Amazon Apparel Recommendations

1.0 Overview of the data

In [1]:

```
#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 CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
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=True)
warnings.filterwarnings("ignore")
```

In [2]:

```
# we have give a json file which consists of all information about
# the products
# loading the data using pandas' read_json file.
data = pd.read_json('tops_fashion.json')
```

In [3]:

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

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

Terminology:

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

In [4]:

```
# each product/item has 19 features in the raw dataset.
data.columns # prints column-names or feature-names.
```

Out[4]:

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

Of these 19 features, we will be using only 7 features in this workshop.

1. asin (Amazon standard identification number)
2. brand (brand to which the product belongs 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 apperal, ex: SHIRT/TSHIRT)
5. medium_image_url (url of the image)
6. title (title of the product.)
7. formatted_price (price of the product)

In [5]:

```
data = data[['asin', 'brand', 'color', 'medium_image_url', 'product_type_name', 'title'
, 'formatted_price']]
```

In [6]:

```
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[6]:

	asin	brand	color	medium_image_url	product_type_name	
0	B016I2TS4W	FNC7C	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Minior Como Super Ironm Long R...
1	B01N49AI08	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG C Wome Tunic
2	B01JDPCOHO	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG C Wome Won
3	B01N19U5H5	Focal18	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Focal Sailor Bubbl Sleev Blous
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	Feath Ladie Sleev Resis

2.0 Missing data for various features.

Basic stats for the feature: product_type_name

In [0]:

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

```
count     183138
unique        72
top        SHIRT
freq      167794
Name: product_type_name, dtype: object
```

- We have total 72 unique type of product_type_names
- 91.62% (167794/183138) of the products are shirts,

In [0]:

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

```
['SHIRT' 'SWEATER' 'APPAREL' 'OUTDOOR_RECREATION_PRODUCT'
 'BOOKS_1973_AND_LATER' 'PANTS' 'HAT' 'SPORTING_GOODS' 'DRESS' 'UNDERWEAR'
 'SKIRT' 'OUTERWEAR' 'BRA' 'ACCESSORY' 'ART_SUPPLIES' 'SLEEPWEAR'
 'ORCA_SHIRT' 'HANDBAG' 'PET_SUPPLIES' 'SHOES' 'KITCHEN' 'ADULT_COSTUME'
 'HOME_BED_AND_BATH' 'MISC_OTHER' 'BLAZER' 'HEALTH_PERSONAL_CARE'
 'TOYS_AND_GAMES' 'SWIMWEAR' 'CONSUMER_ELECTRONICS' 'SHORTS' 'HOME'
 'AUTO_PART' 'OFFICE_PRODUCTS' 'ETHNIC_WEAR' 'BEAUTY'
 'INSTRUMENT_PARTS_AND_ACCESSORIES' 'POWERSPORTS_PROTECTIVE_GEAR' 'SHIRTS'
 'ABIS_APPAREL' 'AUTO_ACCESSORY' 'NONAPPARELMISC' 'TOOLS' 'BABY_PRODUCT'
 'SOCKSHOSIERY' 'POWERSPORTS_RIDING_SHIRT' 'EYEWEAR' 'SUIT'
 'OUTDOOR_LIVING' 'POWERSPORTS_RIDING_JACKET' 'HARDWARE' 'SAFETY_SUPPLY'
 'ABIS_DVD' 'VIDEO_DVD' 'GOLF_CLUB' 'MUSIC_POPULAR_VINYL'
 'HOME_FURNITURE_AND_DECOR' 'TABLET_COMPUTER' 'GUILD_ACCESSORIES'
 'ABIS_SPORTS' 'ART_AND_CRAFT_SUPPLY' 'BAG' 'MECHANICAL_COMPONENTS'
 'SOUND_AND_RECORDING_EQUIPMENT' 'COMPUTER_COMPONENT' 'JEWELRY'
 'BUILDING_MATERIAL' 'LUGGAGE' 'BABY_COSTUME' 'POWERSPORTS_VEHICLE_PART'
 'PROFESSIONAL_HEALTHCARE' 'SEEDS_AND_PLANTS' 'WIRELESS_ACCESSORY']
```

In [7]:

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

Out[7]:

```
[('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 [14]:

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

```
count     182987
unique     10577
top         Zago
freq         223
Name: brand, dtype: object
```

- There are 10577 unique brands
- There are 151 (183138 - 182987) missing values.

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())
```

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

- We have 7380 unique colors
- 7.2% of products are black in color
- 64956 of 183138 products have brand information. That's approximately 35.4%.

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())
```

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

- Only 28,395 (15.5% of whole data) have products with price information

In [0]:

```
price_count = Counter(list(data['formatted_price']))
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())
```

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

- All of the products have a title.
- Titles are fairly descriptive of what the product is.
- We use titles extensively in this workshop.
- As they are short and informative.

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 information
# data['formatted_price'].isnull() => gives the information
#about the dataframe row's which have null values 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 price=NULL : 28395
```

In [0]:

```
# consider products which have color information
# data['color'].isnull() => gives the information about the dataframe row's which have
 null values 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 color=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 this 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.content))
        img.save('images/28k_images/'+row['asin']+'.jpeg')


'''
```

Out[0]:

```
"\nfrom PIL import Image\nimport requests\nfrom io import BytesIO\n\nfor i
ndex, row in images.iterrows():\n        url = row['large_image_url']\n
response = requests.get(url)\n        img = Image.open(BytesIO(response.co
ntent))\n        img.save('workshop/images/28k_images/'+row['asin']+'.jpe
g')\n\n\n"
```

3.0 Remove near duplicate items

3.1 Understand about duplicates.

In [0]:

```
# read data from pickle file from previous stage
data = pd.read_pickle('pickels/28k_apparel_data')

# find number of products that have duplicate titles.
print(sum(data.duplicated('title')))
```

```
2325
```

- We have 2325 products which have same title but different color

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.

3.2 Remove duplicates : Part 1

In [0]:

```
# read data from pickle file from previous stage
data = pd.read_pickle('pickels/28k_apparel_data')
```

In [0]:

```
data.head()
```

Out[0]:

	asin	brand	color	medium_image_url	product_type_name	
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	Featl Ladie Slee Resi
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	Wom Uniq 100% T - S Olym
11	B001LOUGE4	Fitness Etc.	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	Ladie Cotto 2x1 F Tank
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	Featl Ladie Mois Free Spor
21	B014ICEDNA	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	Supe Chib Dear Cast Shor

In [0]:

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

```
After removal of products with short description: 27949
```

In [0]:

```
# Sort the whole data based on title (alphabetical order of title)
data_sorted.sort_values('title',inplace=True, ascending=False)
data_sorted.head()
```

Out[0]:

	asin	brand	color	medium_image_url	product_type_name
61973	B06Y1KZ2WB	Éclair	Black/Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT
133820	B010RV33VE	xiaoming	Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT
81461	B01DDSDLNS	xiaoming	White	https://images-na.ssl-images-amazon.com/images...	SHIRT
75995	B00X5LYO9Y	xiaoming	Red Anchors	https://images-na.ssl-images-amazon.com/images...	SHIRT
151570	B00WPJG35K	xiaoming	White	https://images-na.ssl-images-amazon.com/images...	SHIRT

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

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

Title 2:
25. tokidoki The Queen of Diamonds Women's Shirt X-Large
26. tokidoki The Queen of Diamonds Women's Shirt Small
27. tokidoki The Queen of Diamonds Women'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-shirt
62. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Pri
nt Head Shirt for woman Neon Wolf t-shirt
63. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Pri
nt Head Shirt for woman Neon Wolf t-shirt
64. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Pri
nt Head Shirt for woman Neon Wolf t-shirt
```

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_points:

    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 sequentially
    j = i+1
    while j < num_data_points:

        # store the list of words of jth string in b, ex: b = ['tokidoki', 'The', 'Quee
n', 'of', 'Diamonds', 'Women's', 'Shirt', 'Small']
        b = data['title'].loc[indices[j]].split()

        # store the maximum length of two strings
        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 will 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 consider
ing it as those two apperals are different
        # if the number of words in which both strings differ are < 2 , we are consider
ing it as those two apperals are same, hence we are ignoring them
        if (length - count) > 2: # number of words in which both sensences differ
            # if both strings are differ by more than 2 words we include the 1st string
index
            stage1_dedupe_asins.append(data_sorted['asin'].loc[indices[i]])

            # if the comaprision between is between num_data_points, num_data_points-1
 strings and they differ in more than 2 words we include both
            if j == num_data_points-1: stage1_dedupe_asins.append(data_sorted['asin'].l
oc[indices[j]])

            # start searching for similar apperals 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')
```

3.3 Remove duplicates : Part 2

```
In the previous cell, we sorted whole data 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 Wrinkle-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_data')
```

In [0]:

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

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', 'Quee
n', 'of', '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 will 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 consider
ing it as those two apperals are same, hence we are ignoring 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 the Google Drive folder.
```

4.0 Text pre-processing

In [0]:

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

# 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 downloaded 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 review 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', 'hers', 'up', 'she', 'd', 'further',
'all', 'than', 'under', 'is', 'off', 'both', 'most', 'few', 'should', 'r
e', 'very', 'just', 'then', 'didn', 'myself', 'in', 'too', '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', 'a
gainst', 'above', 'down', 'they', 'below', 'me', 'again', 'for', 'why', 'b
een', 'yourselves', 'more', 'her', 'that', 'can', 'am', 'was', 'themselve
s', 'mightn', 'does', 'those', 'only', 'hasn', 'any', 'ma', 'are', 'nor',
'out', 'you', 'ourselves', 'the', 'an', 'has', 'where', 'i', 'while', 'our
s', 'its', 'your', 'had', 'were', 'being', 'no', 'or', 'needn', 've', 'y',
'a', 'each', 'have', 'through', 'when', 'mustn', 'by', 'won', 'from', 'ow
n', 'will', 'there', 't', 'him', 'these', 'doesn', 'theirs', 'my', 'did',
'of', 'who', 'until', 'wouldn', 'we', 'do', 'having', 'yours', 'other', 'w
asn', 'it', 'with', 'once', 'here', 'don', 'o', 'whom', 'this', 'if', 'bu
t', '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, 'title')
# 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_image_url	product_type_name	
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	feath ladie slee resis
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	wom uniq cotto spec olym wor..
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	feath ladie mois free spor
27	B014ICEJ1Q	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	supe chibi dear neck
46	B01NACPBG2	Fifth Degree	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	fifth wom gold grap jun..

In [0]:

```
data.to_pickle('pickels/16k_apperal_data_preprocessed')
```

5.0 Text based product similarity

In [0]:

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

Out[0]:

	asin	brand	color	medium_image_url	product_type_name	
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	feath ladie slee resis
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	wom uniq cotto spec olym wor..
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	feath ladie mois free spor
27	B014ICEJ1Q	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	supe chibi dear neck
46	B01NACPBG2	Fifth Degree	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	fifth wom gold grap jun..

In [0]:

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


#Display an image
def display_img(url,ax,fig):
    # we get the url of the apparel and download 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 decision.
def plot_heatmap(keys, values, labels, url, text):
        # keys: list of words of recommended title
        # values: len(values) ==  len(keys), values(i) represents the occurence of the
 word keys(i)
        # labels: len(labels) == len(keys), the values of labels depends on the model w
e are using
                # if model == 'bag of words': labels(i) = values(i)
                # if model == 'tfidf weighted bag 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_ratios=[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
title1 and list of words of title2), in black if not
        ax = sns.heatmap(np.array([values]), annot=np.array([labels]))
        ax.set_xticklabels(keys) # set that axis labels as the words of title
        ax.set_title(text) # apparel title

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

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

        # displays combine figure ( heat map and 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 to 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 t
he distance between two title vec's
    intersection = set(vec1.keys()) & set(vec2.keys())

    # we set the values of non intersecting words to zero, this is just to show the dif
ference 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 values of labels depends on the model we ar
e using
        # if model == 'bag of words': labels(i) = values(i)
        # if model == 'tfidf weighted bag of words':labels(i) = tfidf(keys(i))
        # if model == 'idf weighted bag of words':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 corpu
s
            # tfidf_title_features[doc_id, index_of_word_in_corpus] will give the tfidf
value of word in given document (doc_id)
            if x in  tfidf_title_vectorizer.vocabulary_:
                labels.append(tfidf_title_features[doc_id, tfidf_title_vectorizer.vocab
ulary_[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 val
ue of word in given document (doc_id)
            if x in  idf_title_vectorizer.vocabulary_:
                labels.append(idf_title_features[doc_id, idf_title_vectorizer.vocabular
y_[x]])
            else:
                labels.append(0)

    plot_heatmap(keys, values, labels, url, text)


# 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 string, 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 retur
```

```
ns dict type object {word1:count}



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

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

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

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

5.1 Bag of Words (BoW) on product titles.

In [0]:

```
from sklearn.feature_extraction.text import CountVectorizer
title_vectorizer = CountVectorizer()
title_features   = title_vectorizer.fit_transform(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) returns
# the a sparase matrix of dimensions #data_points * #words_in_corpus

# What is a sparse vector?

# title_features[doc_id, index_of_word_in_corpus] = number of times the word occured in
that doc
```

Out[0]:

```
(16042, 12609)
```

In [ ]:

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

    # pairwise_dist will store the distance from given input apparel to all remaining a
pparels
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <
X, Y> / (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = pairwise_distances(title_features,title_features[doc_id])

    # np.argsort will return indices of the smallest 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 distace'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_in
dices[i]], data['medium_image_url'].loc[df_indices[i]], 'bag_of_words')
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print ('Brand:', data['brand'].loc[df_indices[i]])
        print ('Title:', data['title'].loc[df_indices[i]])
        print ('Euclidean similarity with the query image :', pdists[i])
        print('='*60)

#call the bag-of-words model for a product to get similar products.
bag_of_words_model(12566, 20) # change the index if you want to.
# In the output heat map each value represents the count value
# of the label word, the color represents the intersection
# with inputs title.

#try 12566
#try 931
```

5.2 TF-IDF based product similarity

In [0]:

```
tfidf_title_vectorizer = TfidfVectorizer(min_df = 0)
tfidf_title_features = tfidf_title_vectorizer.fit_transform(data['title'])
# tfidf_title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(courpus) returns the a sparase matrix of dimensions #
data_points * #words_in_corpus
# tfidf_title_features[doc_id, index_of_word_in_corpus] = tfidf values of the word in g
iven doc
```

In [ ]:

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

    # pairwise_dist will store the distance from given input apparel to all remaining a
pparels
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <
X, Y> / (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = pairwise_distances(tfidf_title_features,tfidf_title_features[doc_id
])

    # np.argsort will return indices of 9 smallest 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 distace'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_i
ndices[i]], data['medium_image_url'].loc[df_indices[i]], 'tfidf')
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print('BRAND :',data['brand'].loc[df_indices[i]])
        print ('Eucliden distance from the given 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
```

5.3 IDF based product similarity

In [0]:

```
idf_title_vectorizer = CountVectorizer()
idf_title_features = idf_title_vectorizer.fit_transform(data['title'])

# idf_title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(courpus) returns the a sparase matrix of dimensions #
data_points * #words_in_corpus
# idf_title_features[doc_id, index_of_word_in_corpus] = number of times the word occure
d in that 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 word in blob.split())

def idf(word):
    # idf = log(#number of docs / #number of docs which had the given word)
    return math.log(data.shape[0] / (n_containing(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 find its idf value
    idf_val = idf(i)

    # to calculate idf_title_features we need to replace the count values with the idf
 values of the word
    # idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0] will retu
rn all documents in which the word i present
    for j in idf_title_features[:, idf_title_vectorizer.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_word_in_courpus] = idf value of word
        idf_title_features[j,idf_title_vectorizer.vocabulary_[i]] = idf_val
```

In [ ]:

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

    # pairwise_dist will store the distance from given input apparel to all remaining a
pparels
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <
X, Y> / (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = pairwise_distances(idf_title_features,idf_title_features[doc_id])

    # np.argsort will return indices of 9 smallest 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 distace'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_in
dices[i]], data['medium_image_url'].loc[df_indices[i]], 'idf')
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print('Brand :',data['brand'].loc[df_indices[i]])
        print ('euclidean distance from the given 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 co
lor represents the intersection with inputs title
```

6.0 Text Semantics based product similarity

In [0]:

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

# in this project we are using a pretrained model by google
# its 3.3G file, once you load this into your memory
# 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 "GoogleNews-vectors-negative300.bin"
# from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
# it's 1.9GB in size.

'''
model = KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=
True)
'''

#if you do NOT have RAM >= 12GB, use the code below.
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 two values
        # if  m_name == 'avg', we will append the model[i], w2v representation of word
 i
        # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)
    vec = []
    for i in sentence.split():
        if i in vocab:
            if m_name == 'weighted' and i in  idf_title_vectorizer.vocabulary_:
                vec.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[
i]] * model[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 just ignoring it
            vec.append(np.zeros(shape=(300,)))
    # we will return a numpy array of shape (#number of words in title * 300 ) 300 = le
n(w2v_model[word])
    # each row represents the word2vec representation of each word (weighted/avg) in gi
ven sentance
    return  np.array(vec)

def get_distance(vec1, vec2):
    # vec1 = np.array(#number_of_words_title1 * 300), each row is a vector of length 30
0 corresponds to each word in give title
    # vec2 = np.array(#number_of_words_title2 * 300), each row is a vector of length 30
0 corresponds to each word in give title

    final_dist = []
    # for each vector in vec1 we caluclate the distance(euclidean) to all vectors in ve
c2
    for i in vec1:
        dist = []
        for j in vec2:
            # np.linalg.norm(i-j) will result the 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 between 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 apparel
    # 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/av
g) of length 300 corresponds to each word in give title
    s1_vec = get_word_vec(sentence1, doc_id1, model)
```

```
    #s2_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/av
g) of length 300 corresponds to each word in give title
    s2_vec = get_word_vec(sentence2, doc_id2, model)

    # s1_s2_dist = np.array(#number of words in title1 * #number of words in title2)
    # s1_s2_dist[i,j] = euclidean distance between 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 imag
e of apparel
    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 pairwise distances
    ax = sns.heatmap(np.round(s1_s2_dist,4), annot=True)
    # set the x axis labels as recommended apparels title
    ax.set_xticklabels(sentence2.split())
    # set the y axis labels as input apparels title
    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 image
    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 in google w2v model
# vocab = model.wv.vocab.keys() # if you are using Google word2Vec

vocab = model.keys()
# this function will add the vectors of each word and returns the avg vector of given s
entance
def build_avg_vec(sentence, num_features, doc_id, m_name):
    # sentace: its title of the apparel
    # num_features: the lenght of word2vec vector, its values = 300
    # m_name: model information it will take two values
        # if  m_name == 'avg', we will append the model[i], w2v representation of word
 i
        # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)

    featureVec = np.zeros((num_features,), dtype="float32")
    # we will intialize a vector of size 300 with all zeros
    # we add each word2vec(wordi) to this fetureVec
    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_ve
ctorizer.vocabulary_[word]] * model[word])
            elif m_name == 'avg':
                featureVec = np.add(featureVec, model[word])
    if(nwords>0):
        featureVec = np.divide(featureVec, nwords)
    # returns the avg vector of given sentance, its of shape (1, 300)
    return featureVec
```

6.1 Average Word2Vec product similarity.

In [0]:

```
doc_id = 0
w2v_title = []
# for every title we build a avg vector representation
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 courpus * 300), each row corresponds to a doc
w2v_title = np.array(w2v_title)
```

In [ ]:

```
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_title, w2v_title[doc_id].reshape(1,-1))

    # np.argsort will return indices of 9 smallest 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 distace's
    df_indices = list(data.index[indices])

    for i in range(0, len(indices)):
        heat_map_w2v(data['title'].loc[df_indices[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_indices[i]])
        print('BRAND :',data['brand'].loc[df_indices[i]])
        print ('euclidean distance from given input 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
```

6.2 IDF weighted Word2Vec for product similarity

In [0]:

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

In [ ]:

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

    # pairwise_dist will store the distance from given input apparel to all remaining a
pparels
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <
X, Y> / (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].resha
pe(1,-1))

    # np.argsort will return indices of 9 smallest 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 distace's
    df_indices = list(data.index[indices])

    for i in range(0, len(indices)):
        heat_map_w2v(data['title'].loc[df_indices[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_indices[i]])
        print('Brand :',data['brand'].loc[df_indices[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
```

6.3 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['brand'].values]
types = [x.replace(" ", "-") for x in data['product_type_name'].values]
colors = [x.replace(" ", "-") for x in data['color'].values]

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

type_vectorizer = CountVectorizer()
type_features = type_vectorizer.fit_transform(types)

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

extra_features = hstack((brand_features, type_features, color_features)).tocsr()
```

In [0]:

```
def heat_map_w2v_brand(sentance1, sentance2, url, doc_id1, doc_id2, df_id1, df_id2, mod
el):

    # 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 apparel
    # df_id1: index of document1 in the data frame
    # df_id2: index of document2 in the data frame
    # 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/av
g) of length 300 corresponds to each word in give title
    s1_vec = get_word_vec(sentance1, doc_id1, model)
    #s2_vec = np.array(#number_of_words_title2 * 300), each row is a vector(weighted/av
g) of length 300 corresponds to each word in give title
    s2_vec = get_word_vec(sentance2, doc_id2, model)

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

    data_matrix = [['Asin','Brand', 'Color', 'Product type'],
                [data['asin'].loc[df_id1],brands[doc_id1], colors[doc_id1], types[doc_id
1]], # input apparel's features
                [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2], types[doc_id
2]]] # recommonded apparel's features

    colorscale = [[0, '#1d004d'],[.5, '#f2e5ff'],[1, '#f2e5d1']] # to color the heading
s of each 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='simple_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 pairwise distances
    ax1 = sns.heatmap(np.round(s1_s2_dist,6), annot=True)
    # set the x axis labels as recommended apparels title
    ax1.set_xticklabels(sentance2.split())
    # set the y axis labels as input apparels title
    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 [ ]:

```
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 from given input apparel to all remaining a
pparels
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <
X, Y> / (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].resha
pe(1,-1))
    ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id])
    pairwise_dist   = (w1 * idf_w2v_dist +  w2 * ex_feat_dist)/float(w1 + w2)

    # np.argsort will return indices of 9 smallest 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 distace'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_indice
s[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_indices[i]])
        print('Brand :',data['brand'].loc[df_indices[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
```

In [ ]:

```
# brand and color weight =50
# title vector weight = 5

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

7.0 Keras and Tensorflow to extract features

In [ ]:

```
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
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
```

In [0]:

```
# https://gist.github.com/fchollet/f35fbc80e066a49d65f1688a7e99f069
# Code reference: https://blog.keras.io/building-powerful-image-classification-models-u
sing-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 feature extraction.

    asins = []
    datagen = ImageDataGenerator(rescale=1. / 255)

    # build the VGG16 network
    model = applications.VGG16(include_top=False, 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_generator(generator, nb_train_samples //
 batch_size)
    bottleneck_features_train = bottleneck_features_train.reshape((16042,25088))

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


save_bottlebeck_features()

'''
```

7.1 Visual features based product similarity.

In [0]:

```
#load the features and corresponding ASINS info.
bottleneck_features_train = np.load('16k_data_cnn_features.npy')
asins = np.load('16k_data_cnn_feature_asins.npy')
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-16)
def get_similar_products_cnn(doc_id, num_results):
    doc_id = asins.index(df_asins[doc_id])
    pairwise_dist = pairwise_distances(bottleneck_features_train, bottleneck_features_t
rain[doc_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['title'])
            print('Euclidean Distance from input image:', pdists[i])
            print('Amazon Url: www.amzon.com/dp/'+ 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 stripes 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 baby blue flower design
Euclidean Distance from input image: 47.9021
Amazon Url: www.amzon.com/dp/B071SBCY9W

Product Title: edv cheetah run purple multi xl
Euclidean Distance from input image: 48.0465
Amazon Url: www.amzon.com/dp/B01CUPYBM0

Product Title: danskin womens vneck loose performance tee xsmall pink omb
re
Euclidean Distance from input image: 48.1019
Amazon Url: www.amzon.com/dp/B01F7PHXY8

Product Title: summer alpaca 3d pastel casual loose tops tee design
Euclidean Distance from input image: 48.1189
Amazon Url: www.amzon.com/dp/B01I80A93G

Product Title: miss chievous juniors striped peplum tank top medium shado
wpeach
Euclidean Distance from input image: 48.1313
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 printed summer tshirts tops
Euclidean Distance from input image: 48.4493
Amazon Url: www.amzon.com/dp/B010NN9RXO

Product Title: nancy lopez whimsy short sleeve whiteblacklemon drop xs
Euclidean Distance from input image: 48.4788
Amazon Url: www.amzon.com/dp/B01MPX6IDX

Product Title: womens tops tees pastel peach ice cream cone print
Euclidean Distance from input image: 48.558
Amazon Url: www.amzon.com/dp/B0734GRKZL

Product Title: uswomens mary j blige without tshirts shirt
Euclidean Distance from input image: 48.6144
Amazon Url: www.amzon.com/dp/B01M0XXFKK

8.0 Assignment

In [2]:

```
#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 CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
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=True)
warnings.filterwarnings("ignore")
```

In [4]:

```
data = pd.read_pickle('16k_apperal_data_preprocessed')
data.head()
```

Out[4]:

	asin	brand	color	medium_image_url	product_type_name	
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	feath ladie slee resis
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	wom uniq cotto spec olym wor..
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	feath ladie mois free spor
27	B014ICEJ1Q	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	supe chibi dear neck
46	B01NACPBG2	Fifth Degree	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	fifth wom gold grap jun..

Utility Functions

In [43]:

```
# Utility Functions which we will use through the rest of the workshop.
import PIL.Image #https://stackoverflow.com/questions/10748822/img-image-openfp-attribu
teerror-class-image-has-no-attribute-open

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

#plotting code to understand the algorithm's decision.
def plot_heatmap(keys, values, labels, url, text):
        # keys: list of words of recommended title
        # values: len(values) ==  len(keys), values(i) represents the occurence of the
 word keys(i)
        # labels: len(labels) == len(keys), the values of labels depends on the model w
e are using
                # if model == 'bag of words': labels(i) = values(i)
                # if model == 'tfidf weighted bag 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_ratios=[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
title1 and list of words of title2), in black if not
        ax = sns.heatmap(np.array([values]), annot=np.array([labels]))
        ax.set_xticklabels(keys) # set that axis labels as the words of title
        ax.set_title(text) # apparel title

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

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

        # displays combine figure ( heat map and 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 to 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 t
he distance between two title vec's
    intersection = set(vec1.keys()) & set(vec2.keys())

    # we set the values of non intersecting words to zero, this is just to show the dif
ference 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 values of labels depends on the model we ar
e using
        # if model == 'bag of words': labels(i) = values(i)
        # if model == 'tfidf weighted bag of words':labels(i) = tfidf(keys(i))
        # if model == 'idf weighted bag of words':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 corpu
s
            # tfidf_title_features[doc_id, index_of_word_in_corpus] will give the tfidf
value of word in given document (doc_id)
            if x in  tfidf_title_vectorizer.vocabulary_:
                labels.append(tfidf_title_features[doc_id, tfidf_title_vectorizer.vocab
ulary_[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 val
ue of word in given document (doc_id)
            if x in  idf_title_vectorizer.vocabulary_:
                labels.append(idf_title_features[doc_id, idf_title_vectorizer.vocabular
y_[x]])
            else:
                labels.append(0)

    plot_heatmap(keys, values, labels, url, text)


# 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 string, 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 retur
ns dict type object {word1:count}



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

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

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

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

8.1 Obtaining IDF values

In [6]:

```
idf_title_vectorizer = CountVectorizer()
idf_title_features = idf_title_vectorizer.fit_transform(data['title'])

# idf_title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(courpus) returns the a sparase matrix of dimensions #
data_points * #words_in_corpus
# idf_title_features[doc_id, index_of_word_in_corpus] = number of times the word occure
d in that doc
```

In [7]:

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

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

In [8]:

```
# 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 find its idf value
    idf_val = idf(i)

    # to calculate idf_title_features we need to replace the count values with the idf
 values of the word
    # idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0] will retu
rn all documents in which the word i present
    for j in idf_title_features[:, idf_title_vectorizer.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_word_in_courpus] = idf value of word
        idf_title_features[j,idf_title_vectorizer.vocabulary_[i]] = idf_val
```

In [10]:

```
type(idf_title_features)
```

Out[10]:

```
scipy.sparse.csr.csr_matrix
```

8.1.1 Importing the W2V model

In [11]:

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

with open('word2vec_model', 'rb') as handle:
    model = pickle.load(handle)
```

In [12]:

```
# 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 two values
        # if  m_name == 'avg', we will append the model[i], w2v representation of word
 i
        # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)
    vec = []
    for i in sentence.split():
        if i in vocab:
            if m_name == 'weighted' and i in  idf_title_vectorizer.vocabulary_:
                vec.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[
i]] * model[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 just ignoring it
            vec.append(np.zeros(shape=(300,)))
    # we will return a numpy array of shape (#number of words in title * 300 ) 300 = le
n(w2v_model[word])
    # each row represents the word2vec representation of each word (weighted/avg) in gi
ven sentance
    return  np.array(vec)

def get_distance(vec1, vec2):
    # vec1 = np.array(#number_of_words_title1 * 300), each row is a vector of length 30
0 corresponds to each word in give title
    # vec2 = np.array(#number_of_words_title2 * 300), each row is a vector of length 30
0 corresponds to each word in give title

    final_dist = []
    # for each vector in vec1 we caluclate the distance(euclidean) to all vectors in ve
c2
    for i in vec1:
        dist = []
        for j in vec2:
            # np.linalg.norm(i-j) will result the 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 between 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 apparel
    # 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/av
g) of length 300 corresponds to each word in give title
    s1_vec = get_word_vec(sentence1, doc_id1, model)
```

```
    #s2_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/av
g) of length 300 corresponds to each word in give title
    s2_vec = get_word_vec(sentence2, doc_id2, model)

    # s1_s2_dist = np.array(#number of words in title1 * #number of words in title2)
    # s1_s2_dist[i,j] = euclidean distance between 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 imag
e of apparel
    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 pairwise distances
    ax = sns.heatmap(np.round(s1_s2_dist,4), annot=True)
    # set the x axis labels as recommended apparels title
    ax.set_xticklabels(sentence2.split())
    # set the y axis labels as input apparels title
    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 image
    ax.grid(False)
    ax.set_xticks([])
    ax.set_yticks([])
    display_img(url, ax, fig)

    plt.show()
```

In [13]:

```
# vocab = stores all the words that are there in google w2v model
# vocab = model.wv.vocab.keys() # if you are using Google word2Vec

vocab = model.keys()
# this function will add the vectors of each word and returns the avg vector of given s
entance
def build_avg_vec(sentence, num_features, doc_id, m_name):
    # sentace: its title of the apparel
    # num_features: the lenght of word2vec vector, its values = 300
    # m_name: model information it will take two values
        # if  m_name == 'avg', we will append the model[i], w2v representation of word
 i
        # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)

    featureVec = np.zeros((num_features,), dtype="float32")
    # we will intialize a vector of size 300 with all zeros
    # we add each word2vec(wordi) to this fetureVec
    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_ve
ctorizer.vocabulary_[word]] * model[word])
            elif m_name == 'avg':
                featureVec = np.add(featureVec, model[word])
    if(nwords>0):
        featureVec = np.divide(featureVec, nwords)
    # returns the avg vector of given sentance, its of shape (1, 300)
    return featureVec
```

8.1.2 IDF weighted Word2Vec values

In [14]:

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

In [20]:

```
w2v_title_weight.shape
```

Out[20]:

```
(16042, 300)
```

8.2 Weighted similarity using title, brand,color and image.

Repalcing null values in brand & color

In [15]:

```
# 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['brand'].values]
colors = [x.replace(" ", "-") for x in data['color'].values]
```

One hot encoding of color & brand

In [16]:

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

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

In [18]:

```
print("Shape of brand matrix : ",brand_features.shape)
print("Shape of color matrix : ",color_features.shape)
```

```
Shape of brand matrix :  (16042, 3835)
Shape of color matrix :  (16042, 1845)
```

Combining brand & color features using hstack

In [19]:

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

Image features

In [24]:

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

#load the features and corresponding ASINS info.
bottleneck_features_train = np.load('16k_data_cnn_features.npy')
asins = np.load('16k_data_cnn_feature_asins.npy')
asins = list(asins)

# load the original 16K dataset
data = pd.read_pickle('16k_apperal_data_preprocessed')
df_asins = list(data['asin'])
```

In [28]:

```
bottleneck_features_train.shape
```

Out[28]:

```
(16042, 25088)
```

In [35]:

```
def heat_map_w2v_brand(sentance1, sentance2, url, doc_id1, doc_id2, df_id1, df_id2, mod
el):

    # 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 apparel
    # df_id1: index of document1 in the data frame
    # df_id2: index of document2 in the data frame
    # 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/av
g) of length 300 corresponds to each word in give title
    s1_vec = get_word_vec(sentance1, doc_id1, model)
    #s2_vec = np.array(#number_of_words_title2 * 300), each row is a vector(weighted/av
g) of length 300 corresponds to each word in give title
    s2_vec = get_word_vec(sentance2, doc_id2, model)

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

    data_matrix = [['Asin','Brand', 'Color', 'Product type'],
                [data['asin'].loc[df_id1],brands[doc_id1], colors[doc_id1]], # input app
arel's features
                [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2]]] # recommend
ed apparel's features

    colorscale = [[0, '#1d004d'],[.5, '#f2e5ff'],[1, '#f2e5d1']] # to color the heading
s of each 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='simple_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 pairwise distances
    ax1 = sns.heatmap(np.round(s1_s2_dist,6), annot=True)
    # set the x axis labels as recommended apparels title
    ax1.set_xticklabels(sentance2.split())
    # set the y axis labels as input apparels title
    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()
```

8.3 Weighted model

In [36]:

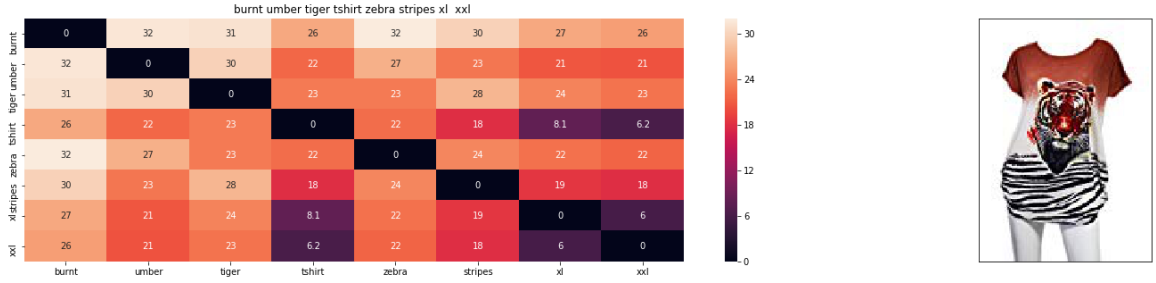
```
def idf_w2v_brand(doc_id, w1, w2, w3, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand and color features

    # pairwise_dist will store the distance from given input apparel to all remaining a
pparels
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <
X, Y> / (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].resha
pe(1,-1))
    ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id])
    img_feat_dist = pairwise_distances(bottleneck_features_train, bottleneck_features_t
rain[doc_id].reshape(1, -1))  #image
    pairwise_dist   = (w1 * idf_w2v_dist +  w2 * ex_feat_dist + w3 * img_feat_dist)/flo
at(w1 + w2+ w3)

    # np.argsort will return indices of 9 smallest 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 distace'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_indice
s[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i],df_indices[0
], df_indices[i], 'weighted')
        print('Product title :',data['title'].loc[df_indices[i]])
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print('Brand :',data['brand'].loc[df_indices[i]])
        print('euclidean distance from input :', pdists[i])
        print('='*125)
```

8.4 Testing

In [47]:

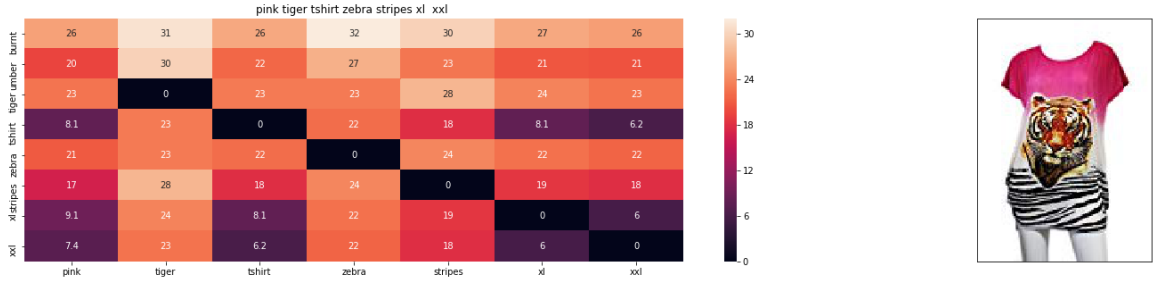
```
# brand and color weight =10
# title vector weight = 100
# image vector weight= 10

idf_w2v_brand(12566, 100, 10, 10, 20)
# in the give heat map, each cell contains the euclidean distance between words i, j
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



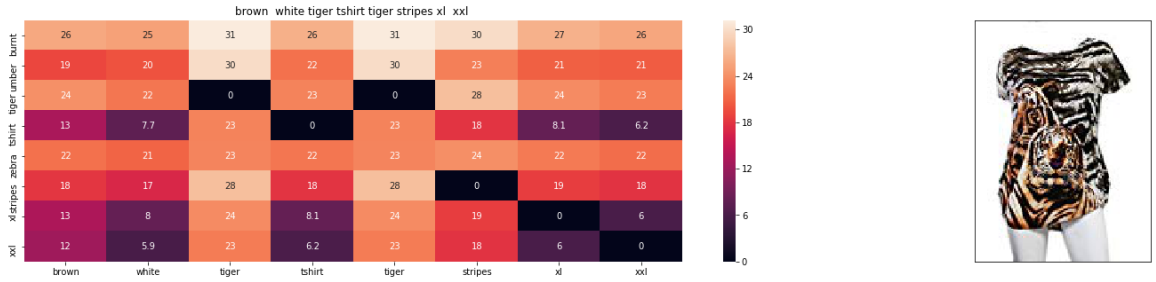
Product title : burnt umber tiger tshirt zebra stripes xl xxl
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 6.257698866344678e-07
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



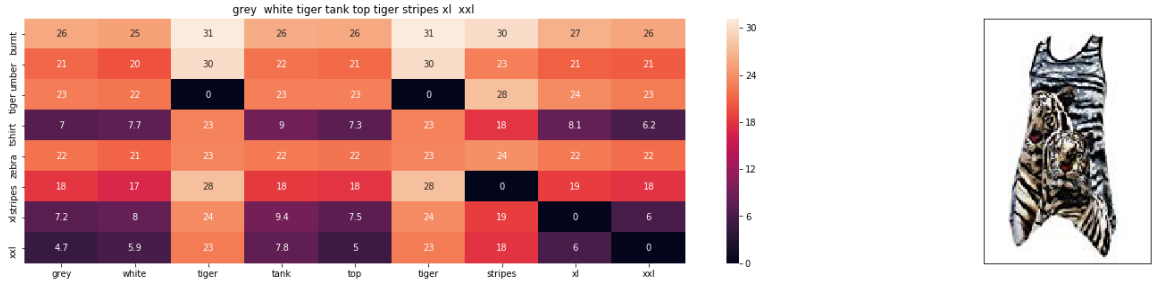
Product title : pink tiger tshirt zebra stripes xl xxl
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 7.546907679270023
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



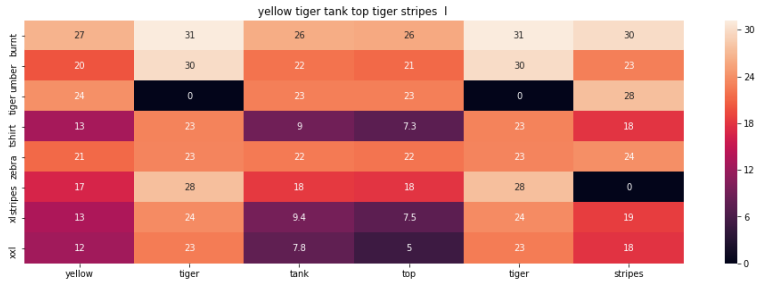
Product title : brown white tiger tshirt tiger stripes xl xxl
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 8.604771423339844
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



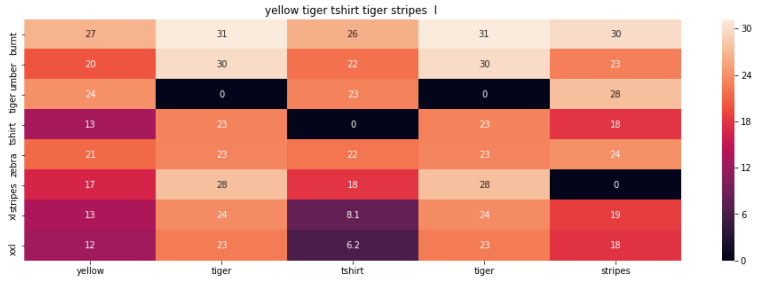
Product title : grey white tiger tank top tiger stripes xl xxl
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 9.078535842925623
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



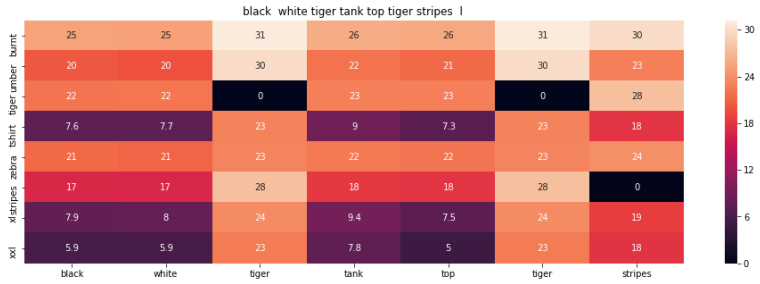
Product title : yellow tiger tank top tiger stripes l
ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 9.315694554676925
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



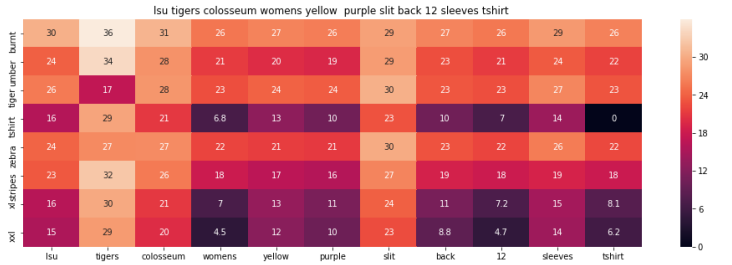
Product title : yellow tiger tshirt tiger stripes l
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 9.377141698231613
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



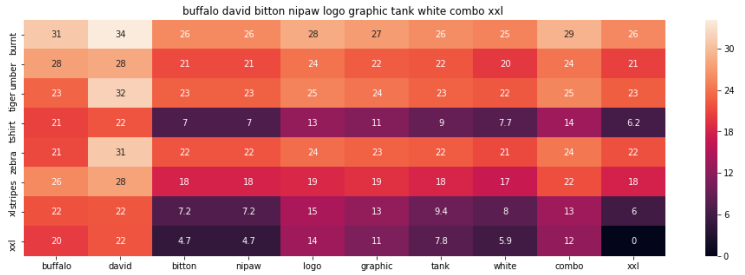
Product title : black white tiger tank top tiger stripes l
ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 9.464477920562341
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



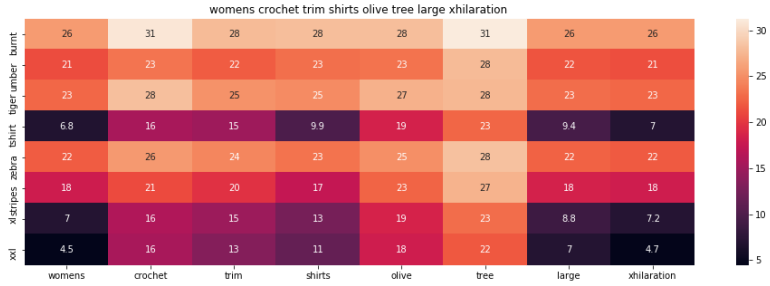
Product title : lsu tigers colosseum womens yellow purple slit back 12 sl
eeves tshirt
ASIN : B073R5Q8HD
Brand : Colosseum
euclidean distance from input : 9.52716109081704
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



Product title : buffalo david bitton nipaw logo graphic tank white combo x xl
ASIN : B018H5AZXQ
Brand : Buffalo
euclidean distance from input : 9.685298359912922
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown

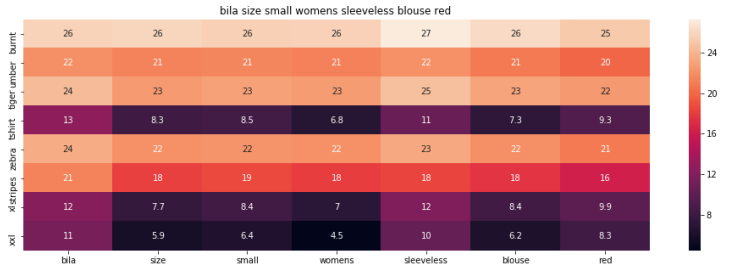


Product title : womens crochet trim shirts olive tree large xhilaration
ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 9.91272476168701
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown

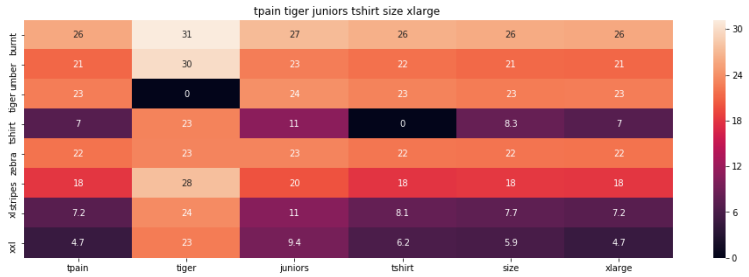
Product title : h bordeaux white womens small striped tee shirt brown
ASIN : B072BVB47Z
Brand : H By Bordeaux
euclidean distance from input : 9.918665059407552
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



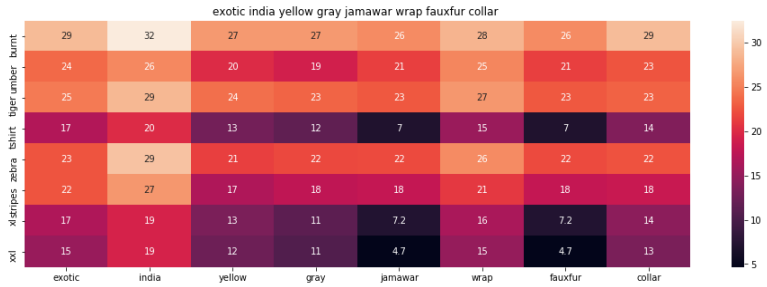
Product title : bila size small womens sleeveless blouse red
ASIN : B01L7ROZNC
Brand : Bila
euclidean distance from input : 9.94118213777342
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



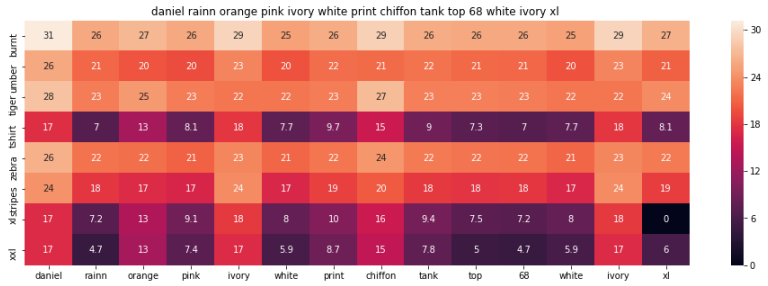
Product title : tpain tiger juniors tshirt size xlarge
ASIN : B01K0H02OG
Brand : Tultex
euclidean distance from input : 9.949287860836572
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



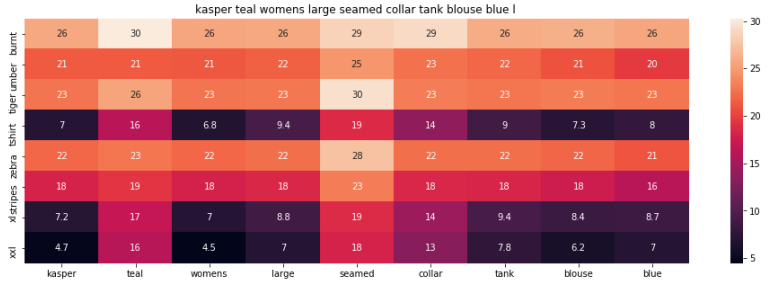
Product title : exotic india yellow gray jamawar wrap fauxfur collar
ASIN : B073ZHRBV8
Brand : Exotic India
euclidean distance from input : 9.978846943897297
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



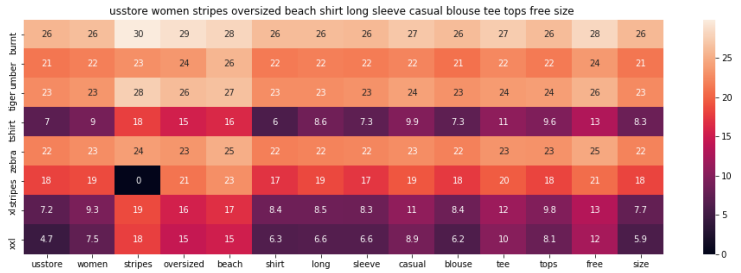
Product title : daniel rainn orange pink ivory white print chiffon tank to
p 68 white ivory xl
ASIN : B01IPV1SFQ
Brand : Daniel Rainn
euclidean distance from input : 10.017984463910329
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



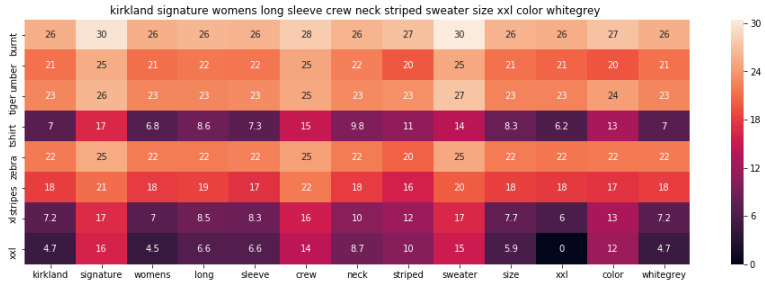
Product title : kasper teal womens large seamed collar tank blouse blue l
ASIN : B0722DJVQP
Brand : Kasper
euclidean distance from input : 10.031784249589828
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



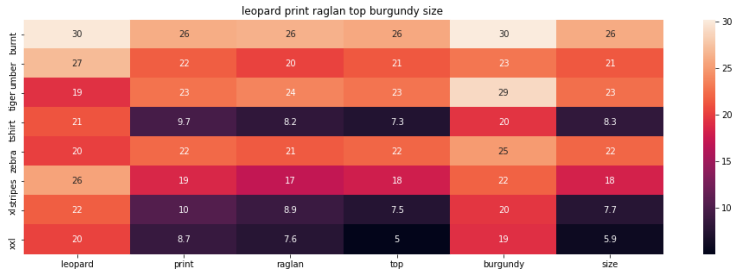
Product title : usstore women stripes oversized beach shirt long sleeve ca
sual blouse tee tops free size
ASIN : B01DNNI1RO
Brand : Usstore
euclidean distance from input : 10.045320841195473
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



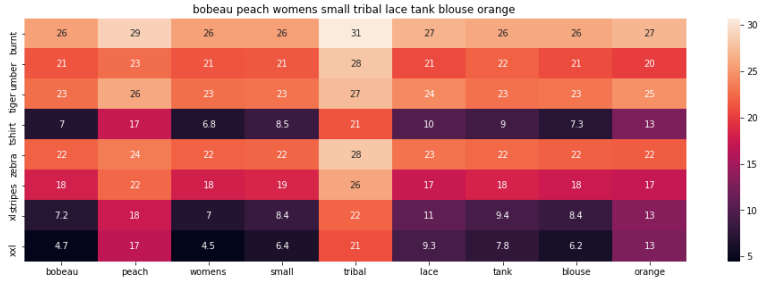
Product title : kirkland signature womens long sleeve crew neck striped sw
eater size xxl color whitegrey
ASIN : B06XTPC3FP
Brand : Kirkland Signature
euclidean distance from input : 10.050960235319822
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



Product title : leopard print raglan top burgundy size
ASIN : B01C6ORLDQ
Brand : 1 Mad Fit
euclidean distance from input : 10.086430053435057
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown

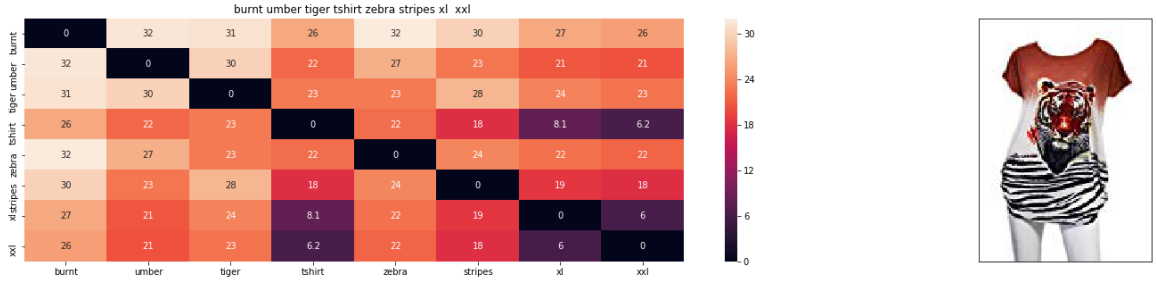


Product title : bobeau peach womens small tribal lace tank blouse orange
ASIN : B072JTHCX6
Brand : Bobeau
euclidean distance from input : 10.089473916337875
==========
==========

In [48]:

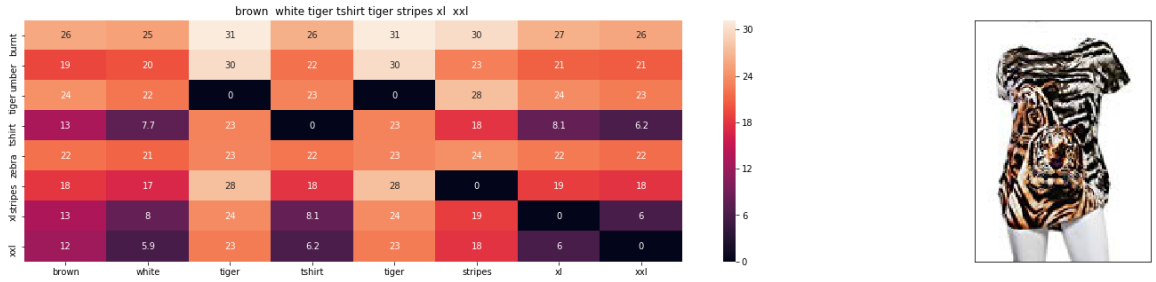
```
# brand and color weight =100
# title vector weight = 10
# image vector weight= 10

idf_w2v_brand(12566, 10, 100, 10, 20)
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



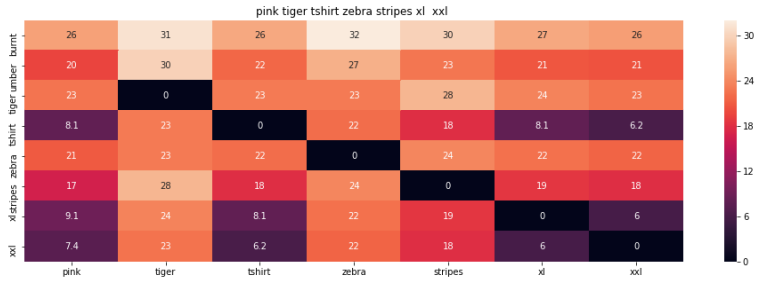
Product title : burnt umber tiger tshirt zebra stripes xl xxl
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 6.257698866344678e-07
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



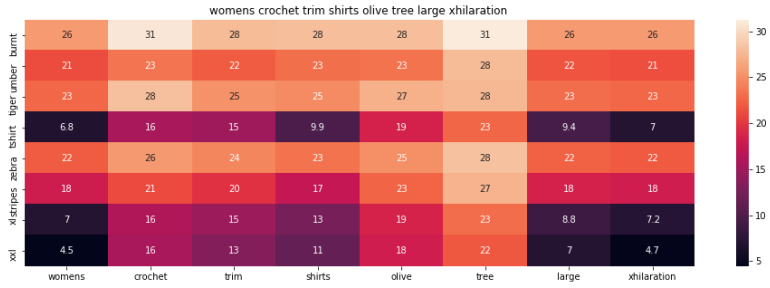
Product title : brown white tiger tshirt tiger stripes xl xxl
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 5.0265655517578125
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



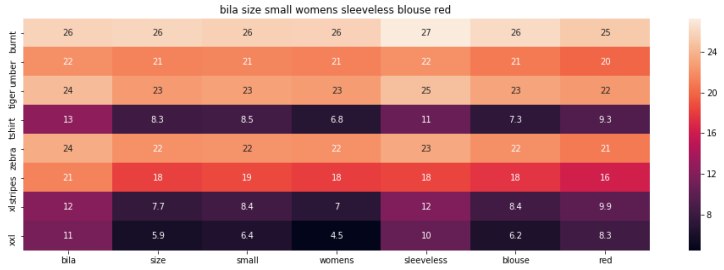
Product title : pink tiger tshirt zebra stripes xl xxl
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 5.559652805629516
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



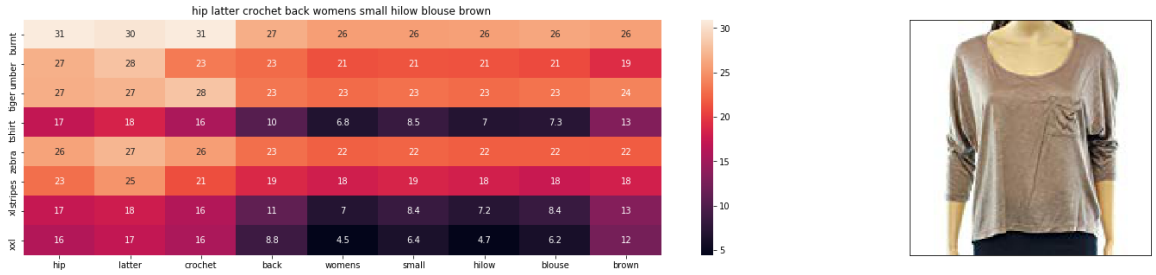
Product title : womens crochet trim shirts olive tree large xhilaration
ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 5.801184397163879
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



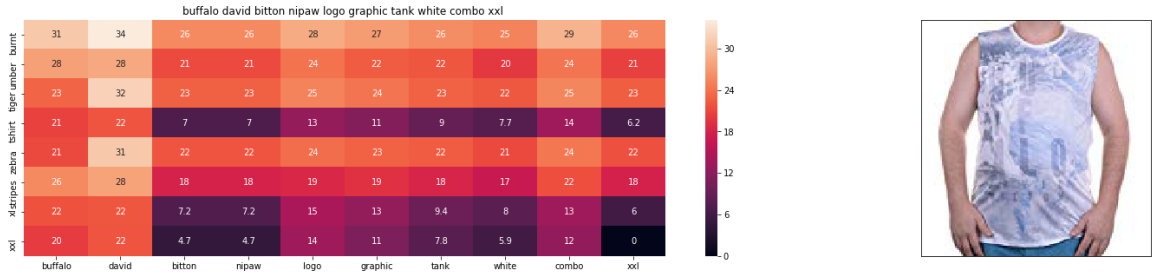
```
Product title : bila size small womens sleeveless blouse red
ASIN : B01L7ROZNC
Brand : Bila
euclidean distance from input : 5.916607551236805
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



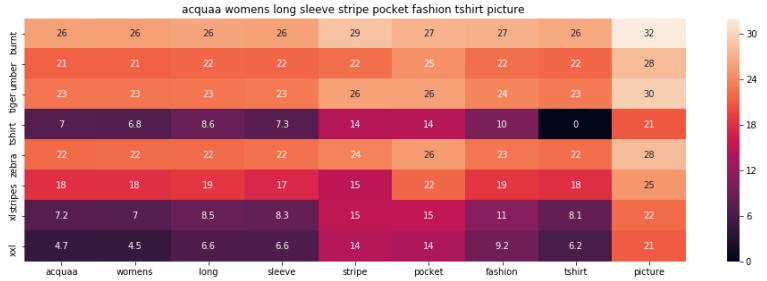

```
Product title : hip latter crochet back womens small hilow blouse brown
ASIN : B074MJN1K9
Brand : Hip
euclidean distance from input : 5.930494076090926
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



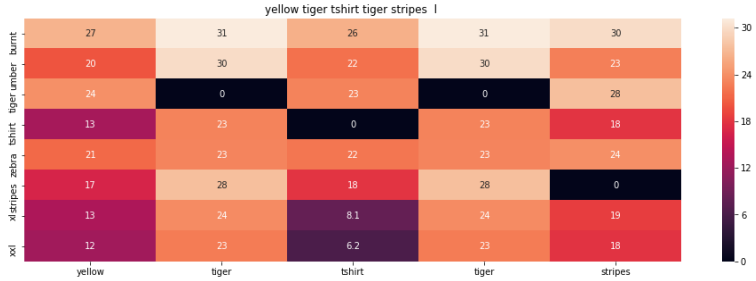

```
Product title : buffalo david bitton nipaw logo graphic tank white combo x
xl
ASIN : B018H5AZXQ
Brand : Buffalo
euclidean distance from input : 5.956514800809712
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



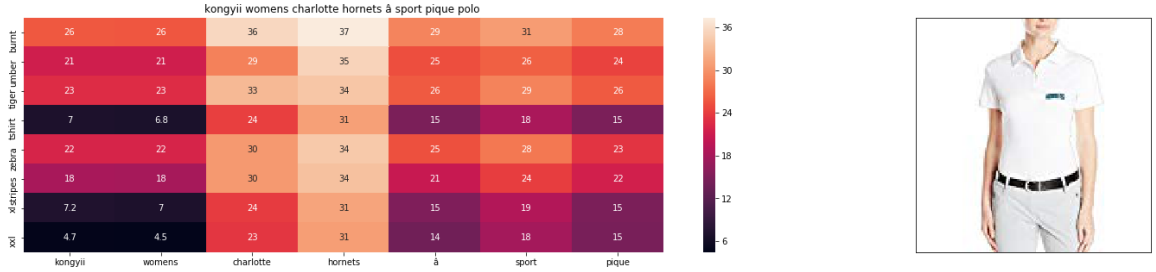
```
Product title : acquaa womens long sleeve stripe pocket fashion tshirt pic
ture
ASIN : B06XK2ZRFH
Brand : Acquaa
euclidean distance from input : 5.979794692993164
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



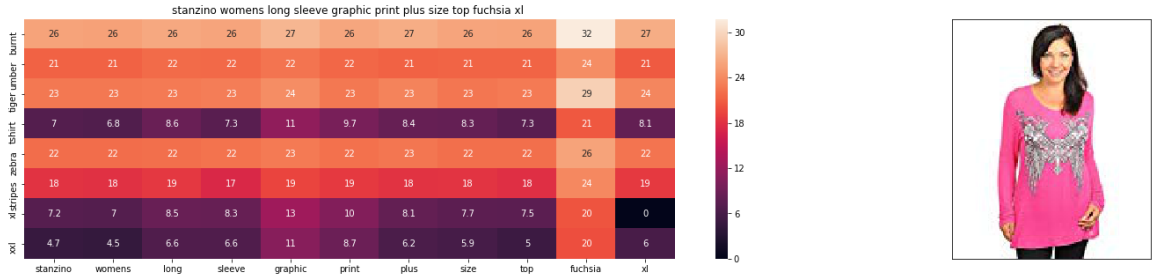
```
Product title : yellow tiger tshirt tiger stripes  l
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 6.017720063828572
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



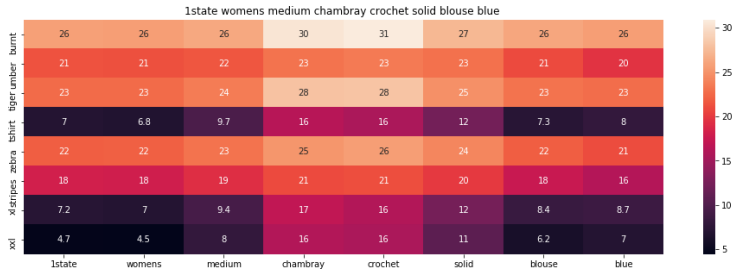
Product title : kongyii womens charlotte hornets â sport pique polo
ASIN : B01FJVZST2
Brand : KONGYII
euclidean distance from input : 6.030832748075183
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



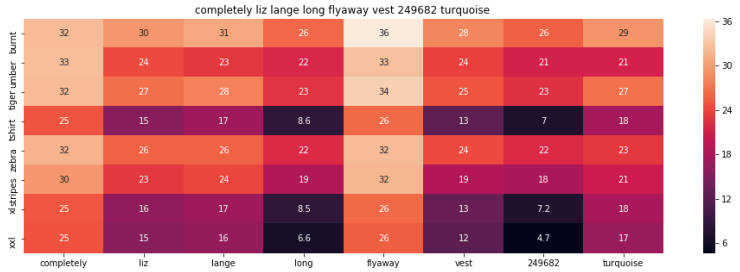
Product title : stanzino womens long sleeve graphic print plus size top fu
chsia xl
ASIN : B00DP4VHWI
Brand : Stanzino
euclidean distance from input : 6.03928662584211
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



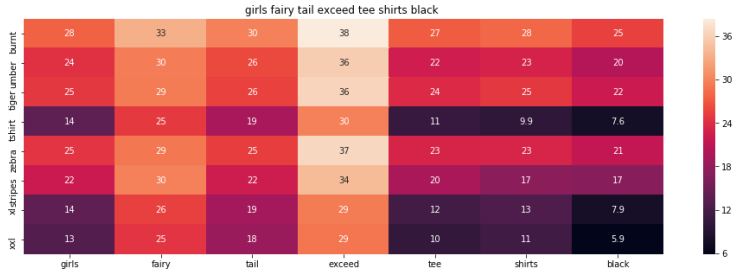
Product title : 1state womens medium chambray crochet solid blouse blue
ASIN : B074MK6LV2
Brand : 1.State
euclidean distance from input : 6.041929003059721
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



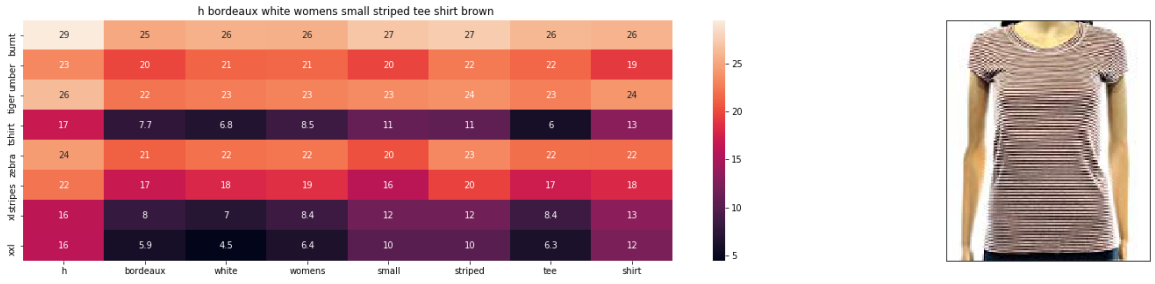
Product title : completely liz lange long flyaway vest 249682 turquoise
ASIN : B074LTBWSW
Brand : Liz Lange
euclidean distance from input : 6.044797958158345
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



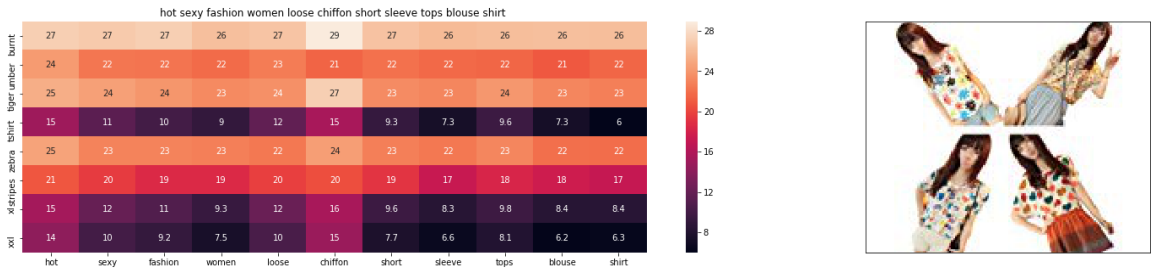
```
Product title : girls fairy tail exceed tee shirts black
ASIN : B01L9F153U
Brand : ATYPEMX
euclidean distance from input : 6.066318651497221
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



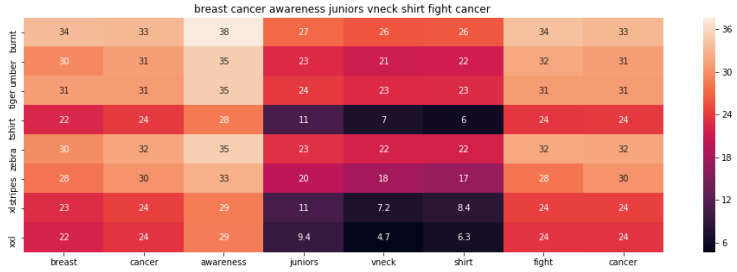

```
Product title : h bordeaux white womens small striped tee shirt brown
ASIN : B072BVB47Z
Brand : H By Bordeaux
euclidean distance from input : 6.0688781102498375
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



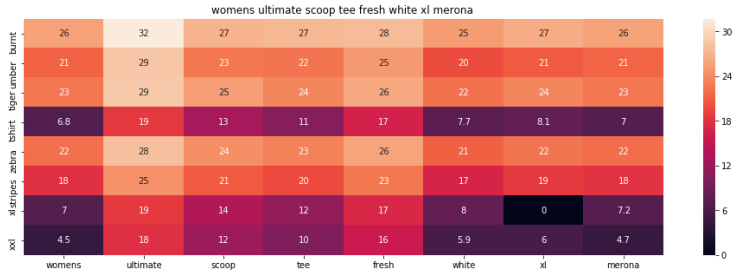

```
Product title : hot sexy fashion women loose chiffon short sleeve tops blo
use shirt
ASIN : B00JMAASRO
Brand : Wotefusi
euclidean distance from input : 6.069353624641752
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



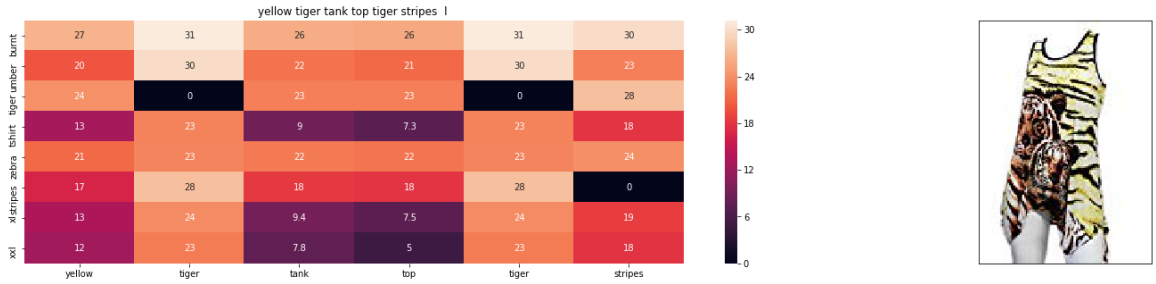
Product title : breast cancer awareness juniors vneck shirt fight cancer
ASIN : B016CU40IY
Brand : Juiceclouds
euclidean distance from input : 6.070301831225411
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



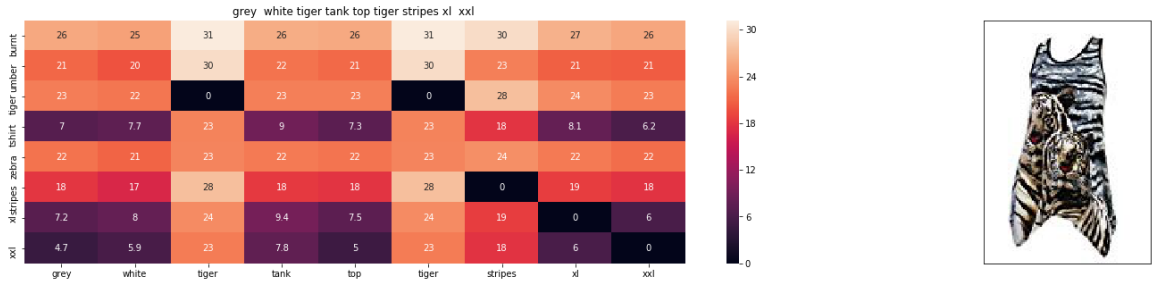
Product title : womens ultimate scoop tee fresh white xl merona
ASIN : B01G7XE50E
Brand : Merona
euclidean distance from input : 6.098146578132964
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



Product title : yellow tiger tank top tiger stripes l
ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 6.109212684932495
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown

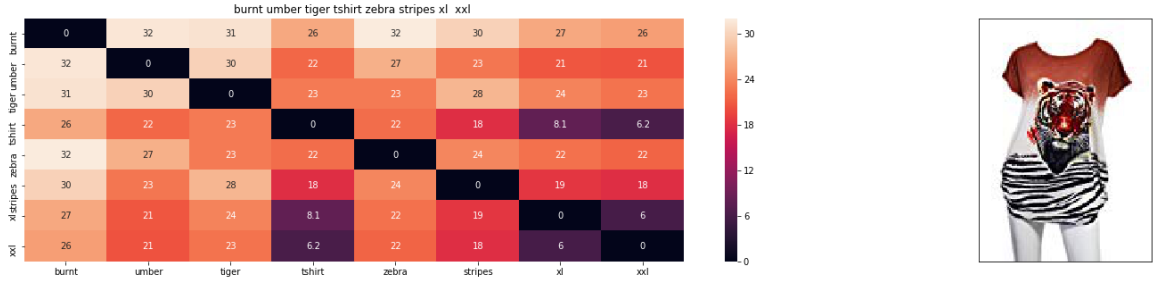


Product title : grey white tiger tank top tiger stripes xl xxl
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 6.119075616501912
==========
==========

In [49]:

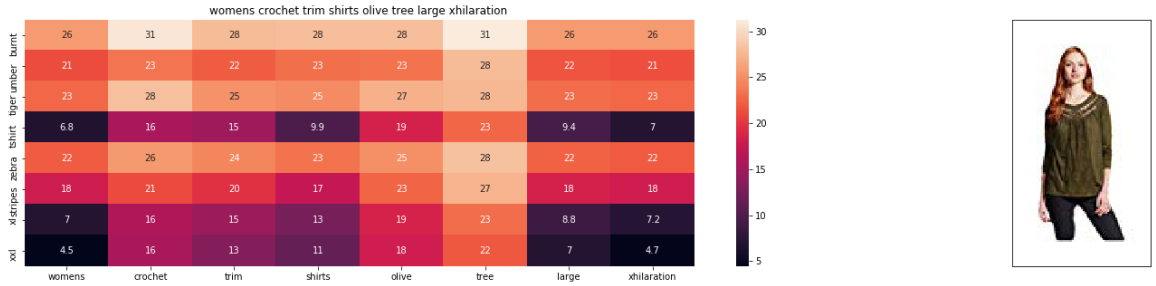
```
# brand and color weight =10
# title vector weight = 10
# image vector weight= 100

idf_w2v_brand(12566, 10, 10, 100, 20)
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



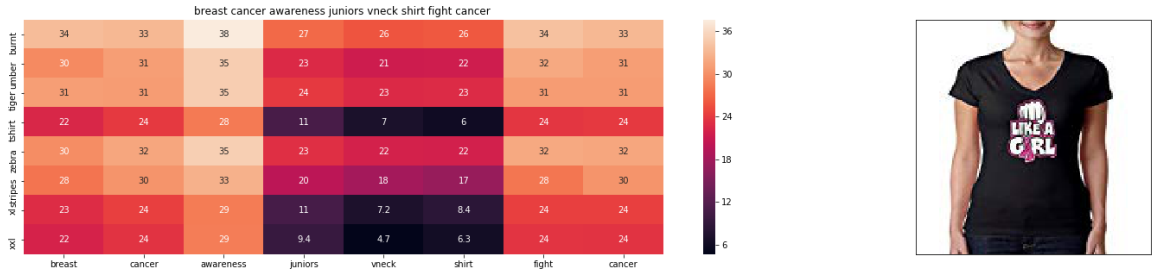
Product title : burnt umber tiger tshirt zebra stripes xl xxl
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 6.257698987610638e-06
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



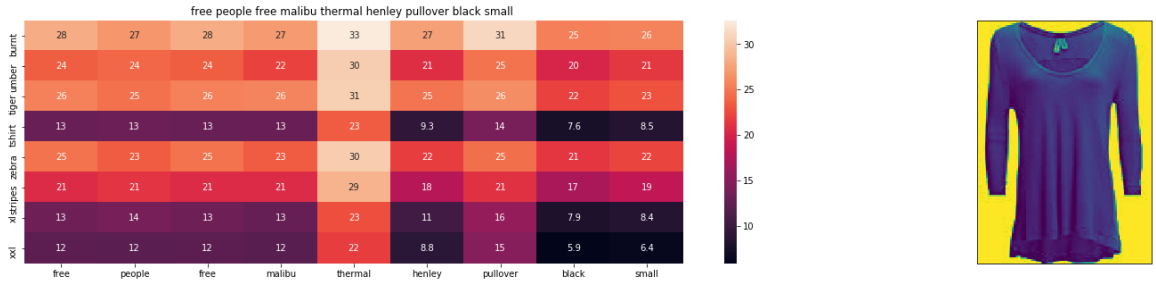
Product title : womens crochet trim shirts olive tree large xhilaration
ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 31.85489528628418
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



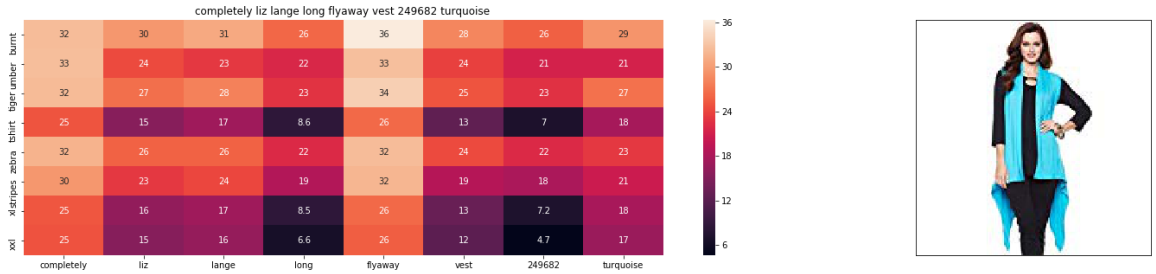
Product title : breast cancer awareness juniors vneck shirt fight cancer
ASIN : B016CU40IY
Brand : Juiceclouds
euclidean distance from input : 33.128092767045885
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



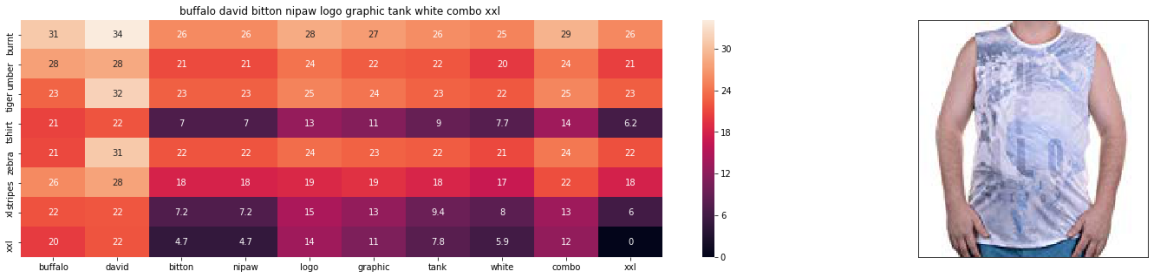
Product title : free people free malibu thermal henley pullover black smal
l
ASIN : B074MXY984
Brand : We The Free
euclidean distance from input : 33.72652264648683
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



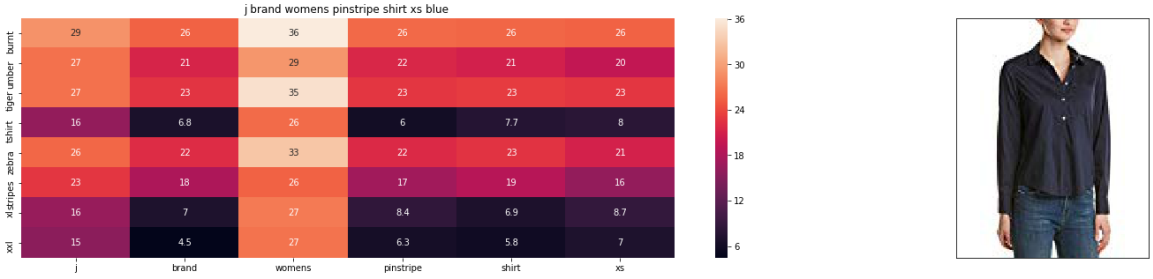
```
Product title : completely liz lange long flyaway vest 249682 turquoise
ASIN : B074LTBWSW
Brand : Liz Lange
euclidean distance from input : 33.73082823407877
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



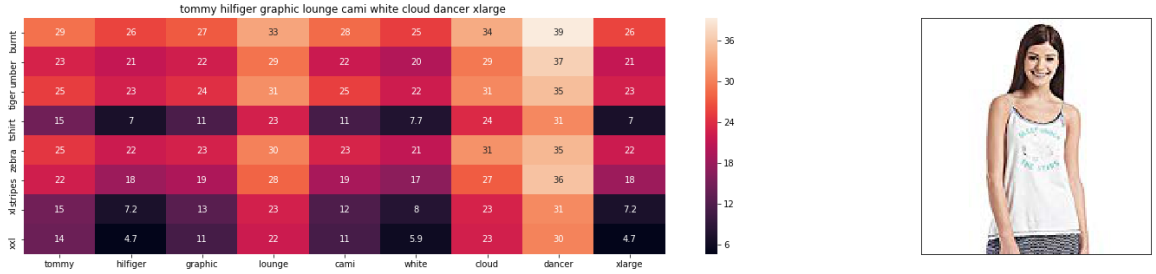

```
Product title : buffalo david bitton nipaw logo graphic tank white combo x
xl
ASIN : B018H5AZXQ
Brand : Buffalo
euclidean distance from input : 33.79095765404134
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



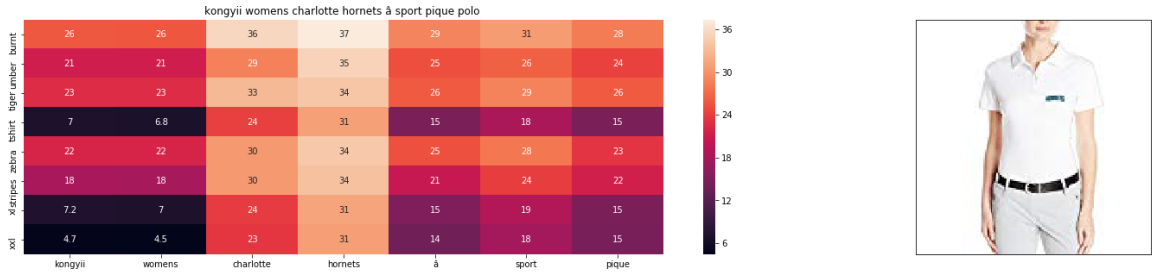

```
Product title : j brand womens pinstripe shirt xs blue
ASIN : B06XYP1X1F
Brand : J Brand Jeans
euclidean distance from input : 33.90401357020997
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown




```
Product title : tommy hilfiger graphic lounge cami white cloud dancer xlar
ge
ASIN : B01BMSFYW2
Brand : igertommy hilf
euclidean distance from input : 34.09271233882569
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown

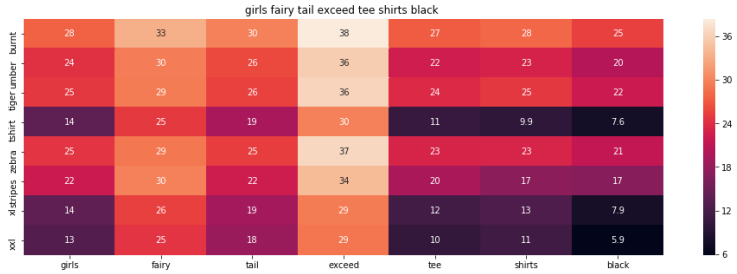



```
Product title : kongyii womens charlotte hornets â sport pique polo
ASIN : B01FJVZST2
Brand : KONGYII
euclidean distance from input : 34.89945233786701
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown

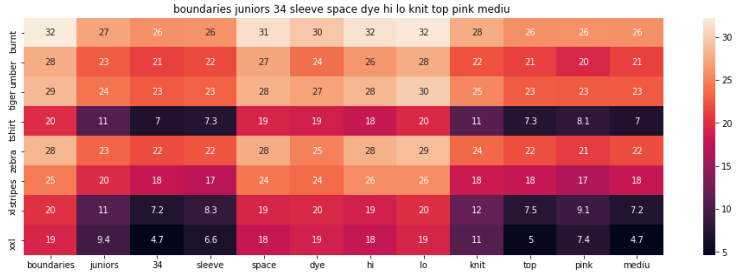
Product title : bila size small womens sleeveless blouse red
ASIN : B01L7ROZNC
Brand : Bila
euclidean distance from input : 35.016890654212226
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



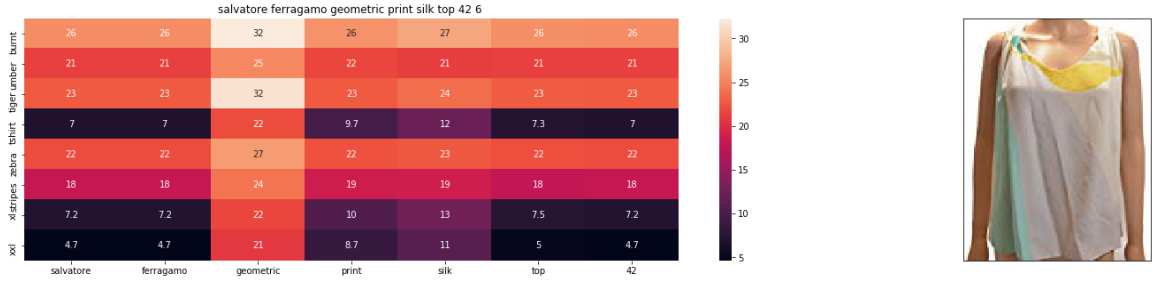
Product title : girls fairy tail exceed tee shirts black
ASIN : B01L9F153U
Brand : ATYPEMX
euclidean distance from input : 35.57922255321939
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



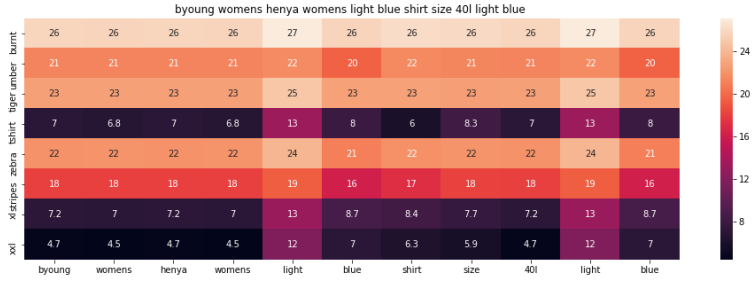
```
Product title : boundaries juniors 34 sleeve space dye hi lo knit top pink
mediu
ASIN : B01EXXFS4M
Brand : No Boundaries
euclidean distance from input : 35.62718144707112
================================================================================
================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



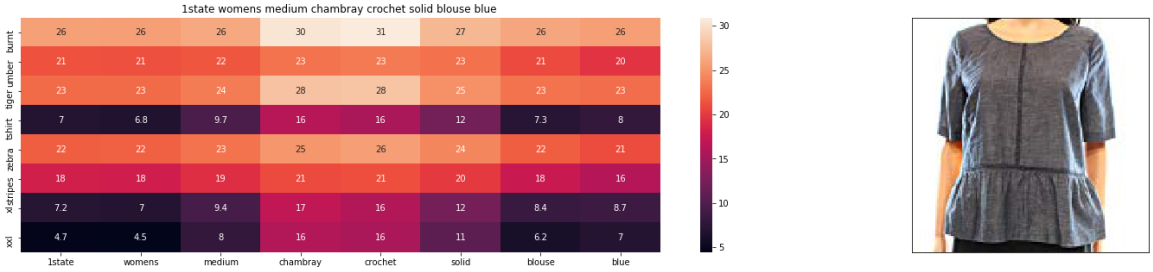

```
Product title : salvatore ferragamo geometric print silk top 42 6
ASIN : B0756JTS1F
Brand : Salvatore Ferragamo
euclidean distance from input : 35.793976094464526
================================================================================
================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



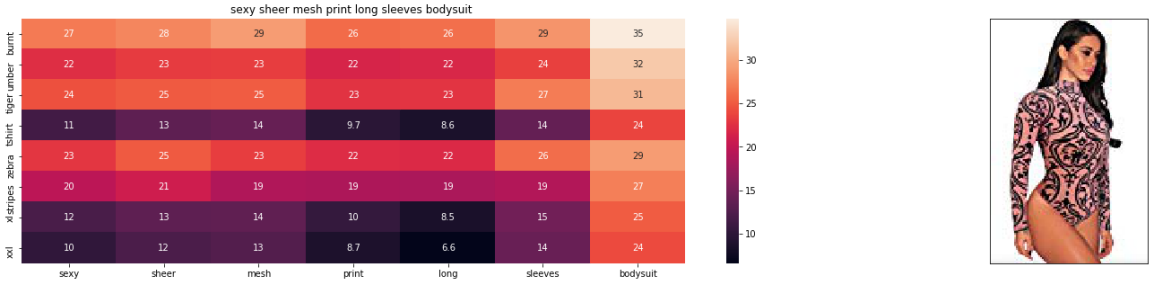
```
Product title : byoung womens henya womens light blue shirt size 40l light
blue
ASIN : B06Y41MRCH
Brand : Byoung
euclidean distance from input : 35.83495529147216
================================================================================
================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



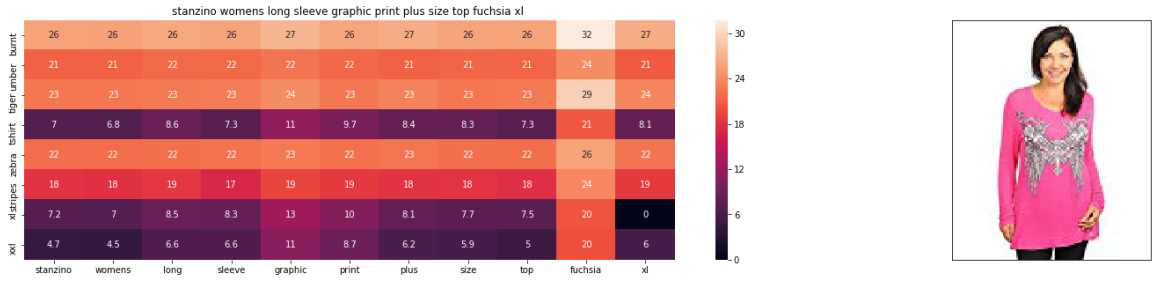
Product title : 1state womens medium chambray crochet solid blouse blue
ASIN : B074MK6LV2
Brand : 1.State
euclidean distance from input : 35.901989683753236
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



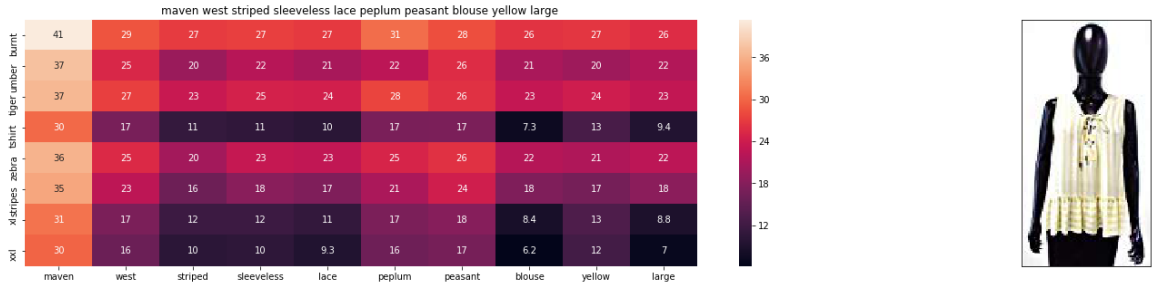
Product title : sexy sheer mesh print long sleeves bodysuit
ASIN : B074Z5C98D
Brand : Ariella's closet
euclidean distance from input : 36.023421923319496
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



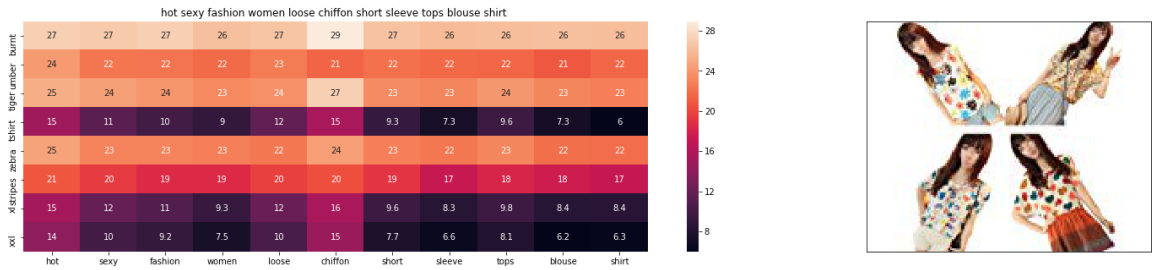
Product title : stanzino womens long sleeve graphic print plus size top fu
chsia xl
ASIN : B00DP4VHWI
Brand : Stanzino
euclidean distance from input : 36.144602331763494
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



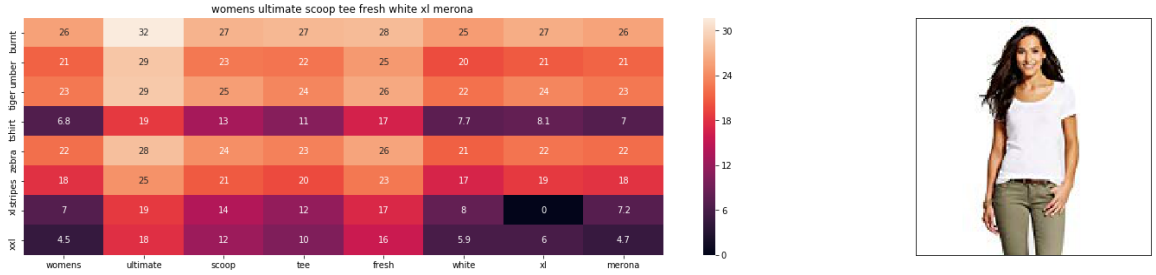
Product title : maven west striped sleeveless lace peplum peasant blouse y
ellow large
ASIN : B01M8GB3AL
Brand : Maven West
euclidean distance from input : 36.18038687360514
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown



```
Product title : hot sexy fashion women loose chiffon short sleeve tops blo
use shirt
ASIN : B00JMAASRO
Brand : Wotefusi
euclidean distance from input : 36.19655806347329
================================================================================
==================================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown




```
Product title : womens ultimate scoop tee fresh white xl merona
ASIN : B01G7XE50E
Brand : Merona
euclidean distance from input : 36.22421957775552
================================================================================
==================================================
```

8.4.1 Observations on testing:

- It can be observed that recommendations were poor when more weightage was given to brand/color & image.[observe case 2 & 3 in the testing phase]
- Recommendations were better when more weightage was given to titles [observe case 1 in the testing phase]

9.0 Procedure followed to solve the case study

1. Basic stats of features brand, color, product type, title & price were studied.
2. With the help of the stats, null values were found and the rows containing them were eliminated. The number of data points reduced from 183k to 28k as a result of this operation.
3. Next, deduplication was performed on the data were products with duplicate titles, products with title count < 4, products with titles differing in last 4 words & titles which were semantically similar. The number of datapoints finally reduced to around 16k as a result of this operation.
4. Next step was to preprocess the title which involved removal of stopwords, html tags, special characters & converting text to lower case.
5. The title was vectorized using BOW, TFIDF, AvgW2V & IDF weighted W2V.
6. Product similarity was tested for each of the featurizations as mentioned above.
7. Features brand, color, product type were featurized using one-hot encoding.
8. In case of image based similarity, the images were converted to vectors using VGG16 CNN.
9. Product similarity was tested using image features.
10. IDF weighted Title features,color, brand & image features were concatenated with different weights applied to each of the features and hence product similarity was tested.