

Out[6]:

	uniq_id	crawl_timestamp	product_url	product_name	product_category_tree	pid	retail_p
0	False	False	False	False	False	False	F
1	False	False	False	False	False	False	F
2	False	False	False	False	False	False	F
3	False	False	False	False	False	False	F
4	False	False	False	False	False	False	F
...	...	...	...	...	...	...	
19995	False	False	False	False	False	False	F
19996	False	False	False	False	False	False	F
19997	False	False	False	False	False	False	F
19998	False	False	False	False	False	False	F
19999	False	False	False	False	False	False	F

20000 rows × 15 columns

In [7]:
```
flipkart.isnull().sum()
```

Out[7]:
```
uniq_id                       0
crawl_timestamp               0
product_url                   0
product_name                  0
product_category_tree         0
pid                           0
retail_price                 78
discounted_price             78
image                         3
is_FK_Advantage_product       0
description                   2
product_rating                0
overall_rating                0
brand                      5864
product_specifications       14
dtype: int64
```

In [8]:
```
flipkart.product_name = flipkart.product_name.str.lower()
```

In [9]:
```
# analysing product name randomly from the datset
idx = np.random.randint(0, flipkart.shape[0])
flipkart.loc[idx, 'product_name']
```

Out[9]:
```
"scott international full sleeve solid men's sweatshirt"
```

In [10]:
```
# dropping duplicates records from the data
flipkart.drop_duplicates(subset='product_name', inplace=True)
```

In [11]:
```
flipkart.shape
```

Out[11]:
```
(12623, 15)
```

In [12]:
```
def preprocess(text):
    stemmer = PorterStemmer()
    text = ' '.join(stemmer.stem(word) for word in text.split())
    text = re.sub(r'[^a-zA-Z]', ' ', text).lower()  # consider only alphabets
    text = ' '.join(text.split()) # to remove spaces between the text
    return text
```

In [13]:
```
flipkart.image.fillna('https://logicacloud.eu/wp-content/themes/logica/images/No-
```

In [14]:
```
# function to clean links
def clean_image_links(link):
    link = link.strip('][').split(', ')[0]
    link = link.replace('"', '')
    return link
```

In [15]:
```
# preprocessing the product name
flipkart['product_name_processed'] = flipkart.product_name.apply(preprocess)
```

In [18]:
```
amazon = pd.read_csv('amz.csv' , encoding= 'unicode_escape')
amazon
```

Out[18]:

	uniq_id	crawl_timestamp	product_url	pr
0	c2d766ca982eca8304150849735ffef9	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	C
1	7f7036a6d550aaa89d34c77bd39a5e48	2016-03-25 22:59:23 +0000	http://www.flipkart.com/fabhomedecor-fabric-do...	Fal F
2	f449ec65dcbc041b6ae5e6a32717d01b	2016-03-25 22:59:23 +0000	http://www.flipkart.com/aw-bellies/p/itmeh4grg...	
3	0973b37acd0c664e3de26e97e5571454	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	C
4	bc940ea42ee6bef5ac7cea3fb5cfbee7	2016-03-25 22:59:23 +0000	http://www.flipkart.com/sicons-all-purpose-arn...	Pu D
...	...	...	...	
19995	7179d2f6c4ad50a17d014ca1d2815156	2015-12-01 10:15:43 +0000	http://www.flipkart.com/walldesign-small-vinyl...	W S
19996	71ac419198359d37b8fe5e3fffdfee09	2015-12-01 10:15:43 +0000	http://www.flipkart.com/wallmantra-large-vinyl...	W L
19997	93e9d343837400ce0d7980874ece471c	2015-12-01 10:15:43 +0000	http://www.flipkart.com/elite-collection-mediu...	C
19998	669e79b8fa5d9ae020841c0c97d5e935	2015-12-01 10:15:43 +0000	http://www.flipkart.com/elite-collection-mediu...	C
19999	cb4fa87a874f715fff567f7b7b3be79c	2015-12-01 10:15:43 +0000	http://www.flipkart.com/elite-collection-mediu...	C

20000 rows × 15 columns

In [19]:
```
amazon.shape
```
Out[19]:
```
(20000, 15)
```

In [20]:
```
amazon.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   uniq_id                  20000 non-null  object
 1   crawl_timestamp          20000 non-null  object
 2   product_url              20000 non-null  object
 3   product_name             20000 non-null  object
 4   product_category_tree    20000 non-null  object
 5   pid                      20000 non-null  object
 6   retail_price             20000 non-null  int64 
 7   discounted_price         20000 non-null  int64 
 8   image                    19997 non-null  object
 9   is_FK_Advantage_product  20000 non-null  bool  
 10  description              19998 non-null  object
 11  product_rating           20000 non-null  object
 12  overall_rating           20000 non-null  object
 13  brand                    14136 non-null  object
 14  product_specifications   19986 non-null  object
dtypes: bool(1), int64(2), object(12)
memory usage: 2.2+ MB
```

In [21]:
```
amazon.describe()
```
Out[21]:

	retail_price	discounted_price
count	20000.000000	20000.00000
mean	2957.095150	2364.59705
std	8993.993257	8994.62368
min	-20.000000	0.00000
25%	647.000000	424.00000
50%	999.000000	663.00000
75%	1986.000000	1235.00000
max	571223.000000	726879.00000

In [22]:
```
amazon.isnull()
```

Out[22]:

	uniq_id	crawl_timestamp	product_url	product_name	product_category_tree	pid	retail_p
0	False	False	False	False	False	False	F
1	False	False	False	False	False	False	F
2	False	False	False	False	False	False	F
3	False	False	False	False	False	False	F
4	False	False	False	False	False	False	F
...	...	...	...	...	...	...	
19995	False	False	False	False	False	False	F
19996	False	False	False	False	False	False	F
19997	False	False	False	False	False	False	F
19998	False	False	False	False	False	False	F
19999	False	False	False	False	False	False	F

20000 rows × 15 columns

In [23]:
```
amazon.isnull().sum()
```

Out[23]:
```
uniq_id                        0
crawl_timestamp                0
product_url                    0
product_name                   0
product_category_tree          0
pid                            0
retail_price                   0
discounted_price               0
image                          3
is_FK_Advantage_product        0
description                    2
product_rating                 0
overall_rating                 0
brand                       5864
product_specifications        14
dtype: int64
```

In [24]:
```
amazon.product_name = amazon.product_name.str.lower()
```

In [26]:
```
amazon.product_name
```
Out[26]:
```
0            alisha solid women's cycling shorts
1            fabhomedecor fabric double sofa bed
2                                     aw bellies
3            alisha solid women's cycling shorts
4          sicons all purpose arnica dog shampoo
                             ...
19995               walldesign small vinyl sticker
19996    wallmantra large vinyl stickers sticker
19997    elite collection medium acrylic sticker
19998    elite collection medium acrylic sticker
19999    elite collection medium acrylic sticker
Name: product_name, Length: 20000, dtype: object
```

In [27]:
```
# analysing product name randomly from the datset
idx = np.random.randint(0, amazon.shape[0])
amazon.loc[idx, 'product_name']
```
Out[27]:
```
'belkin play max modem router'
```

In [28]:
```
# dropping duplicates records from the data
amazon.drop_duplicates(subset='product_name', inplace=True)
```

In [29]:
```
amazon.shape
```
Out[29]:
```
(12628, 15)
```

In [30]:
```
# resetting index
amazon.reset_index(drop=True, inplace=True)
```

In [31]:
```
amazon.image.fillna('https://logicacloud.eu/wp-content/themes/logica/images/No-Im
```

In [33]:
```
amazon['product_name_processed'] = amazon.product_name.apply(preprocess)
```

In [34]:
```
flipkart.shape
```
Out[34]:
```
(12623, 16)
```

In [35]:
```
amazon.shape
```
Out[35]:
```
(12628, 16)
```

In [36]:
```
# creating instace of TfidfVectorizer
tfidfvectoriser_both = TfidfVectorizer()
```

In [40]:
```
# merging both the datasets
both = pd.concat([amazon, flipkart], axis=0)
both
```

Out[40]:

	uniq_id	crawl_timestamp	product_url	pro
0	c2d766ca982eca8304150849735ffef9	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	cy
1	7f7036a6d550aaa89d34c77bd39a5e48	2016-03-25 22:59:23 +0000	http://www.flipkart.com/fabhomedecor-fabric-do...	fal f
2	f449ec65dcbc041b6ae5e6a32717d01b	2016-03-25 22:59:23 +0000	http://www.flipkart.com/aw-bellies/p/itmeh4grg...	
3	bc940ea42ee6bef5ac7cea3fb5cfbee7	2016-03-25 22:59:23 +0000	http://www.flipkart.com/sicons-all-purpose-arn...	pu d
4	c2a17313954882c1dba461863e98adf2	2016-03-25 22:59:23 +0000	http://www.flipkart.com/eternal-gandhi-super-s...	ete s c
...	...	...	...	
19936	33b347833631a5040957c7c4b81f7ad7	2015-12-01 10:15:43 +0000	http://www.flipkart.com/purple-women-heels/p/i...	pu
19948	f2f027ad6a6df617c9f125173da71e44	2015-12-01 10:15:43 +0000	http://www.flipkart.com/uberlyfe-large-vinyl-s...	ul
19958	fb1ed40dd636c9eb3f8ccb281a04558d	2015-12-01 10:15:43 +0000	http://www.flipkart.com/witches-comfy-hues-wom...	
19962	725fb81399c181c86be89675429b8d27	2015-12-01 10:15:43 +0000	http://www.flipkart.com/stylistry-women-heels/...	styl
19976	d8b681d31a99ae133659764b3fc2e06a	2015-12-01 10:15:43 +0000	http://www.flipkart.com/uberlyfe-extra-large-v...	ul

25251 rows × 16 columns

In [41]:
```
both.shape
```
Out[41]:
```
(25251, 16)
```

In [42]:
```
# fitting the vectorizer on the entire corpus
tfidfvectoriser_both.fit(df_both.product_name_processed)
```
Out[42]:
```
TfidfVectorizer()
```

In [43]:
```
# creating vector representation of amazon dataset
tfidf_amazon = tfidfvectoriser_both.transform(amazon.product_name_processed)
```

In [44]:
```
tfidf_amazon.shape # (no. of records, embedding for each sentence)
```
Out[44]:
```
(12628, 9141)
```

In [49]:
```
# creating vector representation of flipkart dataset
tfidf_flipkart = tfidfvectoriser_both.transform(flipkart.product_name_processed)
```

In [50]:
```
tfidf_flipkart.shape
```
Out[50]:
```
(12623, 9141)
```

In [52]:
```
# calculating pairwise similarity for each flipkart product with respect to every
ps_flipkart = np.dot(tfidf_flipkart, tfidf_amazon.T).toarray()
ps_flipkart
```
Out[52]:
```
array([[1.        , 0.        , 0.        , ..., 0.04094593, 0.        ,
        0.        ],
       [0.        , 1.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 1.        , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.02327439, 0.        , 0.        , ..., 0.03089349, 0.        ,
        0.        ],
       [0.04094593, 0.        , 0.        , ..., 1.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.15089272,
        1.        ]])
```

In [55]:
```
# calculating pairwise similarity for each amazon product with respect to every f
ps_amazon = np.dot(tfidf_amazon, tfidf_flipkart.T).toarray()
ps_amazon
```

Out[55]:
```
array([[1.        , 0.        , 0.        , ..., 0.02327439, 0.04094593,
        0.        ],
       [0.        , 1.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 1.        , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.04094593, 0.        , 0.        , ..., 0.03089349, 1.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.15089272],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        1.        ]])
```

In [56]:
```
ps_flipkart.shape
```

Out[56]:
```
(12623, 12628)
```

In [57]:
```
ps_amazon.shape
```

Out[57]:
```
(12628, 12623)
```

In [59]:
```
# pick random product from amazon dataset
amazon.loc[823, 'product_name']
```

Out[59]:
```
'aroma care intense color nail polish combo 175007 49.5 ml'
```

In [62]:
```
# provide index of your selected product
np.argsort(ps_amazon[342])[::-1] # always exclude very first index (similarity wi
```

Out[62]:
```
array([  342,   341,  6571, ...,  6760,  6759, 12622], dtype=int64)
```

In [65]:
```
# considering second element from above result to find similar product in the fli
flipkart.loc[341, 'product_name']
```

Out[65]:
```
'asics gel-cumulus 17 running shoes'
```

In [73]:
```
# take product name from flipkart dataset
inp = flipkart.loc[idx, 'product_name'] # try replacing 899 with other index
print(inp)
```

```
belkin play max modem router
```

In [74]:
```
# preprocess the input
inp_f = preprocess(inp)
print(inp_f)
```

```
belkin play max modem router
```

In [75]:
```
# convert the text into vector embedding
embed = tfidfvectoriser_both.transform([inp_f])
embed
```

Out[75]:
```
<1x9141 sparse matrix of type '<class 'numpy.float64'>'
	with 5 stored elements in Compressed Sparse Row format>
```

In [76]:
```
# 1 sentence has vector embedding of size 9141
embed.shape
```

Out[76]:
```
(1, 9141)
```

In [79]:
```
# find the cosine similarity of our input sent with all sentence embedd of amazor
res = np.dot(embed, tfidf_amazon.T)
res
```

Out[79]:
```
<1x12628 sparse matrix of type '<class 'numpy.float64'>'
	with 114 stored elements in Compressed Sparse Row format>
```

In [80]:
```
# select the index which has highest similarity
idx = np.argmax(res)
idx
```

Out[80]:
```
3969
```

In [81]:
```
# display the most similar product from amazon dataset
amazon.loc[idx, 'product_name']
```

Out[81]:
```
'belkin play max modem router'
```