In [26]:
```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Drive already mounted at /content/drive/;
to attempt to forcibly remount, call driv
e.mount("/content/drive/", force_remount=
True).
```

In [27]:
```
%cd /content/drive/My Drive
```

```
/content/drive/My Drive
```

Amazon Apparel Recommendations

[4.2] Data and Code:

https://drive.google.com/open?id=0BwNkduBnePt2VWhCYXhMV3p4dTg

[4.3] Overview of the data

In [28]:
```
#import all the necessary packages.

from PIL import Image
import requests
from io import BytesIO
import matplotlib.pyplot as plt
```

```
import numpy as np
import pandas as pd
import warnings
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
import math
import time
import re
import os
import seaborn as sns
from collections import Counter
from sklearn.feature_extraction.text import Cou
ntVectorizer
from sklearn.feature_extraction.text import Tfi
dfVectorizer
from sklearn.metrics.pairwise import cosine_sim
ilarity
from sklearn.metrics import pairwise_distances
from matplotlib import gridspec
from scipy.sparse import hstack
import plotly
import plotly.figure_factory as ff
from plotly.graph_objs import Scatter, Layout

plotly.offline.init_notebook_mode(connected=Tru
e)
warnings.filterwarnings("ignore")
```

In [0]:

```
# we have give a json file which consists of al
l information about
# the products
# loading the data using pandas' read_json fil
```

```
e.
data = pd.read_json('tops_fashion.json')
```

In [0]:
```
print ('Number of data points : ', data.shape[0
], \
       'Number of features/variables:', data.sh
ape[1])
```

```
Number of data points :  183138 Number of
features/variables: 19
```

Terminology:

What is a dataset?
Rows and columns
Data-point
Feature/variable

In [0]:
```
# each product/item has 19 features in the raw
 dataset.
data.columns # prints column-names or feature-n
ames.
```

Out[0]:
```
Index(['asin', 'author', 'availability',
'availability_type', 'brand', 'color',
       'editorial_reivew', 'editorial_rev
iew', 'formatted_price',
       'large_image_url', 'manufacturer',
'medium_image_url', 'model',
       'product_type_name', 'publisher',
'reviews', 'sku', 'small_image_url',
       'title'],
      dtype='object')
```

Of these 19 features, we will be using only 6

features in this workshop.

```
1. asin  ( Amazon standard identificatio
n number)
2. brand ( brand to which the product be
longs to )
3. color ( Color information of apparel,
it can contain many colors as   a value
 ex: red and black stripes )
4. product_type_name (type of the appera
l, ex: SHIRT/TSHIRT )
5. medium_image_url  ( url of the image
 )
6. title (title of the product.)
7. formatted_price (price of the produc
t)
```

In [0]:

```
data = data[['asin', 'brand', 'color', 'medium_
image_url', 'product_type_name', 'title', 'form
atted_price']]
```

In [0]:

```
print ('Number of data points : ', data.shape[0
], \
       'Number of features:', data.shape[1])
data.head() # prints the top rows in the table.
```

```
Number of data points :  183138 Number of
features: 7
```

Out[0]:

	asin	brand	color	medium_imag
0	B016I2TS4W	FNC7C	None	https://images-r im amazon.com/ima
1				https://images-r

	asin	brand	color	medium_image
1	B01N49AKO6	FIG Clothing	None	https://images-r amazon.com/ima
2	B01JDPCOHO	FIG Clothing	None	https://images-r im amazon.com/ima
3	B01N19U5H5	Focal18	None	https://images-r im amazon.com/ima
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-r im amazon.com/ima

[5.1] Missing data for various features.

Basic stats for the feature: product_type_name

In [0]:

```
# We have total 72 unique type of product_type_
names
print(data['product_type_name'].describe())

# 91.62% (167794/183138) of the products are sh
irts,
```

```
count     183138
unique         72
top         SHIRT
```

```
freq      167794
Name: product_type_name, dtype: object
```

In [0]:

```python
# names of different product types
print(data['product_type_name'].unique())
```

```
['SHIRT' 'SWEATER' 'APPAREL' 'OUTDOOR_REC
REATION_PRODUCT'
 'BOOKS_1973_AND_LATER' 'PANTS' 'HAT' 'SP
ORTING_GOODS' 'DRESS' 'UNDERWEAR'
 'SKIRT' 'OUTERWEAR' 'BRA' 'ACCESSORY' 'A
RT_SUPPLIES' 'SLEEPWEAR'
 'ORCA_SHIRT' 'HANDBAG' 'PET_SUPPLIES' 'S
HOES' 'KITCHEN' 'ADULT_COSTUME'
 'HOME_BED_AND_BATH' 'MISC_OTHER' 'BLAZE
R' 'HEALTH_PERSONAL_CARE'
 'TOYS_AND_GAMES' 'SWIMWEAR' 'CONSUMER_EL
ECTRONICS' 'SHORTS' 'HOME'
 'AUTO_PART' 'OFFICE_PRODUCTS' 'ETHNIC_WE
AR' 'BEAUTY'
 'INSTRUMENT_PARTS_AND_ACCESSORIES' 'POWE
RSPORTS_PROTECTIVE_GEAR' 'SHIRTS'
 'ABIS_APPAREL' 'AUTO_ACCESSORY' 'NONAPPA
RELMISC' 'TOOLS' 'BABY_PRODUCT'
 'SOCKSHOSIERY' 'POWERSPORTS_RIDING_SHIR
T' 'EYEWEAR' 'SUIT'
 'OUTDOOR_LIVING' 'POWERSPORTS_RIDING_JAC
KET' 'HARDWARE' 'SAFETY_SUPPLY'
 'ABIS_DVD' 'VIDEO_DVD' 'GOLF_CLUB' 'MUSI
C_POPULAR_VINYL'
 'HOME_FURNITURE_AND_DECOR' 'TABLET_COMPU
TER' 'GUILD_ACCESSORIES'
 'ABIS_SPORTS' 'ART_AND_CRAFT_SUPPLY' 'BA
G' 'MECHANICAL_COMPONENTS'
 'SOUND_AND_RECORDING_EQUIPMENT' 'COMPUTE
R_COMPONENT' 'JEWELRY'
```

```
 'BUILDING_MATERIAL' 'LUGGAGE' 'BABY_COST
UME' 'POWERSPORTS_VEHICLE_PART'
 'PROFESSIONAL_HEALTHCARE' 'SEEDS_AND_PLA
NTS' 'WIRELESS_ACCESSORY']
```

In [0]:

```
# find the 10 most frequent product_type_names.
product_type_count = Counter(list(data['product
_type_name']))
product_type_count.most_common(10)
```

Out[0]:

```
[('SHIRT', 167794),
 ('APPAREL', 3549),
 ('BOOKS_1973_AND_LATER', 3336),
 ('DRESS', 1584),
 ('SPORTING_GOODS', 1281),
 ('SWEATER', 837),
 ('OUTERWEAR', 796),
 ('OUTDOOR_RECREATION_PRODUCT', 729),
 ('ACCESSORY', 636),
 ('UNDERWEAR', 425)]
```

Basic stats for the feature: brand

In [0]:

```
# there are 10577 unique brands
print(data['brand'].describe())

# 183138 - 182987 = 151 missing values.
```

```
count     182987
unique     10577
top         Zago

freq         223
Name: brand, dtype: object
```

In [0]:
```
brand_count = Counter(list(data['brand']))
brand_count.most_common(10)
```

Out[0]:
```
[('Zago', 223),
 ('XQS', 222),
 ('Yayun', 215),
 ('YUNY', 198),
 ('XiaoTianXin-women clothes', 193),
 ('Generic', 192),
 ('Boohoo', 190),
 ('Alion', 188),
 ('Abetteric', 187),
 ('TheMogan', 187)]
```

Basic stats for the feature: color

In [0]:
```
print(data['color'].describe())


# we have 7380 unique colors
# 7.2% of products are black in color
# 64956 of 183138 products have brand informati
on. That's approx 35.4%.
```

```
count     64956
unique     7380
top       Black
freq      13207
Name: color, dtype: object
```

In [0]:
```
color_count = Counter(list(data['color']))
color_count.most_common(10)
```

Out[0]:
```
[(None, 118182),
 ('Black', 13207),
```

```
 ('White', 8616),
 ('Blue', 3570),
 ('Red', 2289),
 ('Pink', 1842),
 ('Grey', 1499),
 ('*', 1388),
 ('Green', 1258),
 ('Multi', 1203)]
```

Basic stats for the feature: formatted_price

In [0]:

```
print(data['formatted_price'].describe())

# Only 28,395 (15.5% of whole data) products wi
th price information
```

```
count      28395
unique      3135
top       $19.99
freq         945
Name: formatted_price, dtype: object
```

In [0]:

```
price_count = Counter(list(data['formatted_pric
e']))
price_count.most_common(10)
```

Out[0]:

```
[(None, 154743),
 ('$19.99', 945),
 ('$9.99', 749),
 ('$9.50', 601),
 ('$14.99', 472),
 ('$7.50', 463),
 ('$24.99', 414),
```

```
 ('$29.99', 370),
 ('$8.99', 343),
 ('$9.01', 336)]
```

Basic stats for the feature: title

In [0]:
```
print(data['title'].describe())

# All of the products have a title.
# Titles are fairly descriptive of what the pro
duct is.
# We use titles extensively in this workshop
# as they are short and informative.
```

```
count
183138
unique
175985
top       Nakoda Cotton Self Print Straig
ht Kurti For Women
freq
77
Name: title, dtype: object
```

In [0]:
```
data.to_pickle('pickels/180k_apparel_data')
```

We save data files at every major step in our processing in "pickle" files. If you are stuck anywhere (or) if some code takes too long to run on your laptop, you may use the pickle files we give you to speed things up.

In [0]:
```
# consider products which have price informatio
n
```

```
# data['formatted_price'].isnull() => gives the
information
#about the dataframe row's which have null valu
es price == None|Null
data = data.loc[~data['formatted_price'].isnull
()]
print('Number of data points After eliminating
 price=NULL :', data.shape[0])
```

```
Number of data points After eliminating p
rice=NULL : 28395
```

In [0]:

```
# consider products which have color informatio
n
# data['color'].isnull() => gives the informati
on about the dataframe row's which have null va
lues price == None|Null
data =data.loc[~data['color'].isnull()]
print('Number of data points After eliminating
 color=NULL :', data.shape[0])
```

```
Number of data points After eliminating c
olor=NULL : 28385
```

We brought down the number of data points from 183K to 28K.

We are processing only 28K points so that most of the workshop participants can run this code on thier laptops in a reasonable amount of time.

For those of you who have powerful computers and some time to spare, you are recommended to use all of the 183K images.

In [0]:

```
data.to_pickle('pickels/28k_apparel_data')
```

In [0]:

```
# You can download all these 28k images using t
his code below.
# You do NOT need to run this code and hence it
is commented.


'''
from PIL import Image
import requests
from io import BytesIO

for index, row in images.iterrows():
        url = row['large_image_url']
        response = requests.get(url)
        img = Image.open(BytesIO(response.conte
nt))
        img.save('images/28k_images/'+row['asi
n']+'.jpeg')


'''
```

Out[0]:

```
"\nfrom PIL import Image\nimport requests
\nfrom io import BytesIO\n\nfor index, ro
w in images.iterrows():\n        url = ro
w['large_image_url']\n        response =
requests.get(url)\n        img = Image.op
en(BytesIO(response.content))\n        im
g.save('workshop/images/28k_images/'+row
['asin']+'.jpeg')\n\n\n"
```

[5.2] Remove near duplicate items

[5.2.1] Understand about duplicates.

In [0]:
```
# read data from pickle file from previous stag
e
data = pd.read_pickle('pickels/28k_apparel_dat
a')

# find number of products that have duplicate t
itles.
print(sum(data.duplicated('title')))
# we have 2325 products which have same title b
ut different color
```

```
2325
```

These shirts are exactly same except in size (S, M,L,XL)

:B00AQ4GMCK	:B00AQ4GMTS
:B00AQ4GMLQ	:B00AQ4GN3I

These shirts exactly same except in color

:B00G278GZ6	:B00G278W6O
:B00G278Z2A	:B00G2786X8

In our data there are many duplicate products like the above examples, we need to de-dupe

them for better results.

[5.2.2] Remove duplicates : Part 1

In [0]:
```
# read data from pickle file from previous stag
e
data = pd.read_pickle('pickels/28k_apparel_dat
a')
```

In [0]:
```
data.head()
```

Out[0]:

	asin	brand	color	medium_ima
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-ir amazon.com/im
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-ir amazon.com/im
11	B001LOUGE4	Fitness Etc.	Black	https://images-ir amazon.com/im
15	B003BSRPB0	FeatherLite	White	https://images-ir amazon.com/im
21	B014ICEDNA	FNC7C	Purple	https://images-ir amazon.com/im

In [0]:

```
# Remove All products with very few words in ti
tle
data_sorted = data[data['title'].apply(lambda x
: len(x.split())>4)]
print("After removal of products with short des
cription:", data_sorted.shape[0])
```

```
After removal of products with short desc
ription: 27949
```

In [0]:

```
# Sort the whole data based on title (alphabeti
cal order of title)
data_sorted.sort_values('title',inplace=True, a
scending=False)
data_sorted.head()
```

Out[0]:

	asin	brand	color	mediu
61973	B06Y1KZ2WB	Éclair	Black/Pink	https://i amazon.
133820	B010RV33VE	xiaoming	Pink	https://i amazon.
81461	B01DDSDLNS	xiaoming	White	https://i amazon.
75995				

	asin	brand	color	mediu
	B00X5LYO9Y	xiaoming	Red Anchors	https://i amazon.
151570	B00WPJG35K	xiaoming	White	https://i amazon.

Some examples of dupliacte titles that differ only in the last few words.

```
Titles 1:
16. woman's place is in the house and th
e senate shirts for Womens XXL White
17. woman's place is in the house and th
e senate shirts for Womens M Grey

Title 2:
25. tokidoki The Queen of Diamonds Wome
n's Shirt X-Large
26. tokidoki The Queen of Diamonds Wome
n's Shirt Small
27. tokidoki The Queen of Diamonds Wome
n's Shirt Large

Title 3:
61. psychedelic colorful Howling Galaxy
Wolf T-shirt/Colorful Rainbow Animal Pri
nt Head Shirt for woman Neon Wolf t-shir
t
62. psychedelic colorful Howling Galaxy
Wolf T-shirt/Colorful Rainbow Animal Pri
nt Head Shirt for woman Neon Wolf t-shir
t
```

```
63. psychedelic colorful Howling Galaxy
Wolf T-shirt/Colorful Rainbow Animal Pri
nt Head Shirt for woman Neon Wolf t-shir
t
64. psychedelic colorful Howling Galaxy
Wolf T-shirt/Colorful Rainbow Animal Pri
nt Head Shirt for woman Neon Wolf t-shir
t
```

In [0]:

```
indices = []
for i,row in data_sorted.iterrows():
    indices.append(i)
```

In [0]:

```
import itertools
stage1_dedupe_asins = []
i = 0
j = 0
num_data_points = data_sorted.shape[0]
while i < num_data_points and j < num_data_poin
ts:

    previous_i = i

    # store the list of words of ith string in
 a, ex: a = ['tokidoki', 'The', 'Queen', 'of',
 'Diamonds', 'Women's', 'Shirt', 'X-Large']
    a = data['title'].loc[indices[i]].split()

    # search for the similar products sequentia
lly
    j = i+1
    while j < num_data_points:

        # store the list of words of jth string
in b, ex: b = ['tokidoki', 'The', 'Queen', 'o
f', 'Diamonds', 'Women's', 'Shirt', 'Small']
```

```
        b = data['title'].loc[indices[j]].split
()

        # store the maximum length of two strin
gs
        length = max(len(a), len(b))

        # count is used to store the number of
 words that are matched in both strings
        count  = 0

        # itertools.zip_longest(a,b): will map
 the corresponding words in both strings, it wi
ll appened None in case of unequal strings
        # example: a =['a', 'b', 'c', 'd']
        # b = ['a', 'b', 'd']
        # itertools.zip_longest(a,b): will give
[('a','a'), ('b','b'), ('c','d'), ('d', None)]
        for k in itertools.zip_longest(a,b):
            if (k[0] == k[1]):
                count += 1

        # if the number of words in which both
 strings differ are > 2 , we are considering it
as those two apperals are different
        # if the number of words in which both
 strings differ are < 2 , we are considering it
as those two apperals are same, hence we are ig
noring them
        if (length - count) > 2: # number of wo
rds in which both sensences differ
            # if both strings are differ by mor
e than 2 words we include the 1st string index
            stage1_dedupe_asins.append(data_sor
ted['asin'].loc[indices[i]])
```

```
            # start searching for similar apper
als corresponds 2nd string
            i = j
            break
        else:
            j += 1
    if previous_i == i:
        break
```

In [0]:
```
data = data.loc[data['asin'].isin(stage1_dedupe
_asins)]
```

We removed the dupliactes which differ only at the end.

In [0]:
```
print('Number of data points : ', data.shape[0
])
```

```
Number of data points :  17593
```

In [0]:
```
data.to_pickle('pickels/17k_apperal_data')
```

[5.2.3] Remove duplicates : Part 2

```
In the previous cell, we sorted whole da
ta in alphabetical order of  titles.The
n, we removed titles which are adjacent
and very similar title

But there are some products whose titles
are not adjacent but very similar.
```

```
Examples:

Titles-1
86261.  UltraClub Women's Classic Wrinkl
e-Free Long Sleeve Oxford Shirt, Pink, X
X-Large
115042. UltraClub Ladies Classic Wrinkle
-Free Long-Sleeve Oxford Light Blue XXL

TItles-2
75004.  EVALY Women's Cool University Of
UTAH 3/4 Sleeve Raglan Tee
109225. EVALY Women's Unique University
Of UTAH 3/4 Sleeve Raglan Tees
120832. EVALY Women's New University Of
UTAH 3/4-Sleeve Raglan Tshirt
```

In [0]:

```
data = pd.read_pickle('pickels/17k_apperal_dat
a')
```

In [0]:

```
# This code snippet takes significant amount of
time.
# O(n^2) time.
# Takes about an hour to run on a decent comput
er.

indices = []
for i,row in data.iterrows():
    indices.append(i)

stage2_dedupe_asins = []
while len(indices)!=0:
    i = indices.pop()
    stage2_dedupe_asins.append(data['asin'].loc
[i])
    # consider the first apperal's title
```

```
    a = data['title'].loc[i].split()
    # store the list of words of ith string in
 a, ex: a = ['tokidoki', 'The', 'Queen', 'of',
 'Diamonds', 'Women's', 'Shirt', 'X-Large']
    for j in indices:

        b = data['title'].loc[j].split()
        # store the list of words of jth string
in b, ex: b = ['tokidoki', 'The', 'Queen', 'o
f', 'Diamonds', 'Women's', 'Shirt', 'X-Large']

        length = max(len(a),len(b))

        # count is used to store the number of
 words that are matched in both strings
        count  = 0

        # itertools.zip_longest(a,b): will map
 the corresponding words in both strings, it wi
ll appened None in case of unequal strings
        # example: a =['a', 'b', 'c', 'd']
        # b = ['a', 'b', 'd']
        # itertools.zip_longest(a,b): will give
[('a','a'), ('b','b'), ('c','d'), ('d', None)]
        for k in itertools.zip_longest(a,b):
            if (k[0]==k[1]):
                count += 1

        # if the number of words in which both
 strings differ are < 3 , we are considering it
as those two apperals are same, hence we are ig
noring them
        if (length - count) < 3:
            indices.remove(j)
```

In [0]:

```
# from whole previous products we will consider
only
# the products that are found in previous cell
data = data.loc[data['asin'].isin(stage2_dedupe
_asins)]
```

In [0]:
```
print('Number of data points after stage two of
dedupe: ',data.shape[0])
# from 17k apperals we reduced to 16k apperals
```

```
Number of data points after stage two of
dedupe:  16042
```

In [0]:
```
data.to_pickle('pickels/16k_apperal_data')
# Storing these products in a pickle file
# candidates who wants to download these files
 instead
# of 180K they can download and use them from t
he Google Drive folder.
```

6. Text pre-processing

In [0]:
```
data = pd.read_pickle('pickels/16k_apperal_dat
a')

# NLTK download stop words. [RUN ONLY ONCE]
# goto Terminal (Linux/Mac) or Command-Prompt
 (Window)
# In the temrinal, type these commands
# $python3
# $import nltk
# $nltk.download()
```

In [0]:

```
# we use the list of stop words that are downlo
aded from nltk lib.
stop_words = set(stopwords.words('english'))
print ('list of stop words:', stop_words)

def nlp_preprocessing(total_text, index, column
):
    if type(total_text) is not int:
        string = ""
        for words in total_text.split():
            # remove the special chars in revie
w like '"#$@!%^&*()_+-~?>< etc.
            word = ("".join(e for e in words if
e.isalnum()))
            # Conver all letters to lower-case
            word = word.lower()
            # stop-word removal
            if not word in stop_words:
                string += word + " "
        data[column][index] = string
```

```
list of stop words: {'such', 'and', 'her
s', 'up', 'she', 'd', 'further', 'all',

'than', 'under', 'is', 'off', 'both', 'mo
st', 'few', 'should', 're', 'very', 'jus
t', 'then', 'didn', 'myself', 'in', 'to
o', 's', 'shouldn', 'herself', 'because',
'how', 'itself', 'what', 'shan', 'weren',
'doing', 'them', 'couldn', 'their', 'so',
'ain', 'haven', 'yourself', 'now', 'll',
'isn', 'about', 'over', 'into', 'before',
'during', 'on', 'as', 'aren', 'against',
'above', 'down', 'they', 'below', 'me',
'again', 'for', 'why', 'been', 'yourselve
s', 'more', 'her', 'that', 'can', 'am',
```

```
 was ,  themselves ,  mighth ,  does ,  t
hose', 'only', 'hasn', 'any', 'ma', 'ar
e', 'nor', 'out', 'you', 'ourselves', 'th
e', 'an', 'has', 'where', 'i', 'while',
'ours', 'its', 'your', 'had', 'were', 'be
ing', 'no', 'or', 'needn', 've', 'y',
'a', 'each', 'have', 'through', 'when',
'mustn', 'by', 'won', 'from', 'own', 'wil
l', 'there', 't', 'him', 'these', 'does
n', 'theirs', 'my', 'did', 'of', 'who',
'until', 'wouldn', 'we', 'do', 'having',
'yours', 'other', 'wasn', 'it', 'with',
'once', 'here', 'don', 'o', 'whom', 'thi
s', 'if', 'but', 'hadn', 'our', 'some',
'm', 'not', 'between', 'himself', 'same',
'at', 'be', 'he', 'after', 'which', 'to',
'his'}
```

In [0]:

```
start_time = time.clock()
# we take each title and we text-preprocess it.
for index, row in data.iterrows():
    nlp_preprocessing(row['title'], index, 'tit
le')
# we print the time it took to preprocess whole
titles
print(time.clock() - start_time, "seconds")
```

```
3.5727220000000006 seconds
```

In [0]:

```
data.head()
```

Out[0]:

	asin	brand	color	medium_ima
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images- i amazon.com/im

	asin	brand	color	medium_ima
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images- i amazon.com/im
15	B003BSRPB0	FeatherLite	White	https://images- i amazon.com/im
27	B014ICEJ1Q	FNC7C	Purple	https://images- i amazon.com/im
46	B01NACPBG2	Fifth Degree	Black	https://images- i amazon.com/im

In [0]:
```
data.to_pickle('pickels/16k_apperal_data_prepro
cessed')
```

Stemming

In [0]:
```
from nltk.stem.porter import *
stemmer = PorterStemmer()
print(stemmer.stem('arguing'))
print(stemmer.stem('fishing'))


# We tried using stemming on our titles and it
 didnot work very well.
```

```
argu
```

```
fish
```

[8] Text based product similarity

In [0]:

```
data = pd.read_pickle('pickels/16k_apperal_data
_preprocessed')
data.head()
```

Out[0]:

	asin	brand	color	medium_ima
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images- i amazon.com/im
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images- i amazon.com/im
15	B003BSRPB0	FeatherLite	White	https://images- i amazon.com/im
27	B014ICEJ1Q	FNC7C	Purple	https://images- i amazon.com/im
46	B01NACPBG2	Fifth Degree	Black	https://images- i amazon.com/im

In [0]:

```
# Utility Functions which we will use through t
he rest of the workshop.


#Display an image
def display_img(url,ax,fig):
    # we get the url of the apparel and downloa
d it
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    # we will display it in notebook
    plt.imshow(img)

#plotting code to understand the algorithm's de
cision.
def plot_heatmap(keys, values, labels, url, tex
t):
        # keys: list of words of recommended ti
tle
        # values: len(values) ==  len(keys), va
lues(i) represents the occurence of the word ke
ys(i)
        # labels: len(labels) == len(keys), the
values of labels depends on the model we are us
ing
                # if model == 'bag of words': l
abels(i) = values(i)
                # if model == 'tfidf weighted b
ag of words':labels(i) = tfidf(keys(i))
                # if model == 'idf weighted bag
of words':labels(i) = idf(keys(i))
        # url : apparel's url

        # we will devide the whole figure into
 two parts
        gs = gridspec.GridSpec(2, 2, width_rati
```

```
os=[4,1], height_ratios=[4,1])
        fig = plt.figure(figsize=(25,3))

        # 1st, ploting heat map that represents
the count of commonly ocurred words in title2
        ax = plt.subplot(gs[0])
        # it displays a cell in white color if
 the word is intersection(lis of words of title
1 and list of words of title2), in black if not
        ax = sns.heatmap(np.array([values]), an
not=np.array([labels]))
        ax.set_xticklabels(keys) # set that axi
s labels as the words of title
        ax.set_title(text) # apparel title

        # 2nd, plotting image of the the appare
l
        ax = plt.subplot(gs[1])
        # we don't want any grid lines for imag
e and no labels on x-axis and y-axis
        ax.grid(False)
        ax.set_xticks([])
        ax.set_yticks([])

        # we call dispaly_img based with parame
te url
        display_img(url, ax, fig)

        # displays combine figure ( heat map an
d image together)
        plt.show()

def plot_heatmap_image(doc_id, vec1, vec2, url,
text, model):

    # doc_id : index of the title1
```

```
    # vec1 : input apparels's vector, it is of
 a dict type {word:count}
    # vec2 : recommended apparels's vector, it
 is of a dict type {word:count}
    # url : apparels image url
    # text: title of recomonded apparel (used t
o keep title of image)
    # model, it can be any of the models,
        # 1. bag_of_words
        # 2. tfidf
        # 3. idf

    # we find the common words in both titles,
 because these only words contribute to the dis
tance between two title vec's
    intersection = set(vec1.keys()) & set(vec2.
keys())

    # we set the values of non intersecting wor
ds to zero, this is just to show the difference
in heatmap
    for i in vec2:
        if i not in intersection:
            vec2[i]=0

    # for labeling heatmap, keys contains list
 of all words in title2
    keys = list(vec2.keys())
    #  if ith word in intersection(lis of words
of title1 and list of words of title2): values
(i)=count of that word in title2 else values(i)
=0
    values = [vec2[x] for x in vec2.keys()]

    # labels: len(labels) == len(keys), the val
ues of labels depends on the model we are using
```

```
        # if model == 'bag of words': labels(i)
= values(i)
        # if model == 'tfidf weighted bag of wo
rds':labels(i) = tfidf(keys(i))
        # if model == 'idf weighted bag of word
s':labels(i) = idf(keys(i))

    if model == 'bag_of_words':
        labels = values
    elif model == 'tfidf':
        labels = []
        for x in vec2.keys():
            # tfidf_title_vectorizer.vocabulary
_ it contains all the words in the corpus
            # tfidf_title_features[doc_id, inde
x_of_word_in_corpus] will give the tfidf value
 of word in given document (doc_id)
            if x in  tfidf_title_vectorizer.voc
abulary_:
                labels.append(tfidf_title_featu
res[doc_id, tfidf_title_vectorizer.vocabulary_[
x]])
            else:
                labels.append(0)
    elif model == 'idf':
        labels = []
        for x in vec2.keys():
            # idf_title_vectorizer.vocabulary_
 it contains all the words in the corpus
            # idf_title_features[doc_id, index_
of_word_in_corpus] will give the idf value of w
ord in given document (doc_id)
            if x in  idf_title_vectorizer.vocab
ulary_:
                labels.append(idf_title_feature
s[doc_id, idf_title_vectorizer.vocabulary_[x]])
```

```
            else:
                labels.append(0)

    plot_heatmap(keys, values, labels, url, tex
t)


# this function gets a list of wrods along with
the frequency of each
# word given "text"
def text_to_vector(text):
    word = re.compile(r'\w+')
    words = word.findall(text)
    # words stores list of all words in given s
tring, you can try 'words = text.split()' this
 will also gives same result
    return Counter(words) # Counter counts the
 occurence of each word in list, it returns dic
t type object {word1:count}



def get_result(doc_id, content_a, content_b, ur
l, model):
    text1 = content_a
    text2 = content_b

    # vector1 = dict{word11:#count, word12:#cou
nt, etc.}
    vector1 = text_to_vector(text1)

    # vector1 = dict{word21:#count, word22:#cou
nt, etc.}
    vector2 = text_to_vector(text2)
```

```
    plot_heatmap_image(doc_id, vector1, vector2
, url, text2, model)
```

[8.2] Bag of Words (BoW) on product titles.

In [0]:

```
from sklearn.feature_extraction.text import Cou
ntVectorizer
title_vectorizer = CountVectorizer()
title_features   = title_vectorizer.fit_transfo
rm(data['title'])
title_features.get_shape() # get number of rows
and columns in feature matrix.
# title_features.shape = #data_points * #words_
in_corpus
# CountVectorizer().fit_transform(corpus) retur
ns
# the a sparase matrix of dimensions #data_poin
ts * #words_in_corpus

# What is a sparse vector?

# title_features[doc_id, index_of_word_in_corpu
s] = number of times the word occured in that d
oc
```

Out[0]:

```
(16042, 12609)
```

In [0]:

```
def bag_of_words_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance fro
m given input apparel to all remaining apparels
```

```
    # the metric we used here is cosine, the co
side distance is mesured as K(X, Y) = <X, Y> /
 (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/me
trics.html#cosine-similarity
    pairwise_dist = pairwise_distances(title_fe
atures,title_features[doc_id])

    # np.argsort will return indices of the sma
llest distances
    indices = np.argsort(pairwise_dist.flatten
())[0:num_results]
    #pdists will store the smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[
0:num_results]

    #data frame indices of the 9 smallest dista
ce's
    df_indices = list(data.index[indices])

    for i in range(0,len(indices)):
        # we will pass 1. doc_id, 2. title1, 3.
title2, url, model
        get_result(indices[i],data['title'].loc
[df_indices[0]], data['title'].loc[df_indices[i
]], data['medium_image_url'].loc[df_indices[i
]], 'bag_of_words')
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print ('Brand:', data['brand'].loc[df_i
ndices[i]])
        print ('Title:', data['title'].loc[df_i
ndices[i]])
        print ('Euclidean similarity with the q
uery image :', pdists[i])
        print('='*60)
```

```
#call the bag-of-words model for a product to g
et similar products.
bag_of_words_model(12566, 20) # change the inde
x if you want to.
# In the output heat map each value represents
 the count value
# of the label word, the color represents the i
ntersection
# with inputs title.

#try 12566
#try 931
```

ASIN : B00JXQB5FQ
Brand: Si Row
Title: burnt umber tiger tshirt zebra str
ipes xl xxl
Euclidean similarity with the query image
: 0.0
==========
==========

ASIN : B00JXQASS6
Brand: Si Row
Title: pink tiger tshirt zebra stripes xl
xxl
Euclidean similarity with the query image
: 1.73205080757

==========

==========

ASIN : B00JXQCWTO
Brand: Si Row
Title: brown white tiger tshirt tiger st
ripes xl xxl
Euclidean similarity with the query image
: 2.44948974278
==========
==========

ASIN : B00JXQCUIC
Brand: Si Row
Title: yellow tiger tshirt tiger stripe
s l
Euclidean similarity with the query ima
ge : 2.64575131106
==========
==========

ASIN : B07568NZX4
Brand: Rustic Grace
Title: believed could tshirt
Euclidean similarity with the query image
: 3.0
==========
==========

ASIN : B01NB0NKRO
Brand: Ideology
Title: ideology graphic tshirt xl white
Euclidean similarity with the query image
: 3.0
==========
==========

ASIN : B00JXQAFZ2
Brand: Si Row
Title: grey white tiger tank top tiger
stripes xl xxl
Euclidean similarity with the query ima
ge : 3.0
==========
==========

ASIN : B01CLS8LMW
Brand: Awake
Title: morning person tshirt troll pictu
re xl
Euclidean similarity with the query image
: 3.16227766017
==========
==========

ASIN : B01KYZUD6C

ASIN : B01KVZUB6G
Brand: Merona
Title: merona green gold stripes
Euclidean similarity with the query image
: 3.16227766017
==========
==========

ASIN : B0733R2CJK
Brand: BLVD
Title: blvd womens graphic tshirt l
Euclidean similarity with the query ima
ge : 3.16227766017
==========
==========

ASIN : B012VQLT6Y
Brand: KM T-shirt
Title: km tiger printed sleeveless vest t
shirt
Euclidean similarity with the query image
: 3.16227766017
==========
==========

ASIN : B00JXQC8L6
Brand: Si Row

Title: blue peacock print tshirt l

Euclidean similarity with the query image
: 3.16227766017
==========
==========

ASIN : B06XC3CZF6
Brand: Fjallraven
Title: fjallraven womens ovik tshirt plu
m xxl
Euclidean similarity with the query image
: 3.16227766017
==========
==========

ASIN : B005IT8OBA
Brand: Hetalia
Title: hetalia us girl tshirt
Euclidean similarity with the query image
: 3.16227766017
==========
==========

ASIN : B0088PN0LA
Brand: Red House
Title: canvas 3001 30s tshirt kelly xl
Euclidean similarity with the query image

: 3.16227766017

: 3.16227766017
==========
==========

ASIN : B06X99V6WC
Brand: Brunello Cucinelli
Title: brunello cucinelli tshirt women wh
ite xl
Euclidean similarity with the query image
: 3.16227766017
==========
==========

ASIN : B06Y1JPW1Q
Brand: Xhilaration
Title: xhilaration womens lace tshirt sal
mon xxl
Euclidean similarity with the query image
: 3.16227766017
==========
==========

ASIN : B06X6GX6WG
Brand: Animal
Title: animal oceania tshirt yellow
Euclidean similarity with the query image
: 3.16227766017
==========
==========

```
ASIN : B017X8PW9U
Brand: Diesel
Title: diesel tserraf tshirt black
Euclidean similarity with the query image
: 3.16227766017
=========================================
====================
```

```
ASIN : B00IAA4JIQ
Brand: I Love Lucy
Title: juniors love lucywaaaaahhhh tshi
rt size xl
Euclidean similarity with the query ima
ge : 3.16227766017
========================================
=====================
```

[8.5] TF-IDF based product similarity

In [0]:

```
tfidf_title_vectorizer = TfidfVectorizer(min_df
= 0)
tfidf_title_features = tfidf_title_vectorizer.f
it_transform(data['title'])
# tfidf_title_features.shape = #data_points * #
words_in_corpus
```

```
# CountVectorizer().fit_transform(courpus) retu
rns the a sparase matrix of dimensions #data_po
ints * #words_in_corpus
# tfidf_title_features[doc_id, index_of_word_in
_corpus] = tfidf values of the word in given do
c
```

In [0]:

```
def tfidf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance fro
m given input apparel to all remaining apparels
    # the metric we used here is cosine, the co
side distance is mesured as K(X, Y) = <X, Y> /
 (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/me
trics.html#cosine-similarity
    pairwise_dist = pairwise_distances(tfidf_ti
tle_features,tfidf_title_features[doc_id])

    # np.argsort will return indices of 9 small
est distances
    indices = np.argsort(pairwise_dist.flatten
())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[
0:num_results]

    #data frame indices of the 9 smallest dista
ce's
    df_indices = list(data.index[indices])

    for i in range(0,len(indices)):
        # we will pass 1. doc_id, 2. title1, 3.
title2, url, model
```

```
        get_result(indices[i], data['title'].lo
c[df_indices[0]], data['title'].loc[df_indices[
i]], data['medium_image_url'].loc[df_indices[i
]], 'tfidf')
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print('BRAND :',data['brand'].loc[df_in
dices[i]])
        print ('Eucliden distance from the give
n image :', pdists[i])
        print('='*125)
tfidf_model(12566, 20)
# in the output heat map each value represents
 the tfidf values of the label word, the color
 represents the intersection with inputs title
```

```
ASIN : B00JXQB5FQ
BRAND : Si Row
Eucliden distance from the given image :
0.0
=========================================
=========================================
=========================================
==
```

```
ASIN : B00JXQASS6
BRAND : Si Row
Eucliden distance from the given image :
0.753633191245
=========================================
```

==========
==========
==

ASIN : B00JXQCWTO
BRAND : Si Row
Eucliden distance from the given image :
0.935764394377
==========
==========
==========
==

ASIN : B00JXQAFZ2
BRAND : Si Row
Eucliden distance from the given image :
0.95861535242
==========
==========
==========
==

ASIN : B00JXQCUIC
BRAND : Si Row
Eucliden distance from the given image :
1.00007496145
==========
==========

==========
==

ASIN : B00JXQAO94
BRAND : Si Row
Eucliden distance from the given image :
1.02321555246
==========
==========
==========
==

ASIN : B00JXQAUWA
BRAND : Si Row
Eucliden distance from the given image
: 1.0319918463
==========
==========
==========
========

ASIN : B06XSCVFT5
BRAND : Studio M
Eucliden distance from the given image :
1.21068436704
==========
==========
==========
==

ASIN : B06Y2GTYPM
BRAND : No Boundaries
Eucliden distance from the given image : 1.21216838107
==========

ASIN : B012VQLT6Y
BRAND : KM T-shirt
Eucliden distance from the given image : 1.21979064028
==========

ASIN : B06Y1VN8WQ
BRAND : Black Swan
Eucliden distance from the given image : 1.220684966
==========

ASIN : B00Z6HEXWI
BRAND : Black Temptation
Eucliden distance from the given image : 1.22128139212
==========

ASIN : B074TR12BH
BRAND : Ultra Flirt
Eucliden distance from the given image : 1.23133640946
==========

ASIN : B072R2JXKW
BRAND : WHAT ON EARTH
Eucliden distance from the given image : 1.23184519726
==========

ASIN : B074T8ZYGX
BRAND : MKP Crop Top
Eucliden distance from the given image : 1.23406074574
==========

ASIN : B071ZDF6T2
BRAND : Mossimo
Eucliden distance from the given image : 1.23527855777
==========

ASIN : B01K0H02OG
BRAND : Tultex
Eucliden distance from the given image : 1.23645729881
==========

```
ASIN : B00H8A6ZLI
BRAND : Vivian's Fashions
Eucliden distance from the given image :
1.24996155053
=========================================
=========================================
=========================================
==
```

```
ASIN : B010NN9RXO
BRAND : YICHUN
Eucliden distance from the given image
: 1.25354614209
========================================
========================================
========================================
========
```

```
ASIN : B06XBY5QXL
BRAND : Liz Claiborne
Eucliden distance from the given image :
1.25388329384
=========================================
=========================================
=========================================
==
```

[8.5] IDF based product

similarity

In [0]:
```
idf_title_vectorizer = CountVectorizer()
idf_title_features = idf_title_vectorizer.fit_t
ransform(data['title'])

# idf_title_features.shape = #data_points * #wo
rds_in_corpus
# CountVectorizer().fit_transform(courpus) retu
rns the a sparase matrix of dimensions #data_po
ints * #words_in_corpus
# idf_title_features[doc_id, index_of_word_in_c
orpus] = number of times the word occured in th
at doc
```

In [0]:
```
def n_containing(word):
    # return the number of documents which had
 the given word
    return sum(1 for blob in data['title'] if w
ord in blob.split())

def idf(word):
    # idf = log(#number of docs / #number of do
cs which had the given word)
    return math.log(data.shape[0] / (n_containi
ng(word)))
```

In [0]:
```
# we need to convert the values into float
idf_title_features  = idf_title_features.astype
(np.float)

for i in idf_title_vectorizer.vocabulary_.keys
():
    # for every word in whole corpus we will fi
```

```
nd its idf value
    idf_val = idf(i)

    # to calculate idf_title_features we need t
o replace the count values with the idf values
 of the word
    # idf_title_features[:, idf_title_vectorize
r.vocabulary_[i]].nonzero()[0] will return all
 documents in which the word i present
    for j in idf_title_features[:, idf_title_ve
ctorizer.vocabulary_[i]].nonzero()[0]:

        # we replace the count values of word i
in document j with  idf_value of word i
        # idf_title_features[doc_id, index_of_w
ord_in_courpus] = idf value of word
        idf_title_features[j,idf_title_vectoriz
er.vocabulary_[i]] = idf_val
```

In [0]:

```
def idf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance fro
m given input apparel to all remaining apparels
    # the metric we used here is cosine, the co
side distance is mesured as K(X, Y) = <X, Y> /
 (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/me
trics.html#cosine-similarity
    pairwise_dist = pairwise_distances(idf_titl
e_features,idf_title_features[doc_id])

    # np.argsort will return indices of 9 small
est distances
```

```
    indices = np.argsort(pairwise_dist.flatten
())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[
0:num_results]

    #data frame indices of the 9 smallest dista
ce's
    df_indices = list(data.index[indices])

    for i in range(0,len(indices)):
        get_result(indices[i],data['title'].loc
[df_indices[0]], data['title'].loc[df_indices[i
]], data['medium_image_url'].loc[df_indices[i
]], 'idf')
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print('Brand :',data['brand'].loc[df_in
dices[i]])
        print ('euclidean distance from the giv
en image :', pdists[i])
        print('='*125)



idf_model(12566,20)
# in the output heat map each value represents
 the idf values of the label word, the color re
presents the intersection with inputs title
```

```
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from the given image :
```

0.0
==========
==========
==========
==

ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from the given image : 12.2050713112
==========
==========
==========
==

ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from the given image : 14.4683626856
==========
==========
==========
==

ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from the given image : 14.4868329248
==========

==========
==========
==

ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from the given image : 14.8333929667
==========
==========
==========
==

ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from the given image : 14.8987445167
==========
==========
==========
========

ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from the given image : 15.2244582873
==========
==========

==========
==

ASIN : B074T8ZYGX
Brand : MKP Crop Top
euclidean distance from the given image : 17.0808129556
==========
==========
==========
==

ASIN : B00KF2N5PU
Brand : Vietsbay
euclidean distance from the given image : 17.0901681256
==========
==========
==========
========

ASIN : B00JPOZ9GM
Brand : Sofra
euclidean distance from the given image : 17.1532153376
==========
==========

==========

ASIN : B074T9KG9Q
Brand : Rain
euclidean distance from the given image : 17.3367152387
==========
==========
==========
==

ASIN : B00H8A6ZLI
Brand : Vivian's Fashions
euclidean distance from the given image : 17.410075941
==========
==========
==========
========

ASIN : B074G5G5RK
Brand : ERMANNO SCERVINO
euclidean distance from the given image : 17.5399213355
==========
==========
==========
==

ASIN : B06XSCVFT5
Brand : Studio M
euclidean distance from the given image : 17.6127585437
==========

ASIN : B06Y6FH453
Brand : Who What Wear
euclidean distance from the given image : 17.6237452825
==========

ASIN : B074V45DCX
Brand : Rain
euclidean distance from the given image : 17.6343424968
==========

ASIN : B07583CQFT
Brand : Very J
euclidean distance from the given image : 17.6375371274
==========

ASIN : B073GJGVBN
Brand : Ivan Levi
euclidean distance from the given image : 17.7230738913
==========

ASIN : B012VQLT6Y
Brand : KM T-shirt
euclidean distance from the given image : 17.7625885612
==========

```
ASIN : B00ZZMYBRG
Brand : HP-LEISURE
euclidean distance from the given image :
17.7795368647
=========================================
=========================================
=========================================
==
```

[9] Text Semantics based product similarity

In [0]:

```
# credits: https://www.kaggle.com/c/word2vec-nl
p-tutorial#part-2-word-vectors
# Custom Word2Vec using your own text data.
# Do NOT RUN this code.
# It is meant as a reference to build your own
 Word2Vec when you have
# lots of data.

'''
# Set values for various parameters
num_features = 300    # Word vector dimensional
ity
min_word_count = 1    # Minimum word count
num_workers = 4       # Number of threads to ru
n in parallel
context = 10          # Context window size
downsampling = 1e-3   # Downsample setting for
 frequent words

# Initialize and train the model (this will tak
e some time)
```

```
from gensim.models import word2vec
print ("Training model...")
model = word2vec.Word2Vec(sen_corpus, workers=n
um_workers, \
            size=num_features, min_count = min_
word_count, \
            window = context)

'''
```

In [0]:

```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

# in this project we are using a pretrained mod
el by google
# its 3.3G file, once you load this into your m
emory
# it occupies ~9Gb, so please do this step only
if you have >12G of ram
# we will provide a pickle file wich contains a
dict ,
# and it contains all our courpus words as keys
and  model[word] as values
# To use this code-snippet, download "GoogleNew
s-vectors-negative300.bin"
# from https://drive.google.com/file/d/0B7XkCwp
I5KDYNlNUTTlSS21pQmM/edit
# it's 1.9GB in size.

'''
model = KeyedVectors.load_word2vec_format('Goog
leNews-vectors-negative300.bin', binary=True)
'''
```

```
#if you do NOT have RAM >= 12GB, use the code b
elow.
with open('word2vec_model', 'rb') as handle:
    model = pickle.load(handle)
```

In [0]:

```
# Utility functions

def get_word_vec(sentence, doc_id, m_name):
    # sentence : title of the apparel
    # doc_id: document id in our corpus
    # m_name: model information it will take tw
o values
        # if  m_name == 'avg', we will append t
he model[i], w2v representation of word i
        # if m_name == 'weighted', we will mult
iply each w2v[word] with the idf(word)
    vec = []
    for i in sentence.split():
        if i in vocab:
            if m_name == 'weighted' and i in  i
df_title_vectorizer.vocabulary_:
                vec.append(idf_title_features[d
oc_id, idf_title_vectorizer.vocabulary_[i]] * m
odel[i])
            elif m_name == 'avg':
                vec.append(model[i])
        else:
            # if the word in our courpus is not
there in the google word2vec corpus, we are jus
t ignoring it
            vec.append(np.zeros(shape=(300,)))
    # we will return a numpy array of shape (#n
umber of words in title * 300 ) 300 = len(w2v_m
odel[word])
    # each row represents the word2vec represen
```

```
tation of each word (weighted/avg) in given sen
tance
    return  np.array(vec)

def get_distance(vec1, vec2):
    # vec1 = np.array(#number_of_words_title1 *
300), each row is a vector of length 300 corres
ponds to each word in give title
    # vec2 = np.array(#number_of_words_title2 *
300), each row is a vector of length 300 corres
ponds to each word in give title

    final_dist = []
    # for each vector in vec1 we caluclate the
 distance(euclidean) to all vectors in vec2
    for i in vec1:
        dist = []
        for j in vec2:
            # np.linalg.norm(i-j) will result t
he euclidean distance between vectors i, j
            dist.append(np.linalg.norm(i-j))
        final_dist.append(np.array(dist))
    # final_dist = np.array(#number of words in
title1 * #number of words in title2)
    # final_dist[i,j] = euclidean distance betw
een vectors i, j
    return np.array(final_dist)


def heat_map_w2v(sentence1, sentence2, url, doc
_id1, doc_id2, model):
    # sentance1 : title1, input apparel
    # sentance2 : title2, recommended apparel
    # url: apparel image url
    # doc_id1: document id of input apparel
    # doc_id2: document id of recommended appar
```

```
el
    # model: it can have two values, 1. avg 2.
 weighted

    #s1_vec = np.array(#number_of_words_title1
 * 300), each row is a vector(weighted/avg) of
 length 300 corresponds to each word in give ti
tle
    s1_vec = get_word_vec(sentence1, doc_id1, m
odel)
    #s2_vec = np.array(#number_of_words_title1
 * 300), each row is a vector(weighted/avg) of
 length 300 corresponds to each word in give ti
tle
    s2_vec = get_word_vec(sentence2, doc_id2, m
odel)

    # s1_s2_dist = np.array(#number of words in
title1 * #number of words in title2)
    # s1_s2_dist[i,j] = euclidean distance betw
een words i, j
    s1_s2_dist = get_distance(s1_vec, s2_vec)



    # devide whole figure into 2 parts 1st part
displays heatmap 2nd part displays image of app
arel
    gs = gridspec.GridSpec(2, 2, width_ratios=[
4,1],height_ratios=[2,1])
    fig = plt.figure(figsize=(15,15))

    ax = plt.subplot(gs[0])
    # ploting the heap map based on the pairwis
e distances
    ax = sns.heatmap(np.round(s1_s2_dist,4), an
```

```
not=True)
    # set the x axis labels as recommended appa
rels title
    ax.set_xticklabels(sentence2.split())
    # set the y axis labels as input apparels t
itle
    ax.set_yticklabels(sentence1.split())
    # set title as recommended apparels title
    ax.set_title(sentence2)

    ax = plt.subplot(gs[1])
    # we remove all grids and axis labels for i
mage
    ax.grid(False)
    ax.set_xticks([])
    ax.set_yticks([])
    display_img(url, ax, fig)

    plt.show()
```

In [0]:

```
# vocab = stores all the words that are there i
n google w2v model
# vocab = model.wv.vocab.keys() # if you are us
ing Google word2Vec

vocab = model.keys()
# this function will add the vectors of each wo
rd and returns the avg vector of given sentance
def build_avg_vec(sentence, num_features, doc_i
d, m_name):
    # sentace: its title of the apparel
    # num_features: the lenght of word2vec vect
or, its values = 300
    # m_name: model information it will take tw
o values
```

```
        # if  m_name == 'avg', we will append t
he model[i], w2v representation of word i
        # if m_name == 'weighted', we will mult
iply each w2v[word] with the idf(word)

    featureVec = np.zeros((num_features,), dtyp
e="float32")
    # we will intialize a vector of size 300 wi
th all zeros
    # we add each word2vec(wordi) to this fetur
eVec
    nwords = 0

    for word in sentence.split():
        nwords += 1
        if word in vocab:
            if m_name == 'weighted' and word in
idf_title_vectorizer.vocabulary_:
                featureVec = np.add(featureVec,
idf_title_features[doc_id, idf_title_vectorizer
.vocabulary_[word]] * model[word])
            elif m_name == 'avg':
                featureVec = np.add(featureVec,
model[word])
    if(nwords>0):
        featureVec = np.divide(featureVec, nwor
ds)
    # returns the avg vector of given sentance,
its of shape (1, 300)
    return featureVec
```

[9.2] Average Word2Vec product similarity.

In [0]:
```
doc_id = 0
w2v_title = []
# for every title we build a avg vector represe
ntation
for i in data['title']:
    w2v_title.append(build_avg_vec(i, 300, doc_
id,'avg'))
    doc_id += 1

# w2v_title = np.array(# number of doc in courp
us * 300), each row corresponds to a doc
w2v_title = np.array(w2v_title)
```

In [0]:
```
def avg_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # dist(x, y) = sqrt(dot(x, x) - 2 * dot(x,
 y) + dot(y, y))
    pairwise_dist = pairwise_distances(w2v_titl
e, w2v_title[doc_id].reshape(1,-1))

    # np.argsort will return indices of 9 small
est distances
    indices = np.argsort(pairwise_dist.flatten
())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[
0:num_results]

    #data frame indices of the 9 smallest dista
ce's
    df_indices = list(data.index[indices])

    for i in range(0, len(indices)):
        heat_map_w2v(data['title'].loc[df_indic
```

```
es[0]],data['title'].loc[df_indices[i]], data[
'medium_image_url'].loc[df_indices[i]], indices
[0], indices[i], 'avg')
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print('BRAND :',data['brand'].loc[df_in
dices[i]])
        print ('euclidean distance from given i
nput image :', pdists[i])
        print('='*125)


avg_w2v_model(12566, 20)
# in the give heat map, each cell contains the
 euclidean distance between words i, j
```

ASIN : B00JXQB5FQ

BRAND : Si Row

```
euclidean distance from given input image
: 0.000690534
==========================================
==========================================
==========================================
==
```

```
ASIN : B00JXQASS6
BRAND : Si Row
euclidean distance from given input ima
ge : 0.589193
==========================================
==========================================
==========================================
========
```

ASIN : B00JXQCWTO

BRAND : Si Row

euclidean distance from given input image : 0.700344

==========

ASIN : B00JXQAFZ2

BRAND : Si Row

euclidean distance from given input image : 0.89284

==========

```
ASIN : B00JXQCUIC
BRAND : Si Row
euclidean distance from given input ima
ge : 0.956013
=========================================
=========================================
=========================================
========
```

lsu tigers colosseum womens yellow purple slit back 12 sleeves tshirt

ASIN : B073R5Q8HD

BRAND : Colosseum

euclidean distance from given input image : 1.02297

==========

ASIN : B06XBY5QXL

BRAND : Liz Claiborne

euclidean distance from given input image : 1.06693

==========

ASIN : B01L8L73M2
BRAND : Hotgirl4 Raglan Design
euclidean distance from given input image : 1.07314
==========

ASIN : B01EJS5H06
BRAND : Vansty
euclidean distance from given input image : 1.07572
==========

==========

ASIN : B01BO1XRK8

BRAND : Le Bos

euclidean distance from given input image : 1.084

==========

==========

==========

========

ASIN : B072R2JXKW

BRAND : WHAT ON EARTH

euclidean distance from given input image : 1.08422

==========

ASIN : B074MJRGW6

BRAND : Two by Vince Camuto

euclidean distance from given input image : 1.0895

==========

ASIN : B00JXQCFRS

BRAND : Si Row

euclidean distance from given input image : 1.09006

==========

ASIN : B01I53HU6K

BRAND : ouxiuli

euclidean distance from given input image : 1.09201

==========

ASIN : B0711NGTQM

BRAND : THILFIGER RTW

euclidean distance from given input image : 1.09234

==========

ASIN : B01EFSLO8Y

BRAND : Vansty

euclidean distance from given input image : 1.0934

==========

ASIN : B0716TVWQ4
BRAND : THILFIGER RTW
euclidean distance from given input image : 1.0942
==========

ASIN : B0716MVPGV
BRAND : V.Secret
euclidean distance from given input image : 1.09483
==========

==========
==========
==

ASIN : B016OPN4OI
BRAND : TIKE Fashions
euclidean distance from given input image : 1.09513
==========
==========
==========
========

ASIN : B018WDJCUA

```
BRAND : INC - International Concepts Woma
n
euclidean distance from given input image
: 1.09669
=========================================
=========================================
=========================================
==
```

[9.4] IDF weighted Word2Vec for product similarity

In [0]:
```
doc_id = 0
w2v_title_weight = []
# for every title we build a weighted vector re
presentation
for i in data['title']:
    w2v_title_weight.append(build_avg_vec(i, 30
0, doc_id,'weighted'))
    doc_id += 1
# w2v_title = np.array(# number of doc in courp
us * 300), each row corresponds to a doc
w2v_title_weight = np.array(w2v_title_weight)
```

In [0]:
```
def weighted_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance fro
m given input apparel to all remaining apparels
```

```
    # the metric we used here is cosine, the co
side distance is mesured as K(X, Y) = <X, Y> /
 (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/me
trics.html#cosine-similarity
    pairwise_dist = pairwise_distances(w2v_titl
e_weight, w2v_title_weight[doc_id].reshape(1,-1
))

    # np.argsort will return indices of 9 small
est distances
    indices = np.argsort(pairwise_dist.flatten
())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[
0:num_results]

    #data frame indices of the 9 smallest dista
ce's
    df_indices = list(data.index[indices])

    for i in range(0, len(indices)):
        heat_map_w2v(data['title'].loc[df_indic
es[0]],data['title'].loc[df_indices[i]], data[
'medium_image_url'].loc[df_indices[i]], indices
[0], indices[i], 'weighted')
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print('Brand :',data['brand'].loc[df_in
dices[i]])
        print('euclidean distance from input :'
, pdists[i])
        print('='*125)

weighted_w2v_model(12566, 20)
#931
```

```
#12566
# in the give heat map, each cell contains the
 euclidean distance between words i, j
```

```
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.00390
625
=========================================
=========================================
=========================================
========
```

```
ASIN : B00JXQASS6
```

Brand : Si Row

euclidean distance from input : 4.06389

==========

ASIN : B00JXQCWTO

Brand : Si Row

euclidean distance from input : 4.77094

==========

ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 5.36016
==========
==========
==========
==

ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 5.68952
==========
==========
==========
========

ASIN : B00JXQAO94

Brand : Si Row

euclidean distance from input : 5.69302

==========

==========

==========

==

ASIN : B00JXQCUIC

Brand : Si Row

euclidean distance from input : 5.89344

==========

==========

==========

==

ASIN : B015H41F6G

Brand : KINGDE

euclidean distance from input : 6.13299

==========

ASIN : B073R5Q8HD

Brand : Colosseum

euclidean distance from input : 6.25671

==========

```
ASIN : B074P8MD22
Brand : Edista
euclidean distance from input : 6.3922
=========================================
=========================================
=========================================
========
```

```
ASIN : B00C0I3U3E
Brand : Stanzino
euclidean distance from input : 6.4149
=========================================
```

==========
==========
==

ASIN : B073R4ZM7Y
Brand : Colosseum
euclidean distance from input : 6.45096
==========
==========
==========
==

ASIN : B01C6ORLDQ

Brand : 1 Mad Fit

euclidean distance from input : 6.46341

==========

==========

==========

==

ASIN : B06XBY5QXL

Brand : Liz Claiborne

euclidean distance from input : 6.53922

==========

==========

==========

==

ASIN : B071YF3WDD

Brand : Merona

euclidean distance from input : 6.5755

==========

==========

==========

========

ASIN : B00H8A6ZLI

Brand : Vivian's Fashions

euclidean distance from input : 6.63821

==========
==========
==========
==

ASIN : B00Z6HEXWI
Brand : Black Temptation
euclidean distance from input : 6.66074
==========
==========
==========
==

```
ASIN : B00ILGH5OY
Brand : Ralph Lauren Active
euclidean distance from input : 6.68391
=========================================
=========================================
=========================================
========
```

```
ASIN : B06Y1VN8WQ
Brand : Black Swan
euclidean distance from input : 6.70576
=========================================
=========================================
=========================================
==
```

```
ASIN : B00KSNTY7Y
Brand : Anna-Kaci
euclidean distance from input : 6.70612
=========================================
=========================================
=========================================
==
```

[9.6] Weighted similarity using brand and color.

In [0]:

```
# some of the brand values are empty.
# Need to replace Null with string "NULL"
data['brand'].fillna(value="Not given", inplace
=True )

# replace spaces with hypen
brands = [x.replace(" ", "-") for x in data['br
and'].values]
types = [x.replace(" ", "-") for x in data['pro
duct_type_name'].values]
colors = [x.replace(" ", "-") for x in data['co
lor'].values]

brand_vectorizer = CountVectorizer()
brand_features = brand_vectorizer.fit_transform
(brands)

type_vectorizer = CountVectorizer()
type_features = type_vectorizer.fit_transform(t
ypes)

color_vectorizer = CountVectorizer()
color_features = color_vectorizer.fit_transform
(colors)

extra_features = hstack((brand_features, type_f
eatures, color_features)).tocsr()
```

In [0]:

```
def heat_map_w2v_brand(sentance1, sentance2, ur
l, doc_id1, doc_id2, df_id1, df_id2, model):

    # sentance1 : title1, input apparel
    # sentance2 : title2, recommended apparel
    # url: apparel image url
```

```
    # doc_id1: document id of input apparel
    # doc_id2: document id of recommended appar
el
    # df_id1: index of document1 in the data fr
ame
    # df_id2: index of document2 in the data fr
ame
    # model: it can have two values, 1. avg 2.
 weighted

    #s1_vec = np.array(#number_of_words_title1
 * 300), each row is a vector(weighted/avg) of
 length 300 corresponds to each word in give ti
tle
    s1_vec = get_word_vec(sentance1, doc_id1, m
odel)
    #s2_vec = np.array(#number_of_words_title2
 * 300), each row is a vector(weighted/avg) of
 length 300 corresponds to each word in give ti
tle
    s2_vec = get_word_vec(sentance2, doc_id2, m
odel)

    # s1_s2_dist = np.array(#number of words in
title1 * #number of words in title2)
    # s1_s2_dist[i,j] = euclidean distance betw
een words i, j
    s1_s2_dist = get_distance(s1_vec, s2_vec)

    data_matrix = [['Asin','Brand', 'Color', 'P
roduct type'],
                [data['asin'].loc[df_id1],brands
[doc_id1], colors[doc_id1], types[doc_id1]], #
 input apparel's features
                [data['asin'].loc[df_id2],brands
[doc_id2], colors[doc_id2], types[doc_id2]]] #
```

```
 recommonded apparel's features

    colorscale = [[0, '#1d004d'],[.5, '#f2e5ff'
],[1, '#f2e5d1']] # to color the headings of ea
ch column

    # we create a table with the data_matrix
    table = ff.create_table(data_matrix, index=
True, colorscale=colorscale)
    # plot it with plotly
    plotly.offline.iplot(table, filename='simpl
e_table')

    # devide whole figure space into 25 * 1:10
 grids
    gs = gridspec.GridSpec(25, 15)
    fig = plt.figure(figsize=(25,5))

    # in first 25*10 grids we plot heatmap
    ax1 = plt.subplot(gs[:, :-5])
    # ploting the heap map based on the pairwis
e distances
    ax1 = sns.heatmap(np.round(s1_s2_dist,6), a
nnot=True)
    # set the x axis labels as recommended appa
rels title
    ax1.set_xticklabels(sentance2.split())
    # set the y axis labels as input apparels t
itle
    ax1.set_yticklabels(sentance1.split())
    # set title as recommended apparels title
    ax1.set_title(sentance2)

    # in last 25 * 10:15 grids we display image
    ax2 = plt.subplot(gs[:, 10:16])
    # we dont display grid lins and axis labels
```

```
to images
    ax2.grid(False)
    ax2.set_xticks([])
    ax2.set_yticks([])

    # pass the url it display it
    display_img(url, ax2, fig)

    plt.show()
```

In [0]:

```
def idf_w2v_brand(doc_id, w1, w2, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand and color features

    # pairwise_dist will store the distance fro
m given input apparel to all remaining apparels
    # the metric we used here is cosine, the co
side distance is mesured as K(X, Y) = <X, Y> /
 (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/me
trics.html#cosine-similarity
    idf_w2v_dist  = pairwise_distances(w2v_titl
e_weight, w2v_title_weight[doc_id].reshape(1,-1
))
    ex_feat_dist = pairwise_distances(extra_fea
tures, extra_features[doc_id])
    pairwise_dist   = (w1 * idf_w2v_dist +  w2
* ex_feat_dist)/float(w1 + w2)

    # np.argsort will return indices of 9 small
est distances
    indices = np.argsort(pairwise_dist.flatten
())[0:num_results]
    #pdists will store the 9 smallest distances
```

```
    pdists  = np.sort(pairwise_dist.flatten())[
0:num_results]

    #data frame indices of the 9 smallest dista
ce's
    df_indices = list(data.index[indices])


    for i in range(0, len(indices)):
        heat_map_w2v_brand(data['title'].loc[df
_indices[0]],data['title'].loc[df_indices[i]],
data['medium_image_url'].loc[df_indices[i]], in
dices[0], indices[i],df_indices[0], df_indices[
i], 'weighted')
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print('Brand :',data['brand'].loc[df_in
dices[i]])
        print('euclidean distance from input :'
, pdists[i])
        print('='*125)

idf_w2v_brand(12566, 5, 5, 20)
# in the give heat map, each cell contains the
 euclidean distance between words i, j
```

ASIN : B00JXQB5FQ

Brand : Si Row

euclidean distance from input : 0.0019531 25

==========

ASIN : B00JXQCWTO

Brand : Si Row

euclidean distance from input : 2.3854711 5326

==========

ASIN : B00JXQASS6

Brand : Si Row

euclidean distance from input : 2.7390510 5609

==========

ASIN : B00JXQAFZ2

Brand : Si Row

euclidean distance from input : 3.3871871 95

==========

ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 3.55186
84389
==========
==========
==========
========

ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 3.5536174
776
==========
==========

==========

ASIN : B00JXQCUIC

Brand : Si Row

euclidean distance from input : 3.65382804889

==========

==========

==========

========

ASIN : B00JXQCFRS

Brand : Si Row

euclidean distance from input : 4.1288116

4569

==========

ASIN : B00JXQC8L6

Brand : Si Row

euclidean distance from input : 4.2039005 2813

==========

ASIN : B00JV63CW2

Brand : Si Row

euclidean distance from input : 4.28658676166

==========

ASIN : B015H41F6G

Brand : KINGDE

euclidean distance from input : 4.38937078798

==========

ASIN : B00JXQBBMI

Brand : Si Row

euclidean distance from input : 4.3979099 2755

==========

ASIN : B073R5Q8HD

Brand : Colosseum

euclidean distance from input : 4.4512285 8369

==========

ASIN : B074P8MD22
Brand : Edista
euclidean distance from input : 4.51897779787
==========

ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 4.52937545794
==========

==========
==========
==

ASIN : B00C0I3U3E
Brand : Stanzino
euclidean distance from input : 4.5303261
4076
==========
==========
==

ASIN : B01ER184O6
Brand : GuPoBoU168

euclidean distance from input : 4.5468170
2403
==========
==========
==========
==

ASIN : B073R4ZM7Y
Brand : Colosseum
euclidean distance from input : 4.5483555
4445
==========
==========
==========
==

ASIN : B071YF3WDD
Brand : Merona
euclidean distance from input : 4.6106274
2555
==========
==========
==========
==

ASIN : B01C6ORLDQ
Brand : 1 Mad Fit
euclidean distance from input : 4.6459178
9282
==========
==========
==========
==

In [0]:
```
# brand and color weight =50
```

```
# title vector weight = 5

idf_w2v_brand(12566, 5, 50, 20)
```

ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.0003551
13636364
==========
==========
==========
==

ASIN : B00JXQCWTO

```
Brand : Si Row
euclidean distance from input : 0.43372
2027865
=========================================
=========================================
=========================================
========
```

```
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 1.6550931
0669
=========================================
=========================================
=========================================
==
```

ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 1.7729360
4103
==========
==========
==========
==

ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 1.8028780
8538
==========
==========
==========
==

ASIN : B00JXQAO94

Brand : Si Row

euclidean distance from input : 1.8031960 9241

==========

ASIN : B00JXQCUIC

Brand : Si Row

euclidean distance from input : 1.8214161 9628

==========

==========
==

ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 1.9077768
5025
==========
==========
==

ASIN : B00JXQC8L6

```
Brand : Si Row
euclidean distance from input : 1.92142
937433
=========================================
=========================================
=========================================
========
```

```
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 1.9364632
3497
=========================================
=========================================
=========================================
==
```

ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 1.9567038
1059
==========
==========
==========
==

ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 1.9806066
343
==========
==========
==========
==

ASIN : B00JV63VC8
Brand : Si Row
euclidean distance from input : 2.0121855
9992
==========
==

ASIN : B00JXQAX2C
Brand : Si Row
euclidean distance from input : 2.0133517
8755
==========

==========
==

ASIN : B00JXQC0C8
Brand : Si Row
euclidean distance from input : 2.0138833
4827
==========
==========
==========
==

ASIN : B00JXQABB0
Brand : Si Row

euclidean distance from input : 2.0367257555

==========

ASIN : B01ER184O6

Brand : GuPoBoU168

euclidean distance from input : 2.65620416778

==========

ASIN : B01LZ7BQ4H

Brand : WAYF

euclidean distance from input : 2.6849067823

==========

ASIN : B01KJUM6JI

Brand : YABINA

euclidean distance from input : 2.68583819266

==========

```
ASIN : B01M06V4X1
Brand : WAYF
euclidean distance from input : 2.6947619
4865
==========================================
==========================================
==========================================
==
```

[10.2] Keras and Tensorflow to extract features

In [0]:

```
import numpy as np
from keras.preprocessing.image import ImageData
Generator
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dens
e
from keras import applications
from sklearn.metrics import pairwise_distances
import matplotlib.pyplot as plt
import requests
from PIL import Image
import pandas as pd
import pickle
```

```
Using TensorFlow backend.
```

```
Using TensorFlow backend.
```

In [0]:

```
# https://gist.github.com/fchollet/f35fbc80e066a49d65f1688a7e99f069
# Code reference: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html



# This code takes 40 minutes to run on a modern GPU (graphics card)
# like Nvidia  1050.
# GPU (NVidia 1050): 0.175 seconds per image

# This codse takes 160 minutes to run on a high end i7 CPU
# CPU (i7): 0.615 seconds per image.

#Do NOT run this code unless you want to wait a few hours for it to generate output

# each image is converted into 25088 length dense-vector


'''
# dimensions of our images.
img_width, img_height = 224, 224

top_model_weights_path = 'bottleneck_fc_model.h5'
train_data_dir = 'images2/'
nb_train_samples = 16042
epochs = 50
batch_size = 1
```

```
def save_bottlebeck_features():

    #Function to compute VGG-16 CNN for image f
eature extraction.

    asins = []
    datagen = ImageDataGenerator(rescale=1. / 2
55)

    # build the VGG16 network
    model = applications.VGG16(include_top=Fals
e, weights='imagenet')
    generator = datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode=None,
        shuffle=False)

    for i in generator.filenames:
        asins.append(i[2:-5])

    bottleneck_features_train = model.predict_g
enerator(generator, nb_train_samples // batch_s
ize)
    bottleneck_features_train = bottleneck_feat
ures_train.reshape((16042,25088))

    np.save(open('16k_data_cnn_features.npy',
 'wb'), bottleneck_features_train)
    np.save(open('16k_data_cnn_feature_asins.np
y', 'wb'), np.array(asins))
```

```
save_bottlebeck_features()

'''
```

[10.3] Visual features based product similarity.

In [0]:

```
#load the features and corresponding ASINS inf
o.
bottleneck_features_train = np.load('16k_data_c
nn_features.npy')
asins = np.load('16k_data_cnn_feature_asins.np
y')
asins = list(asins)

# load the original 16K dataset
data = pd.read_pickle('pickels/16k_apperal_data
_preprocessed')
df_asins = list(data['asin'])


from IPython.display import display, Image, SVG
, Math, YouTubeVideo


#get similar products using CNN features (VGG-1
6)
def get_similar_products_cnn(doc_id, num_result
s):
    doc_id = asins.index(df_asins[doc_id])
    pairwise_dist = pairwise_distances(bottlene
ck_features_train, bottleneck_features_train[do
c_id].reshape(1,-1))
```

```
    indices = np.argsort(pairwise_dist.flatten
())[0:num_results]
    pdists  = np.sort(pairwise_dist.flatten())[
0:num_results]

    for i in range(len(indices)):
        rows = data[['medium_image_url','title'
]].loc[data['asin']==asins[indices[i]]]
        for indx, row in rows.iterrows():
            display(Image(url=row['medium_image
_url'], embed=True))
            print('Product Title: ', row['titl
e'])
            print('Euclidean Distance from inpu
t image:', pdists[i])
            print('Amazon Url: www.amzon.com/d
p/'+ asins[indices[i]])

get_similar_products_cnn(12566, 20)
```

```
Product Title:  burnt umber tiger tshirt
zebra stripes xl  xxl
Euclidean Distance from input image: 0.0
Amazon Url: www.amzon.com/dp/B00JXQB5FQ
```

Product Title: pink tiger tshirt zebra s tripes xl xxl
Euclidean Distance from input image: 30.0501
Amazon Url: www.amzon.com/dp/B00JXQASS6

Product Title: yellow tiger tshirt tiger stripes l
Euclidean Distance from input image: 41.2611
Amazon Url: www.amzon.com/dp/B00JXQCUIC

Product Title: brown white tiger tshirt tiger stripes xl xxl
Euclidean Distance from input image: 44.0002
Amazon Url: www.amzon.com/dp/B00JXQCWTO

Product Title: kawaii pastel tops tees pink flower design
Euclidean Distance from input image: 47.3825
Amazon Url: www.amzon.com/dp/B071FCWD97

Product Title: womens thin style tops tees pastel watermelon print
Euclidean Distance from input image: 47.7184
Amazon Url: www.amzon.com/dp/B01JUNHBRM

Product Title: kawaii pastel tops tees b
aby blue flower design
Euclidean Distance from input image: 47.9
021
Amazon Url: www.amzon.com/dp/B071SBCY9W

Product Title: edv cheetah run purple
multi xl
Euclidean Distance from input image: 4
8.0465
Amazon Url: www.amzon.com/dp/B01CUPYBM0

Product Title: danskin womens vneck loos

e performance tee xsmall pink ombre
Euclidean Distance from input image: 48.1
019
Amazon Url: www.amzon.com/dp/B01E7BHXX8

Product Title: summer alpaca 3d pastel c
asual loose tops tee design
Euclidean Distance from input image: 48.1
189
Amazon Url: www.amzon.com/dp/B01I80A93G

Product Title: miss chievous juniors str
iped peplum tank top medium shadowpeach
Euclidean Distance from input image: 48.1
313
Amazon Url: www.amzon.com/dp/B0177DM70S

Product Title: red pink floral heel sleeveless shirt xl xxl
Euclidean Distance from input image: 48.1695
Amazon Url: www.amzon.com/dp/B00JV63QQE

Product Title: moana logo adults hot v neck shirt black xxl
Euclidean Distance from input image: 48.2568
Amazon Url: www.amzon.com/dp/B01LX6H43D

Product Title: abaday multicolor cartoon cat print short sleeve longline shirt large
Euclidean Distance from input image: 48.2657
Amazon Url: www.amzon.com/dp/B01CR57YY0

Product Title: kawaii cotton pastel tops tees peach pink cactus design
Euclidean Distance from input image: 48.3626
Amazon Url: www.amzon.com/dp/B071WYLBZS

Product Title: chicago chicago 18 shirt women pink
Euclidean Distance from input image: 48.3836
Amazon Url: www.amzon.com/dp/B01GXAZTRY

Product Title: yichun womens tiger print

ed summer tshirts tops
Euclidean Distance from input image: 48.4
493
Amazon Url: www.amzon.com/dp/B010NN9RXO

Product Title: nancy lopez whimsy short
sleeve whiteblacklemon drop xs
Euclidean Distance from input image: 48.4
788
Amazon Url: www.amzon.com/dp/B01MPX6IDX

Product Title: womens tops tees pastel p
each ice cream cone print
Euclidean Distance from input image: 48.5
58
Amazon Url: www.amzon.com/dp/B0734GRKZL

```
Product Title:  uswomens mary j blige wit
hout tshirts shirt
Euclidean Distance from input image: 48.6
144
Amazon Url: www.amzon.com/dp/B01M0XXFKK
```

Assignment

Loading Data

In [29]:

```
#load the features and corresponding ASINS inf
o.
bottleneck_features_train = np.load('Applied_AI
_Workshop_Code_Data/16k_data_cnn_features.npy')
#shape = (16042, 25088)
asins = np.load('Applied_AI_Workshop_Code_Data/
16k_data_cnn_feature_asins.npy')
asins = list(asins)
data = pd.read_pickle('Applied_AI_Workshop_Code
_Data/pickels/16k_apperal_data_preprocessed')
data.head()
```

Out[29]:

	asin	brand	color	medium_ima
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images- i amazon.com/im

	asin	brand	color	medium_ima
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-i amazon.com/im
15	B003BSRPB0	FeatherLite	White	https://images-i amazon.com/im
27	B014ICEJ1Q	FNC7C	Purple	https://images-i amazon.com/im
46	B01NACPBG2	Fifth Degree	Black	https://images-i amazon.com/im

TFIDF Vectorization

In [0]:
```
idf_title_vectorizer = CountVectorizer()
idf_title_features = idf_title_vectorizer.fit_t
ransform(data['title'])

# idf_title_features.shape = #data_points * #wo
rds_in_corpus
# CountVectorizer().fit_transform(courpus) retu
rns the a sparase matrix of dimensions #data_po
ints * #words_in_corpus
# idf_title_features[doc_id, index_of_word_in_c
orpus] = number of times the word occured in th
at doc
```

In [0]:
```
def n_containing(word):
```

```
    # return the number of documents which has
 the given word
    return sum(1 for blob in data['title'] if w
ord in blob.split())

def idf(word):
    # idf = log(#number of docs / #number of do
cs which had the given word)
    return math.log(data.shape[0] / (n_containi
ng(word)))
```

In [0]:

```
# we need to convert the values into float
idf_title_features  = idf_title_features.astype
(np.float)

for i in idf_title_vectorizer.vocabulary_.keys
():
    # for every word in whole corpus we will fi
nd its idf value
    idf_val = idf(i)

    # to calculate idf_title_features we need t
o replace the count values with the idf values
 of the word
    # idf_title_features[:, idf_title_vectorize
r.vocabulary_[i]].nonzero()[0] will return all
 documents in which the word i present
    for j in idf_title_features[:, idf_title_ve
ctorizer.vocabulary_[i]].nonzero()[0]:

        # we replace the count values of word i
in document j with  idf_value of word i
        # idf_title_features[doc_id, index_of_w
ord_in_courpus] = idf value of word
        idf_title_features[j,idf_title_vectoriz
```

```
er.vocabulary_[i]] = idf_val #idf_title_feature
s.shape = (16042, 12609)
```

IDF weighted Word2Vec for product similarity

In [0]:
```
import pickle
with open('Applied_AI_Workshop_Code_Data/word2v
ec_model', 'rb') as handle:
    model = pickle.load(handle)
```

In [0]:
```
# vocab = stores all the words that are there i
n google w2v model
# vocab = model.wv.vocab.keys() # if you are us
ing Google word2Vec

vocab = model.keys()
# this function will add the vectors of each wo
rd and returns the avg vector of given sentance
def build_avg_vec(sentence, num_features, doc_i
d, m_name):
    # sentace: its title of the apparel
    # num_features: the lenght of word2vec vect
or, its values = 300
    # m_name: model information it will take tw
o values
        # if  m_name == 'avg', we will append t
he model[i], w2v representation of word i
        # if m_name == 'weighted', we will mult
iply each w2v[word] with the idf(word)
```

```
    featureVec = np.zeros((num_features,), dtyp
e="float32")
    # we will intialize a vector of size 300 wi
th all zeros
    # we add each word2vec(wordi) to this fetur
eVec
    nwords = 0

    for word in sentence.split():
        nwords += 1
        if word in vocab:
            if m_name == 'weighted' and word in
idf_title_vectorizer.vocabulary_:
                featureVec = np.add(featureVec,
idf_title_features[doc_id, idf_title_vectorizer
.vocabulary_[word]] * model[word])
            elif m_name == 'avg':
                featureVec = np.add(featureVec,
model[word])
    if(nwords>0):
        featureVec = np.divide(featureVec, nwor
ds)
    # returns the avg vector of given sentance,
its of shape (1, 300)
    return featureVec
```

In [0]:

```
doc_id = 0
w2v_title_weight = []
# for every title we build a weighted vector re
presentation
for i in data['title']:
    w2v_title_weight.append(build_avg_vec(i, 30
0, doc_id,'weighted'))
    doc_id += 1
# w2v_title = np.array(# number of doc in courp
```

```
us * 300), each row corresponds to a doc
w2v_title_weight = np.array(w2v_title_weight)
```

OneHotEncoding the Color and Brand Features

weighted similarity using Color and Brand Feature

In [0]:

```
# some of the brand values are empty.
# Need to replace Null with string "NULL"
data['brand'].fillna(value="Not given", inplace
=True )

# replace spaces with hypen
brands = [x.replace(" ", "-") for x in data['br
and'].values]
colors = [x.replace(" ", "-") for x in data['co
lor'].values]

brand_vectorizer = CountVectorizer()
brand_features = brand_vectorizer.fit_transform
(brands)

color_vectorizer = CountVectorizer()
color_features = color_vectorizer.fit_transform
(colors)
```

In [0]:

```
from IPython.display import display, SVG, Math,
YouTubeVideo

#Display an image
```

```
def display_img(url,ax,fig):
    # we get the url of the apparel and downloa
d it
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    # we will display it in notebook
    plt.imshow(img)

# Utility functions

def get_word_vec(sentence, doc_id, m_name):
    # sentence : title of the apparel
    # doc_id: document id in our corpus
    # m_name: model information it will take tw
o values
        # if  m_name == 'avg', we will append t
he model[i], w2v representation of word i
        # if m_name == 'weighted', we will mult
iply each w2v[word] with the idf(word)
    vec = []
    for i in sentence.split():
        if i in vocab:
            if m_name == 'weighted' and i in  i
df_title_vectorizer.vocabulary_:
                vec.append(idf_title_features[d
oc_id, idf_title_vectorizer.vocabulary_[i]] * m
odel[i])
            elif m_name == 'avg':
                vec.append(model[i])
        else:
            # if the word in our courpus is not
there in the google word2vec corpus, we are jus
t ignoring it
            vec.append(np.zeros(shape=(300,)))
    # we will return a numpy array of shape (#n
umber of words in title * 300 ) 300 = len(w2v_m
```

```
odel[word])
    # each row represents the word2vec represen
tation of each word (weighted/avg) in given sen
tance
    return  np.array(vec)

def get_distance(vec1, vec2):
    # vec1 = np.array(#number_of_words_title1 *
300), each row is a vector of length 300 corres
ponds to each word in give title
    # vec2 = np.array(#number_of_words_title2 *
300), each row is a vector of length 300 corres
ponds to each word in give title

    final_dist = []
    # for each vector in vec1 we caluclate the
 distance(euclidean) to all vectors in vec2
    for i in vec1:
        dist = []
        for j in vec2:
            # np.linalg.norm(i-j) will result t
he euclidean distance between vectors i, j
            dist.append(np.linalg.norm(i-j))
        final_dist.append(np.array(dist))
    # final_dist = np.array(#number of words in
title1 * #number of words in title2)
    # final_dist[i,j] = euclidean distance betw
een vectors i, j
    return np.array(final_dist)

def create_table(df_id1, df_id2, doc_id1,doc_id
2):
    import plotly
    import plotly.figure_factory as ff
    from plotly.graph_objs import Scatter, Layo
ut
```

```
    plotly.offline.init_notebook_mode(connected
=False)

    data_matrix = [['Asin','Brand', 'Color'],
                   [data['asin'].loc[df_id1],br
ands[doc_id1], colors[doc_id1]], # input appare
l's features
                   [data['asin'].loc[df_id2],br
ands[doc_id2], colors[doc_id2]]] # recommonded
 apparel's features

    colorscale = [[0, '#1d004d'],[.5, '#f2e5ff'
],[1, '#f2e5d1']] # to color the headings of ea
ch column

    # we create a table with the data_matrix
    table = ff.create_table(data_matrix, index=
True, colorscale=colorscale)
    # plot it with plotly
    plotly.offline.iplot(table, filename='simpl
e_table')


def heat_map_idf_brand_col_img(sentance1, senta
nce2, url, doc_id1, doc_id2, df_id1, df_id2, mo
del):

    # sentance1 : title1, input apparel
    # sentance2 : title2, recommended apparel
    # url: apparel image url
    # doc_id1: document id of input apparel
    # doc_id2: document id of recommended appar
el
    # df_id1: index of document1 in the data fr
ame
    # df_id2: index of document2 in the data fr
```

```
ame
    # model: it can have two values, 1. avg 2.
 weighted

    #s1_vec = np.array(#number_of_words_title1
 * 300), each row is a vector(weighted/avg) of
 length 300 corresponds to each word in give ti
tle
    s1_vec = get_word_vec(sentance1, doc_id1, m
odel)
    #s2_vec = np.array(#number_of_words_title2
 * 300), each row is a vector(weighted/avg) of
 length 300 corresponds to each word in give ti
tle
    s2_vec = get_word_vec(sentance2, doc_id2, m
odel)

    # s1_s2_dist = np.array(#number of words in
title1 * #number of words in title2)
    # s1_s2_dist[i,j] = euclidean distance betw
een words i, j
    s1_s2_dist = get_distance(s1_vec, s2_vec)

    #Create a table
    #create_table(df_id1, df_id2, doc_id1,doc_i
d2)
    '''Extremely high ram consumption. Try to a
void running the above line'''


    # devide whole figure space into 25 * 1:10
 grids
    gs = gridspec.GridSpec(25, 15)
    fig = plt.figure(figsize=(25,5))

    # in first 25*10 grids we plot heatmap
```

```
    ax1 = plt.subplot(gs[:, :-5])
    # ploting the heap map based on the pairwis
e distances
    ax1 = sns.heatmap(np.round(s1_s2_dist,6), a
nnot=True)
    # set the x axis labels as recommended appa
rels title
    ax1.set_xticklabels(sentance2.split())
    # set the y axis labels as input apparels t
itle
    ax1.set_yticklabels(sentance1.split())
    # set title as recommended apparels title
    ax1.set_title(sentance2)

    # in last 25 * 10:15 grids we display image
    ax2 = plt.subplot(gs[:, 10:16])
    # we dont display grid lins and axis labels
to images
    ax2.grid(False)
    ax2.set_xticks([])
    ax2.set_yticks([])

    # pass the url it display it
    display_img(url, ax2, fig)

    plt.show()
```

In [0]:

```
def idf_w2v_brand_col_visual(doc_id, w1, w2, w3
, w4, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand features
    # w3: weight for color features
    # w4: weight for cnn features
```

```
    # pairwise_dist will store the distance fro
m given input apparel to all remaining apparels
    # the metric we used here is cosine, the co
side distance is mesured as K(X, Y) = <X, Y> /
 (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/me
trics.html#cosine-similarity
    # For Title
    idf_w2v_dist  = pairwise_distances(w2v_titl
e_weight, w2v_title_weight[doc_id].reshape(1,-1
))
    # For Brand
    brand_feat_dist = pairwise_distances(brand_
features, brand_features[doc_id].reshape(1,-1))
    # For Color
    col_feat_dist = pairwise_distances(color_fe
atures, color_features[doc_id].reshape(1,-1))
    # For Image
    cnn_feat_dist = pairwise_distances(bottlene
ck_features_train, bottleneck_features_train[do
c_id].reshape(1,-1))
    # Weighted Eucledean Distance
    pairwise_dist  = (w1*idf_w2v_dist +  w2*bra
nd_feat_dist + w3*col_feat_dist + w4*cnn_feat_d
ist)/float(w1 + w2 + w3 +w4)

    # np.argsort will return indices of 20 smal
lest distances
    indices = np.argsort(pairwise_dist.flatten
())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[
0:num_results]

    #data frame indices of the 9 smallest dista
ce's
```

```
    df_indices = list(data.index[indices])

    for i in range(0, len(indices)):
        heat_map_idf_brand_col_img(data['title'
].loc[df_indices[0]],data['title'].loc[df_indic
es[i]], data['medium_image_url'].loc[df_indices
[i]], indices[0], indices[i],df_indices[0], df_
indices[i], 'weighted')
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print('Brand :',data['brand'].loc[df_in
dices[i]])
        print('Color :',data['color'].loc[df_in
dices[i]])
        print('Product Type : ',data['product_t
ype_name'].loc[df_indices[i]])
        print('Euclidean distance from input :'
, pdists[i])

        print('='*125)
```

Giving preference to Title

In [39]:

```
idf_w2v_brand_col_visual(12566, 50, 5, 5, 5, 20
)
```

```
ASIN : B00JXQB5FQ
Brand : Si Row
Color : Brown
```

```
Product Type :  TOYS_AND_GAMES
Euclidean distance from input : 5.77633
7415087395e-07
========================================
========================================
========================================
========
```

```
ASIN : B00JXQASS6
Brand : Si Row
Color : Pink
Product Type :  TOYS_AND_GAMES
Euclidean distance from input : 6.9663763
19326176
========================================
========================================
========================================
==
```

```
ASIN : B00JXQCWTO
Brand : Si Row
Color : Brown
Product Type :  TOYS_AND_GAMES
Euclidean distance from input : 7.94286
59292367785
========================================
```

==========
==========
========

ASIN : B00JXQAFZ2
Brand : Si Row
Color : Grey
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 8.3801869
31931345
==========
==========
==========
==

ASIN : B00JXQAUWA
Brand : Si Row
Color : Yellow
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 8.5991026
65855623
==========
==========
==========
==

ASIN : B00JXQCUIC
Brand : Si Row
Color : Yellow
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 8.6558231
0605995
==========
==========
==========
==

ASIN : B00JXQAO94
Brand : Si Row
Color : White
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 8.7364411
57442162
==========
==========
==========
==

```
ASIN : B073R5Q8HD
Brand : Colosseum
Color : Yellow
Product Type :  SPORTING_GOODS
Euclidean distance from input : 8.86431
7652480512
=========================================
=========================================
=========================================
========
```

```
ASIN : B018H5AZXQ
Brand : Buffalo
Color : White Combo
Product Type :  SHIRT
Euclidean distance from input : 9.0183224
76254585
=========================================
=========================================
=========================================
==
```

```
ASIN : B072BVB47Z
Brand : H By Bordeaux
Color : Brown
Product Type :  SHIRT
```

Euclidean distance from input : 9.15569
082406851
==========
==========
==========
========

ASIN : B06XBHNM7J
Brand : Xhilaration
Color : Olive Tree
Product Type : SHIRT
Euclidean distance from input : 9.2282545
39430669
==========
==========
==========
==

ASIN : B01L7ROZNC
Brand : Bila
Color : Red
Product Type : SHIRT
Euclidean distance from input : 9.2464909
26594094
==========
==========

==========
==

ASIN : B01K0H02OG
Brand : Tultex
Color : Black
Product Type : SHIRT
Euclidean distance from input : 9.2539731
3249854
==========
==========
==========
==

ASIN : B073ZHRBV8
Brand : Exotic India
Color : Gray
Product Type : SHIRT
Euclidean distance from input : 9.2854528
57412114
==========
==========
==========
==

ASIN : B0722DJVQP
Brand : Kasper
Color : Blue
Product Type : SHIRT
Euclidean distance from input : 9.33012
3645193853
==========
==========
==========
========

ASIN : B01IPV1SFQ
Brand : Daniel Rainn
Color : White Ivory
Product Type : SHIRT
Euclidean distance from input : 9.3309317
74109866
==========
==========
==========
==

ASIN : B01DNNI1RO
Brand : Usstore
Color : as pictures
Product Type : SHIRT
Euclidean distance from input : 9.35065
0920515404
==========
==========
==========
========

ASIN : B06XTPC3FP
Brand : Kirkland Signature
Color : White
Product Type : SWEATER
Euclidean distance from input : 9.3520189
72571368
==========
==========
==========
==

ASIN : B072JTHCX6
Brand : Bobeau
Color : Orange
Product Type : SHIRT

```
Euclidean distance from input : 9.38337
5645268973
========================================
========================================
========================================
========
```

```
ASIN : B01C6ORLDQ
Brand : 1 Mad Fit
Color : Burgundy
Product Type :  SHIRT
Euclidean distance from input : 9.3847603
43139277
========================================
========================================
========================================
==
```

Giving preference to brand

In [0]:
```
idf_w2v_brand_col_visual(12566, 5, 50, 5, 5, 20
)
```

ASIN : B00JXQB5FQ
Brand : Si Row
Color : Brown
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 5.7763374
15087395e-07
==========
==========
==========
==

ASIN : B00JXQASS6
Brand : Si Row
Color : Pink
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 4.1529162
77399719
==========
==========
==========
==

ASIN : B00JXQCUIC
Brand : Si Row
Color : Yellow
Product Type : TOYS_AND_GAMES

Euclidean distance from input : 4.5757475
92660385
==========
==========
==========
==

ASIN : B00JXQCWTO
Brand : Si Row
Color : Brown
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 4.6399066
631610575
==========
==========
==========
==

ASIN : B00JXQAUWA
Brand : Si Row
Color : Yellow
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 4.66020
2319833237
==========
==========

==========
========

ASIN : B00JXQAFZ2
Brand : Si Row
Color : Grey
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 4.6693065
64358853
==========
==========
==========
==

ASIN : B00JXQAO94
Brand : Si Row
Color : White
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 4.7951191
82907906
==========
==========
==========
==

ASIN : B00JXQABB0
Brand : Si Row
Color : Red
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 4.9105036
22229571
==========
==========
==========
==

ASIN : B06XBHNM7J
Brand : Xhilaration
Color : Olive Tree
Product Type : SHIRT
Euclidean distance from input : 4.9362980
17030185
==========
==========
==========
==

```
ASIN : B00JV63QQE
Brand : Si Row
Color : Red
Product Type :  TOYS_AND_GAMES
Euclidean distance from input : 4.96458
52896161395
========================================
========================================
========================================
========
```

```
ASIN : B00JXQBBMI
Brand : Si Row
Color : Yellow
Product Type :  TOYS_AND_GAMES
Euclidean distance from input : 4.9941998
40866891
========================================
========================================
========================================
==
```

```
ASIN : B00JXQC8L6
Brand : Si Row
Color : Blue
```

Product Type : TOYS_AND_GAMES
Euclidean distance from input : 5.06826
92307970285
==========
==========
==========
========

ASIN : B00JXQC0C8
Brand : Si Row
Color : Blue
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 5.0749217
10702378
==========
==========
==========
==

ASIN : B018H5AZXQ
Brand : Buffalo
Color : White Combo
Product Type : SHIRT
Euclidean distance from input : 5.0796799
28087877
==========

==========
==========
==

ASIN : B00JXQCFRS
Brand : Si Row
Color : Grey
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 5.1523404
04831734
==========
==========
==========
==

ASIN : B01L7ROZNC
Brand : Bila
Color : Red
Product Type : SHIRT
Euclidean distance from input : 5.1825640
367619705
==========
==========
==========
==

ASIN : B00JV63CW2

Brand : Si Row

Color : Red

Product Type : TOYS_AND_GAMES

Euclidean distance from input : 5.2101790795237415

==========

ASIN : B00JXQAX2C

Brand : Si Row

Color : Pink

Product Type : TOYS_AND_GAMES

Euclidean distance from input : 5.2688980102817045

==========

```
ASIN : B01FJVZST2
Brand : KONGYII
Color : White
Product Type :  SHIRT
Euclidean distance from input : 5.2880026
79997397
==========================================
==========================================
==========================================
==
```

```
ASIN : B00JV63VC8
Brand : Si Row
Color : Purple
Product Type :  TOYS_AND_GAMES
Euclidean distance from input : 5.2897087
68432245
==========================================
==========================================
==========================================
==
```

Giving preference to color

In [40]:
```
idf_w2v_brand_col_visual(12566, 5, 5, 50, 5, 20
)
```

ASIN : B00JXQB5FQ
Brand : Si Row
Color : Brown
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 5.7763374
15087395e-07
==========
==========
==========
==

ASIN : B072BVB47Z
Brand : H By Bordeaux
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.2174259
47922926
==========
==========
==========
==

ASIN : B074MJN1K9
Brand : Hip
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.2751901
265362475
==========
==========
==========
==

ASIN : B014OUHUZY
Brand : Black Temptation
Color : Brown
Product Type : BLAZER
Euclidean distance from input : 4.4883685
77223557
==========
==========
==========
==

ASIN : B074J7BCYM

Brand : Nanon
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.60197
6930735466
==========
==========
==========
========

ASIN : B00BTJKAQ0
Brand : Anna-Kaci
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.6206304
69689003
==========
==========
==========
==

ASIN : B071LDTQ1F
Brand : Hip
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.6260825
89633453

==========
==========
==========
==

ASIN : B00JXQCWTO
Brand : Si Row
Color : Brown
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 4.6399066
631610575
==========
==========
==========
==

ASIN : B074QVMXSQ
Brand : bellatrix
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.6408836
73711728
==========
==========
==========
==

ASIN : B073ZCN5LG
Brand : Brunello Cucinelli
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.6476215
06911058
==========
==========
==========
==

ASIN : B01KJUM6JI
Brand : YABINA
Color : Brown
Product Type : BOOKS_1973_AND_LATER
Euclidean distance from input : 4.6553735
95868429
==========
==========
==========
==

ASIN : B003SPYNAM
Brand : Marrikas
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.6661801637207425
==========

ASIN : B017YBAI9A
Brand : La Fleva
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.6783087 51032903
==========

ASIN : B072M4ZF89
Brand : Abound
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.69123
9262184535
==========
==========
==========
========

ASIN : B074P8YWV4
Brand : Bobeau
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.7067877
921762715
==========
==========
==========
==

ASIN : B01CE40W16
Brand : Marolaya
Color : Brown

Product Type : SHIRT
Euclidean distance from input : 4.72528
7430660346
==========
==========
==========
========

ASIN : B016MGC5VW
Brand : VOGUE CODE
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.7412705
64152644
==========
==========
==========
==

ASIN : B0758356K3
Brand : Soprano
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.7447773
64554577
==========

==========
==========
==

ASIN : B07288KFHF
Brand : Soprano
Color : Brown
Product Type : SHIRT
Euclidean distance from input : 4.7495443
86320652
==========
==========
==========
==

ASIN : B00MJPVIDW
Brand : Xclusive Collection
Color : Brown
Product Type : DRESS
Euclidean distance from input : 4.7497288
63055889
==========
==========

==========
==

Giving preference to image feature

In [0]:
```
idf_w2v_brand_col_visual(12566, 5, 5, 5, 50, 20
)
```

ASIN : B00JXQB5FQ
Brand : Si Row
Color : Brown
Product Type : TOYS_AND_GAMES
Euclidean distance from input : 5.7763375
27025205e-06
==========
==========
==========
==

ASIN : B06XBHNM7J
Brand : Xhilaration
Color : Olive Tree
Product Type : SHIRT
Euclidean distance from input : 29.482565

79290498
==========
==========
==========
==

ASIN : B016CU40IY
Brand : Juiceclouds
Color : Black
Product Type : SHIRT
Euclidean distance from input : 30.649793
04592252
==========
==========
==========
==

ASIN : B074MXY984
Brand : We The Free
Color : Black
Product Type : SHIRT
Euclidean distance from input : 31.2094
46306665402
==========
==========

==========
========

ASIN : B074LTBWSW
Brand : Liz Lange
Color : Turquoise
Product Type : SHIRT
Euclidean distance from input : 31.210358
663733476
==========
==========
==========
==

ASIN : B018H5AZXQ
Brand : Buffalo
Color : White Combo
Product Type : SHIRT
Euclidean distance from input : 31.269700
286219283
==========
==========
==========
==

ASIN : B06XYP1X1F
Brand : J Brand Jeans
Color : Navy/Blk Stp
Product Type : SHIRT
Euclidean distance from input : 31.386133
5167518
==========
==========
==========
==

ASIN : B01BMSFYW2
Brand : igertommy hilf
Color : white cloud dancer
Product Type : SHIRT
Euclidean distance from input : 31.560316
995474007
==========
==========
==========
==

ASIN : B01FJVZST2
Brand : KONGYII
Color : White
Product Type : SHIRT
Euclidean distance from input : 32.2848
9418821894
==========
==========
==========
========

ASIN : B01L7ROZNC
Brand : Bila
Color : Red
Product Type : SHIRT
Euclidean distance from input : 32.393298
78792222
==========
==========
==========
==

ASIN : B01L9F153U
Brand : ATYPEMX
Color : Black

Product Type : SHIRT
Euclidean distance from input : 32.9123
7438700575
==========
==========
==========
========

ASIN : B01EXXFS4M
Brand : No Boundaries
Color : Pink
Product Type : SHIRT
Euclidean distance from input : 32.960838
55264949
==========
==========
==========
==

ASIN : B0756JTS1F
Brand : Salvatore Ferragamo
Color : Multi-color
Product Type : SWEATER
Euclidean distance from input : 33.1241

54817698354
==========
==========
==========
========

ASIN : B06Y41MRCH
Brand : Byoung
Color : Chambray Blue
Product Type : SHIRT
Euclidean distance from input : 33.156467
33615543
==========
==========
==========
==

ASIN : B074MK6LV2
Brand : 1.State
Color : Blue
Product Type : SHIRT
Euclidean distance from input : 33.210313
27672931
==========
==========

==========
==

ASIN : B074Z5C98D
Brand : Ariella's closet
Color : Multi Color Black & Pink
Product Type : APPAREL
Euclidean distance from input : 33.347471
61979491
==========
==========
==========
==

ASIN : B00DP4VHWI
Brand : Stanzino
Color : Fuchsia
Product Type : SHIRT
Euclidean distance from input : 33.434263
41335416
==========
==========

==========

ASIN : B01M8GB3AL
Brand : Maven West
Color : Yellow
Product Type : SHIRT
Euclidean distance from input : 33.4714
89715603965
==========
==========
==========
========

ASIN : B00JMAASRO
Brand : Wotefusi
Color : Multicolor
Product Type : DRESS
Euclidean distance from input : 33.482222
55031705
==========
==========
==========
==