

Amazon Apparel Recommendations

[4.3] Overview of the data

In [78]:

```
#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 [3]:

```
# 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 [4]:

```
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 [5]:

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

Out[5]:

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

Of these 19 features, we will be using only 6 features in this workshop.

1. asin (Amazon standard 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 apparel, 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 [6]:

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

In [7]:

```
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[7]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
0	B016I2TS4W	FNC7C	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Minions Como Superheroes Ironman Long Sleeve R...	None
1	B01N49AI08	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG Clothing Womens Izo Tunic	None
2	B01JDPCOHO	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG Clothing Womens Won Top	None
3	B01N19U5H5	Focal18	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Focal18 Sailor Collar Bubble Sleeve Blouse Shi...	None
4	B004GSI2OS	FeatherLite	Onyx Black/Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	Featherlite Ladies' Long Sleeve Stain Resistan...	\$26.26

[5.1] Missing data for various features.

Basic stats for the feature: product_type_name

In [8]:

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

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

# 91.62% (167794/183138) of the products are shirts,

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

In [9]:

```
# 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 [10]:

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

Out[10]:

```
[('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 [11]:

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

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

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

In [12]:

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

```
Siara_CoachMode_Connor(10)
```

Out[12]:

```
[('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 [13]:

```
print(data['color'].describe())  
  
# we have 7380 unique colors  
# 7.2% of products are black in color  
# 64956 of 183138 products have brand information. That's approx 35.4%.
```

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

In [14]:

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

Out[14]:

```
[(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 [15]:

```
print(data['formatted_price'].describe())  
  
# Only 28,395 (15.5% of whole data) products with price information  
  
count      28395  
unique     3135  
top        $19.99  
freq       945  
Name: formatted_price, dtype: object
```

In [16]:

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

```
Out[16]:
```

```
[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 [17]:
```

```
print(data['title'].describe())  
  
# 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.
```

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

```
In [0]:
```

```
data.to_pickle('pickels/180k_apparel_data')
```

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

```
In [18]:
```

```
# 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 [19]:
```

```
# 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 their 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 index, row in  
images.iterrows():\n    url = row['large_image_url']\n    response = requests.get(url)\n    img = Image.open(BytesIO(response.content))\n    img.save('workshop/images/28k_images/'+row['asin']+'.jpeg')\n\n"
```

[5.2] Remove near duplicate items

[5.2.1] Understand about duplicates.

```
In [20]:
```

```
# 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')))  
# we have 2325 products which have same title but different color
```

2325

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

□ B00AQ4GMCK	□ B00AQ4GMTS
□ B00AQ4GMLQ	□ B00AQ4GN3I

These shirts exactly same except in color

□ B00G278GZ6	□ B00G278W6O
□ B00G278Z2A	□ B00G2786X8

In our data there are many duplicate products like the above examples, we need to de-dupe them for better results.

[5.2.2] Remove duplicates : Part 1

```
In [21]:
```

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

In [22]:

```
data.head()
```

Out[22]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
4	B004GSI2OS	FeatherLite	Onyx Black/Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	Featherlite Ladies' Long Sleeve Stain Resistan...	\$26.26
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	Women's Unique 100% Cotton T - Special Olympic...	\$9.99
11	B001LOUGE4	Fitness Etc.	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	Ladies Cotton Tank 2x1 Ribbed Tank Top	\$11.99
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	FeatherLite Ladies' Moisture Free Mesh Sport S...	\$20.54
21	B014ICEDNA	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	Supernatural Chibis Sam Dean And Castiel Short...	\$7.50

In [23]:

```
# 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 [24]:

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

Out[24]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
61973	B06Y1KZ2WB	Éclair	Black/Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	Éclair Women's Printed Thin Strap Blouse Black...	\$24.99
133820	B010RV33VE	xiaoming	Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Womens Sleeveless Loose Long T-shirts...	\$18.19
81461	B01DDSDLNS	xiaoming	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Women's White Long Sleeve Single Brea...	\$21.58
75005	DOOKELV000V		Red	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Stripes Tank Tops/Blouse	\$15.04

75995	B00XJLTC091	xiaoming asin	xiaoming brand	Anchor... or	images- amazon.com/images... medium_image_url	SHIRT product_type_name	Patch/Dear Sleeve	\$15.91 title	\$15.91 formatted_price
							Anchor...		
151570	B00WPJG35K	xiaoming	White		https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Sleeve Sheer Loose Tassel Kimono Woma...		\$14.32

Some examples of duplicate 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 Print Head Shirt for woman Neon Wolf t-shirt
- 62. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print Head Shirt for woman Neon Wolf t-shirt
- 63. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print Head Shirt for woman Neon Wolf t-shirt
- 64. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print Head Shirt for woman Neon Wolf t-shirt

In [25]:

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

In [26]:

```
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', 'Queen', '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
        appended None in case of unequal strings
```

```

# example: a =['a', 'b', 'c', 'd']
# b = ['a', 'b', 'd']
# itertools.zip_longest(a,b): will give [('a','a'), ('b','b'), ('c','d'), ('d', None)]
for k in itertools.zip_longest(a,b):
    if (k[0] == k[1]):
        count += 1

    # if the number of words in which both strings differ are > 2 , we are considering it as those two apperals are different
    # if the number of words in which both strings differ are < 2 , we are considering it as those two apperals are same, hence we are 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'].loc[indices[j]])

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

```

In [27]:

```
data = data.loc[data['asin'].isin(stage1_dedupe_asins)]
```

We removed the dupliactes which differ only at the end.

In [28]:

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

Number of data points : 17593

In [0]:

```
data.to_pickle('pickels/17k_apperal_data')
```

[5.2.3] Remove duplicates : Part 2

In the previous cell, we sorted whole data in alphabetical order of titles. Then, 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, XX-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 [29]:
```

```
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_dedupes_asins = []  
while len(indices) != 0:  
    i = indices.pop()  
    stage2_dedupes_asins.append(data['asin'].loc[i])  
    # consider the first apparel's title  
    a = data['title'].loc[i].split()  
    # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'X-Large']  
    for j in indices:  
  
        b = data['title'].loc[j].split()  
        # store the list of words of jth string in b, ex: b = ['tokidoki', 'The', 'Queen', '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  
        appended None in case of unequal strings  
        # example: a = ['a', 'b', 'c', 'd']  
        # b = ['a', 'b', 'd']  
        # itertools.zip_longest(a,b): will give [(a,a), (b,b), (c,d), (d, None)]  
        for k in itertools.zip_longest(a, b):  
            if (k[0] == k[1]):  
                count += 1  
  
        # if the number of words in which both strings differ are < 3 , we are considering it as those two apperals are same, hence we are 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_dedupes_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.
```

6. Text pre-processing

```
In [30]:
```

```

data = pd.read_pickle('pickels/16k_apperial_data')

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

```

In [31]:

```

# 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()))
            # Convert 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: {"hadn't", 'doesn', 'ain', 'we', "doesn't", "mustn't", 'with', 'himself', 'more', 'ours', 'should've', 'am', 'itself', 'be', 'there', 'an', 'and', 'shouldn', 'it's', 'does', 'him', 'just', 'into', 'no', 'against', 'about', 'mightn', "don't", 'your', 'down', 'when', "you'll", 'won', 'at', 'who', 'how', 'ourselves', 'each', 'been', 'not', "that'll", 'were', 'any', 'weren', 'yourself', 'before', 'those', "isn't", 'that', 'under', 'my', "mightn't", 'haven', 'same', 'because', 'shan', 'he', 'this', 'in', 'their', 'on', 'ma', 'mustn', 'hadn', 'than', 'its', "shan't", 'wouldn', "aren't", "you'd", 'can', 'for', 'where', 'through', 'above', 'off', 'll', 'of', 'both', 'me', 've', 'to', 'herself', 'y', 'have', 'these', 'aren', 'don', 'out', 'up', 'it', 'doing', 's', 'didn', 'why', "haven't", 'own', 'yours', 't', "won't", 'whom', 'do', 'ome', 'nor', 'then', 'having', 'here', 'o', 'theirs', 'most', 'or', 'as', 'while', 'weren't', 'once', 'hasn't', "you've", 'should', 'has', "she's", 'will', 'her', 'are', 'further', 'after', 'isn', 'very', 'them', 'had', 'now', 'his', 'between', 'what', 'i', 'hasn', 'needn', 'you', 'only', 'which', 'm', 'she', 'shouldn', 'd', 'themselves', 'did', 'few', 'needn', 'other', 'couldn', 'below', 'myself', 'was', 'such', 'they', 'a', 'too', 'the', 'being', 'over', 'but', "you're", 'again', 'during', 'so', "wouldn", 'is', "didn't", 'if', "wasn't", 're', 'hers', 'until', 'wasn', 'couldn', 'yourselves', 'our', 'all', 'by', 'from'}

In [32]:

```

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")

```

7.0601589 seconds

In [33]:

```
data.head()
```

Out[33]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	featherlite ladies long sleeve stain resistant...	\$26.26
6	B012YX2ZPI	HX-Kingdom	Fashion T-	White	SHIRT	womens unique 100 cotton special olympics	\$9.99

	asin	shirts	brand	color	amazon.com/images_medium_image_url	product_type_name	special olympics wor...	title	formatted_price
15	B003BSRPB0	FeatherLite	White		https://images-na.ssl-images-amazon.com/images...	SHIRT	featherlite ladies moisture free mesh sport sh...		\$20.54
27	B014ICEJ1Q	FNC7C	Purple		https://images-na.ssl-images-amazon.com/images...	SHIRT	supernatural chibis sam dean castiel neck tshi...		\$7.39
46	B01NACPBG2	Fifth Degree	Black		https://images-na.ssl-images-amazon.com/images...	SHIRT	fifth degree womens gold foil graphic tees jun...		\$6.95

In [0]:

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

Stemming

In [34]:

```
from nltk.stem.porter import *
stemmer = PorterStemmer()
print(stemmer.stem('arguing'))
print(stemmer.stem('fishing'))

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

argu
fish

[8] Text based product similarity

In [35]:

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

Out [35]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	featherlite ladies long sleeve stain resistant...	\$26.26
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	womens unique 100 cotton special olympics wor...	\$9.99
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	featherlite ladies moisture free mesh sport sh...	\$20.54
27	B014ICEJ1Q	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	supernatural chibis sam dean castiel neck tshi...	\$7.39
46	B01NACPBG2	Fifth Degree	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	fifth degree womens gold foil graphic tees jun...	\$6.95

	asin	brand	color	amazon.com/images/ medium_image_url	product_type_name	graphic tees jun... title	formatted_price
--	------	-------	-------	--	-------------------	---------------------------------	-----------------

In [36]:

```
# 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 occurrence of the word
    # url : apparel's url
    # labels: len(labels) == len(keys), the values of labels depends on the model we 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))

    # 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, plotting heat map that represents the count of commonly occurred 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 the 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 difference in
    # heatmap
    for i in vec2:
        if i not in intersection:
            vec2[i]=0

    # for labeling heatmap, keys contains list of all words in title2
    keys = list(vec2.keys())
```

```

    # if ith word in intersection(list 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 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))

if model == 'bag_of_words':
    labels = values
elif model == 'tfidf':
    labels = []
    for x in vec2.keys():
        # tfidf_title_vectorizer.vocabulary_ it contains all the words in the corpus
        # tfidf_title_features[doc_id, 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.vocabulary_[x]])
        else:
            labels.append(0)
elif model == 'idf':
    labels = []
    for x in vec2.keys():
        # idf_title_vectorizer.vocabulary_ it contains all the words in the corpus
        # idf_title_features[doc_id, index_of_word_in_corpus] will give the idf value of word in given document (doc_id)
        if x in idf_title_vectorizer.vocabulary_:
            labels.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[x]])
        else:
            labels.append(0)

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

# this function gets a list of words 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 occurrence of each word in list, it returns dict type object {word1:count}

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

    # vector1 = dict{word1:#count, word2:#count, etc.}
    vector1 = text_to_vector(text1)

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

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

```

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

In [37]:

```

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 sparse 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 occurred in that doc

```

Out[37]:

(16042, 12609)

In [39]:

```
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 apparels
    # the metric we used here is cosine, the coside distance is mesured as  $K(X, Y) = \langle X, Y \rangle / (\|X\| * \|Y\|)$ 
    # 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_indices[i]], data['medium_image_url'].loc[df_indices[0]], '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)

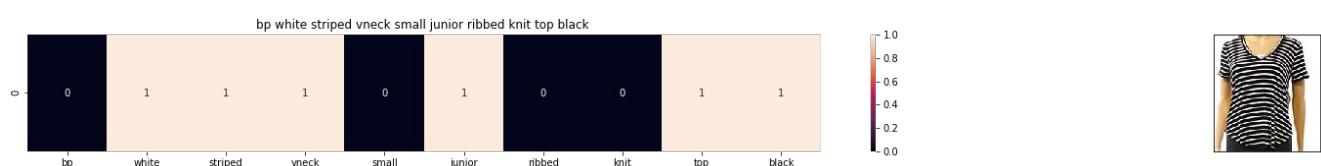
    #call the bag-of-words model for a product to get similar products.
    bag_of_words_model(1345, 20) # change the index if you want to.
    # In the output heat map each value represents the count value
    # of the label word, the color represents the intersection
    # with inputs title.

#try 12566
#try 931
```



ASIN : B01M3YMFF7
Brand: Lush Clothing
Title: lush white junior striped vneck belted tunic top black xs
Euclidean similarity with the query image : 0.0

=====



ASIN : B072PTJJ5Y
Brand: BP
Title: bp white striped vneck small junior ribbed knit top black
Euclidean similarity with the query image : 2.8284271247461903

=====



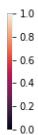
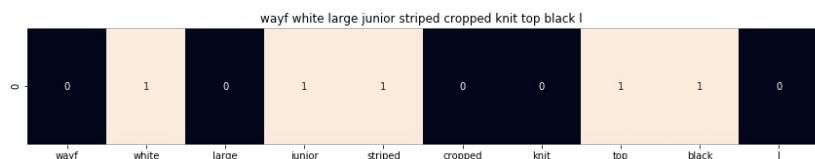


ASIN : B06XVYC2CV

Brand: Alexis

Title: alexis sienne striped top xs

Euclidean similarity with the query image : 3.0

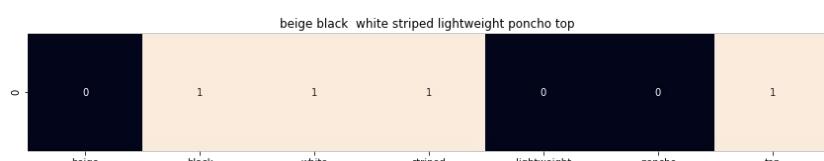


ASIN : B01HD7LHOO

Brand: WAYF

Title: wayf white large junior striped cropped knit top black l

Euclidean similarity with the query image : 3.0

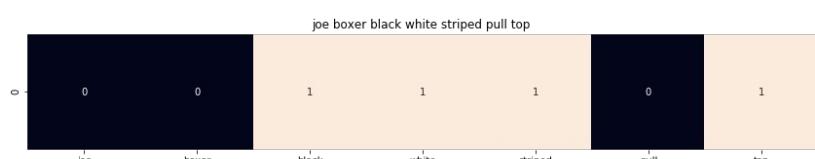


ASIN : B01EESHW3G

Brand: Luxury Divas

Title: beige black white striped lightweight poncho top

Euclidean similarity with the query image : 3.0

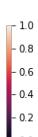
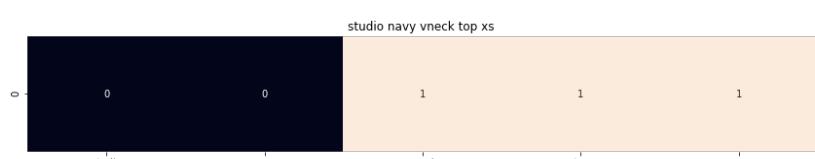


ASIN : B071JJPPGH

Brand: Joe Boxer

Title: joe boxer black white striped pull top

Euclidean similarity with the query image : 3.0

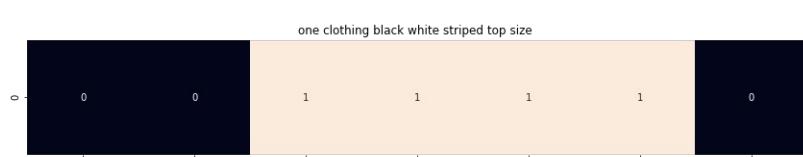


ASIN : B01J504G7I

Brand: Studio M

Title: studio navy vneck top xs

Euclidean similarity with the query image : 3.0

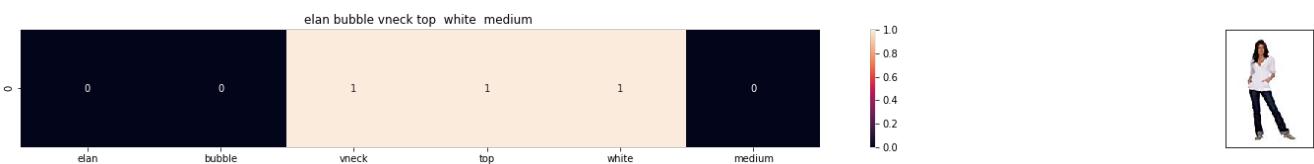


ASIN : B06XQ4KLQL

Brand: One Clothing

Title: one clothing black white striped top size

Euclidean similarity with the query image : 3.0

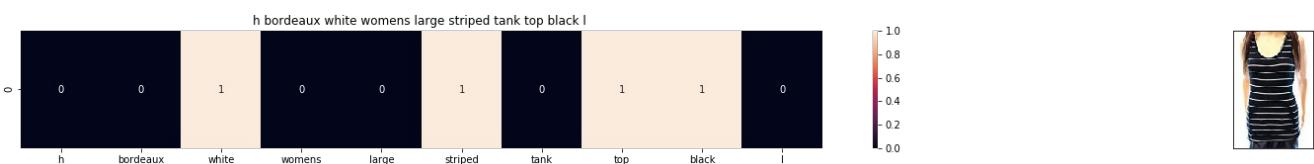


ASIN : B00CF4GDUA

Brand: ELAN

Title: elan bubble vneck top white medium

Euclidean similarity with the query image : 3.1622776601683795

=====

ASIN : B01MSH5UCH

Brand: H By Bordeaux

Title: h bordeaux white womens large striped tank top black l

Euclidean similarity with the query image : 3.1622776601683795

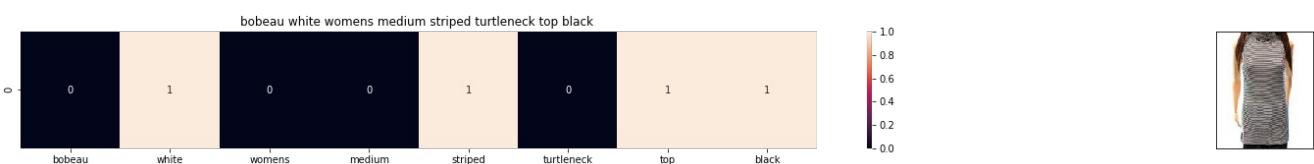
=====

ASIN : B00JPOZ9GM

Brand: Sofra

Title: womens tank top white

Euclidean similarity with the query image : 3.1622776601683795

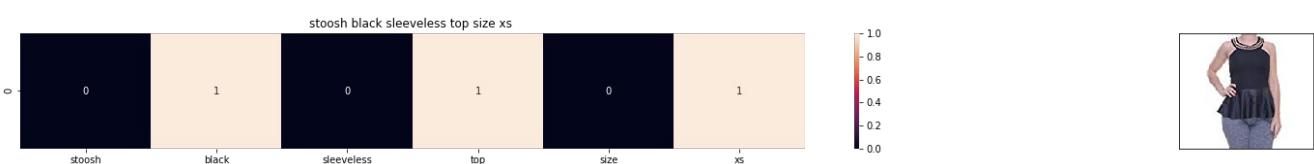
=====

ASIN : B07336B8WY

Brand: Bobeau

Title: bobeau white womens medium striped turtleneck top black

Euclidean similarity with the query image : 3.1622776601683795

=====

ASIN : B06XGPGWM8

Brand: Stoosh

Title: stoosh black sleeveless top size xs

Euclidean similarity with the query image : 3.1622776601683795

=====

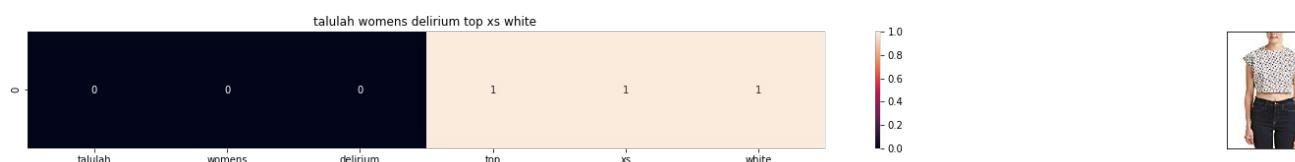


ASIN : B072N8CS7N

Brand: A.L.C.

Title: alc womens mirella top xs black

Euclidean similarity with the query image : 3.1622776601683795

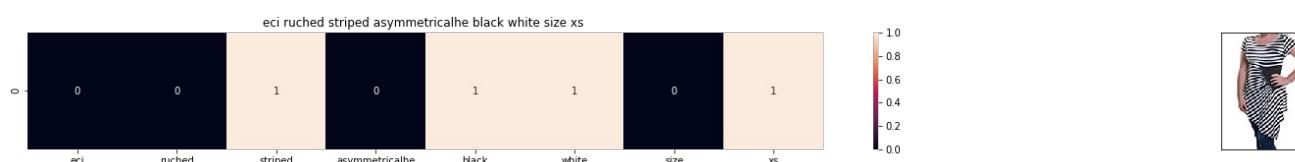


ASIN : B071KG15NM

Brand: Talulah

Title: talulah womens delirium top xs white

Euclidean similarity with the query image : 3.1622776601683795



ASIN : B015OVZIIS

Brand: ECI

Title: eci ruched striped asymmetricalhe black white size xs

Euclidean similarity with the query image : 3.1622776601683795

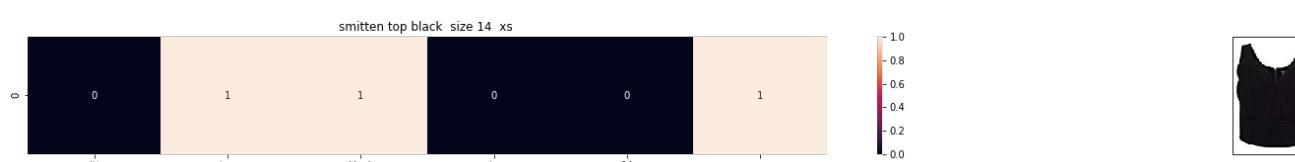


ASIN : B074TSFJHM

Brand: M Missoni

Title: missoni womens top black

Euclidean similarity with the query image : 3.1622776601683795



ASIN : B01D0121J8

Brand: City Chic

Title: smitten top black size 14 xs

Euclidean similarity with the query image : 3.1622776601683795



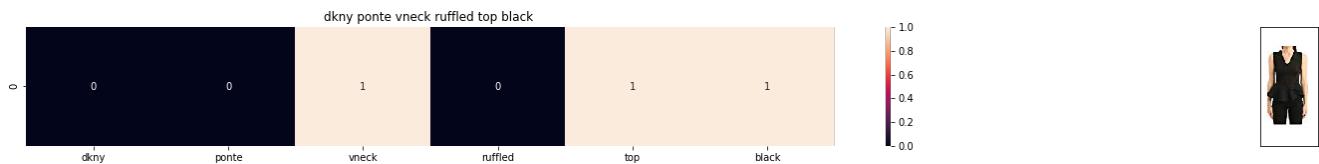
ASIN : B071HZXMF6

Brand: Brooks Brothers

Title: brooks brothers womens tunic xs white

Euclidean similarity with the query image : 3.1622776601683795

```
=====
```



ASIN : B06XH7HQ2T

Brand: DKNY

Title: dkny ponte vneck ruffled top black

Euclidean similarity with the query image : 3.1622776601683795

[8.5] TF-IDF based product similarity

In [40]:

```
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 given doc
```

In [41]:

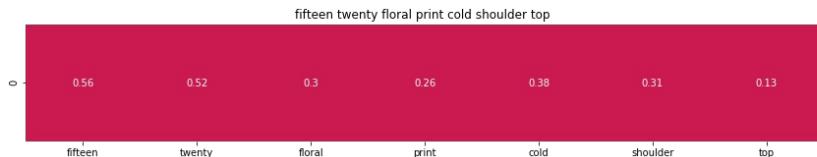
```
def tfidf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance from given input apparel to all remaining apparels
    # 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_indices[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(125, 20)
# in the output heat map each value represents the tfidf values of the label word, the color represents the intersection with inputs title
```



ASIN : B06XJ66M4P

BRAND : FIFTEEN TWENTY

Eucliden distance from the given image : 0.0

=====

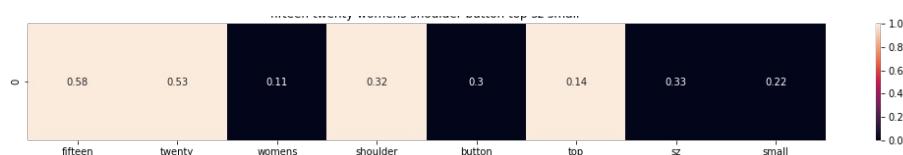
=====

=====

=====

=====

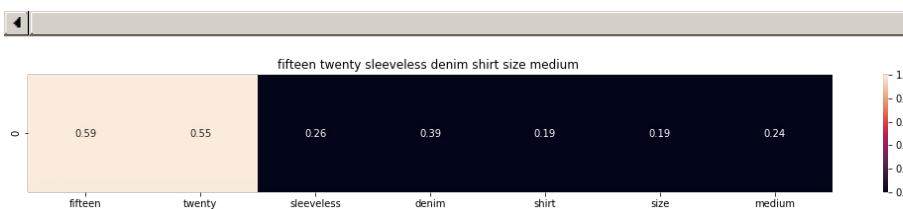
fifteen twenty womens shoulder button top sz small



ASIN : B074KT1R5Y

BRAND : FIFTEEN TWENTY

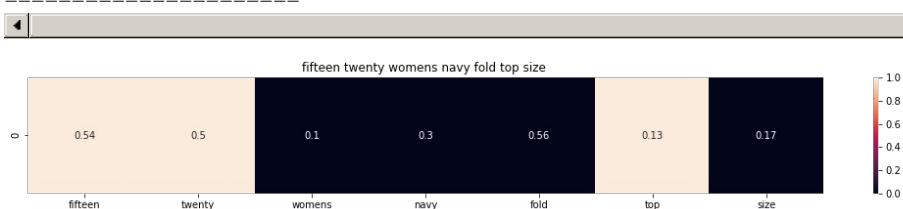
Euclidean distance from the given image : 0.7513763192537632



ASIN : B01IFYEVQG

BRAND : FIFTEEN TWENTY

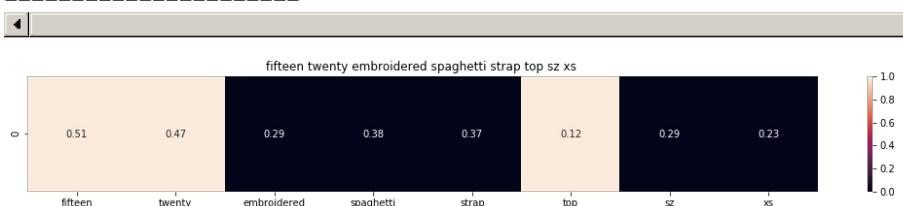
Euclidean distance from the given image : 0.8778660847234199



ASIN : B074KRLG5F

BRAND : FIFTEEN TWENTY

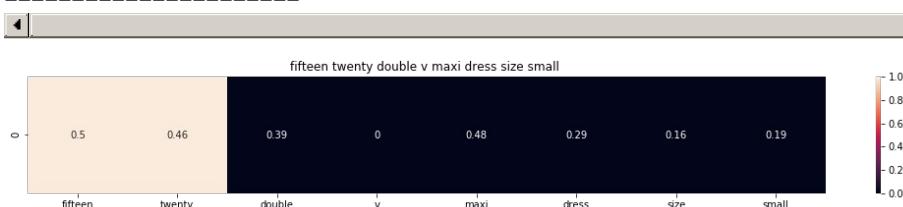
Euclidean distance from the given image : 0.9233000469813402



ASIN : B01I0NAIDM

BRAND : FIFTEEN TWENTY

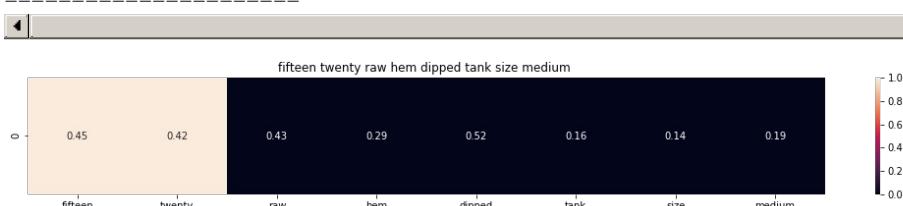
Euclidean distance from the given image : 0.9528092363412269



ASIN : B01IG4CBE4

BRAND : FIFTEEN TWENTY

Euclidean distance from the given image : 0.9793303963833676



ASIN : B01IG2V2LY

BRAND : FIFTEEN TWENTY

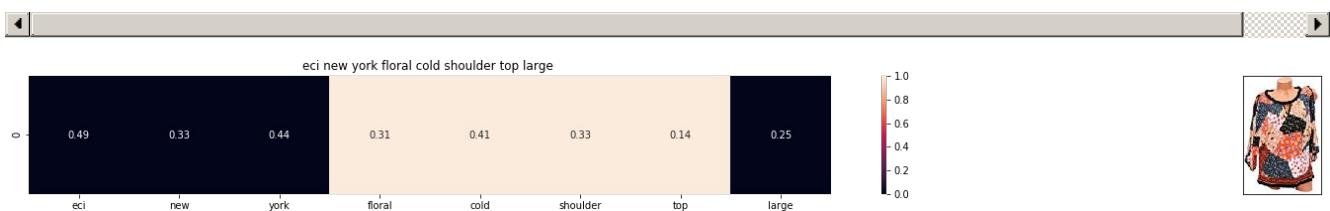
Euclidean distance from the given image : 1.0308282103366617



ASIN : B01KYMCMIG

BRAND : Hip

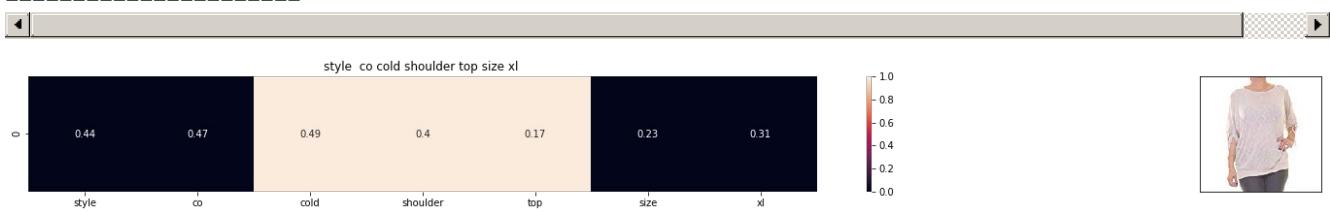
Euclidean distance from the given image : 1.1089736611777519



ASIN : B06XP9VLYL

BRAND : ECI New York

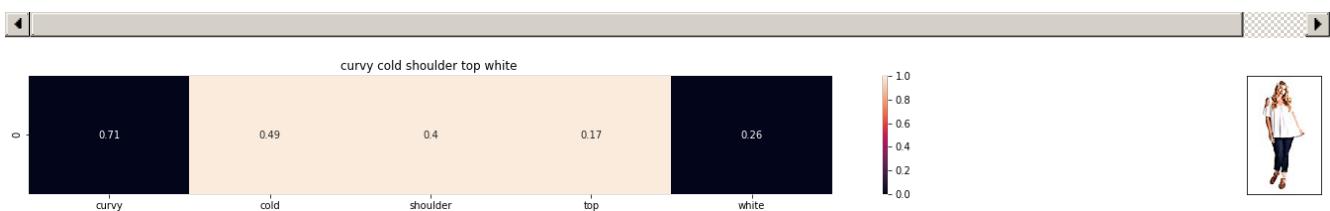
Euclidean distance from the given image : 1.1232720835087358



ASIN : B01N2HO9AV

BRAND : Style& Co.

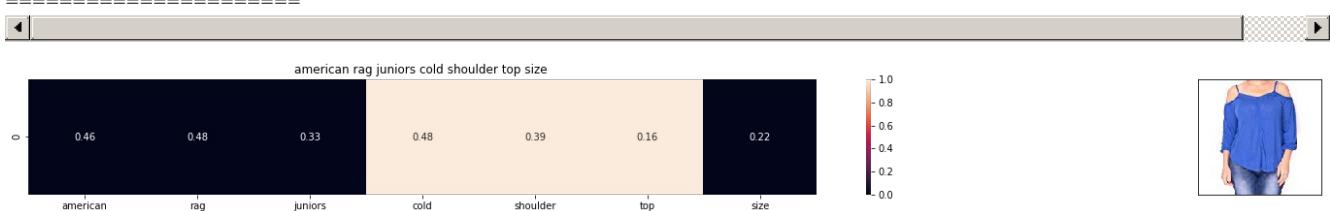
Euclidean distance from the given image : 1.1512363644335064



ASIN : B01ILGCQ9W

BRAND : Emerald

Euclidean distance from the given image : 1.1521612635840166

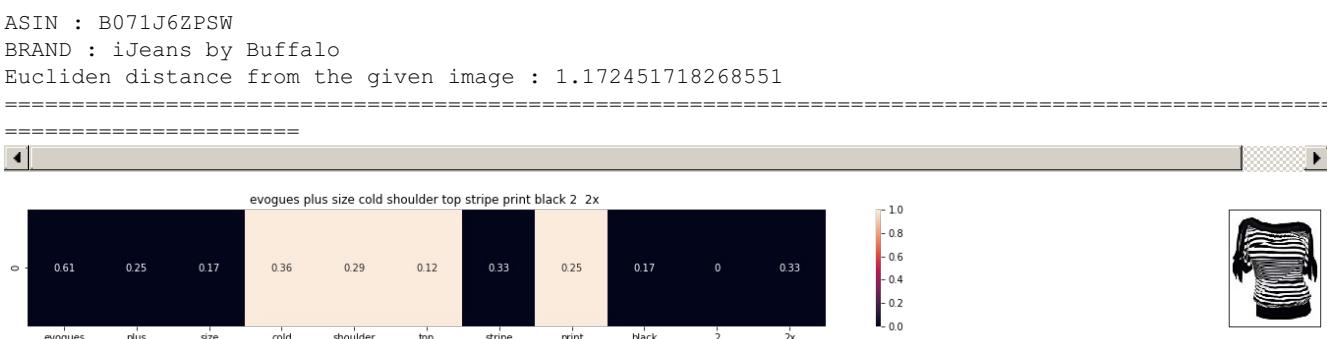


ASIN : B06XY6HK4Z

BRAND : American Rag

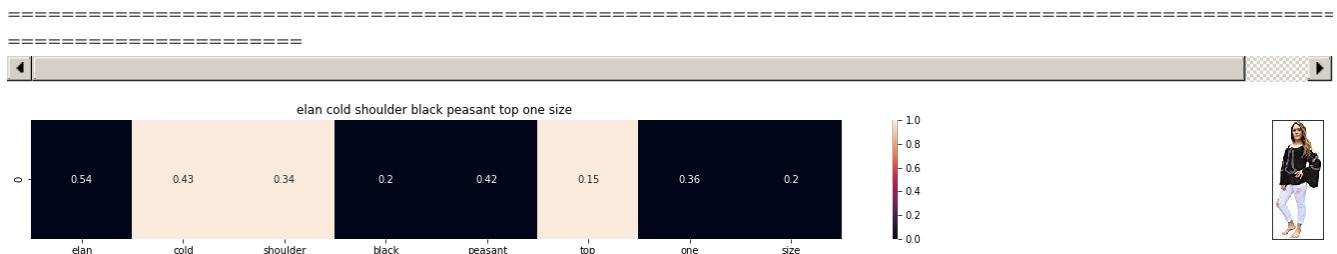
Euclidean distance from the given image : 1.1626150506396635





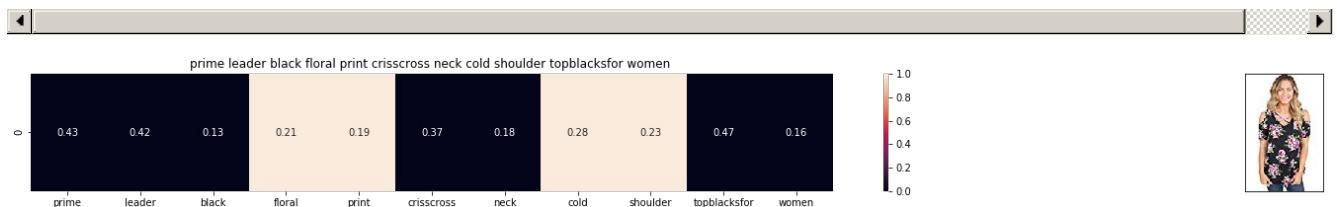
ASIN : B07174CJXJ
BRAND : MG Collection

Euclidean distance from the given image : 1.1915521786532



ASIN : B019R2XM6U
BRAND : ELAN

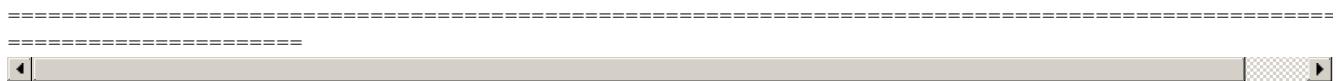
Euclidean distance from the given image : 1.1917139692531162



ASIN : B07313BLD2

BRAND : Prime Leader's Shirt

Euclidean distance from the given image : 1.1917291940339043



[8.5] IDF based product similarity

In [42]:

```
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 occured in that d  
oc
```

In [43]:

```
def nContaining(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] / (nContaining(word)))
```

In [44]:

```
# 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 t  
he word  
    # idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0] will return all doc  
ments 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_corpus] = idf value of word
idf_title_features[j,idf_title_vectorizer.vocabulary_[i]] = idf_val
```

In [45]:

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

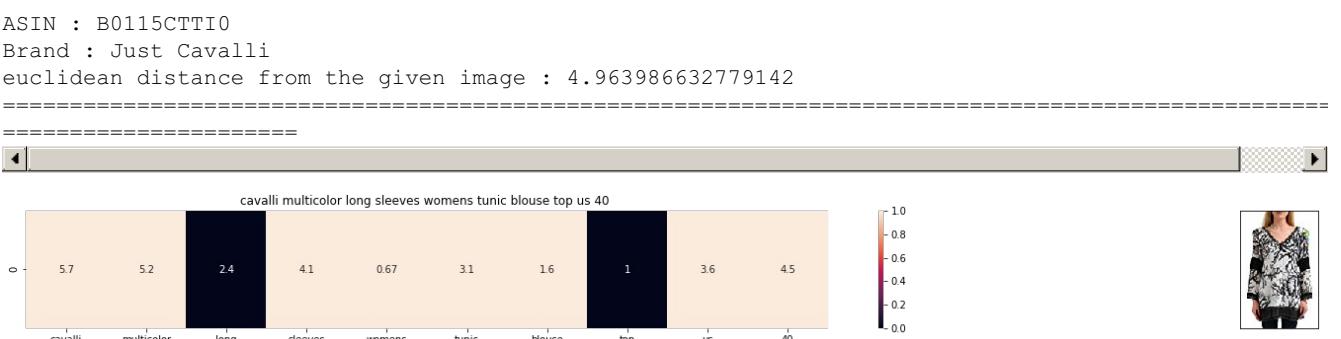
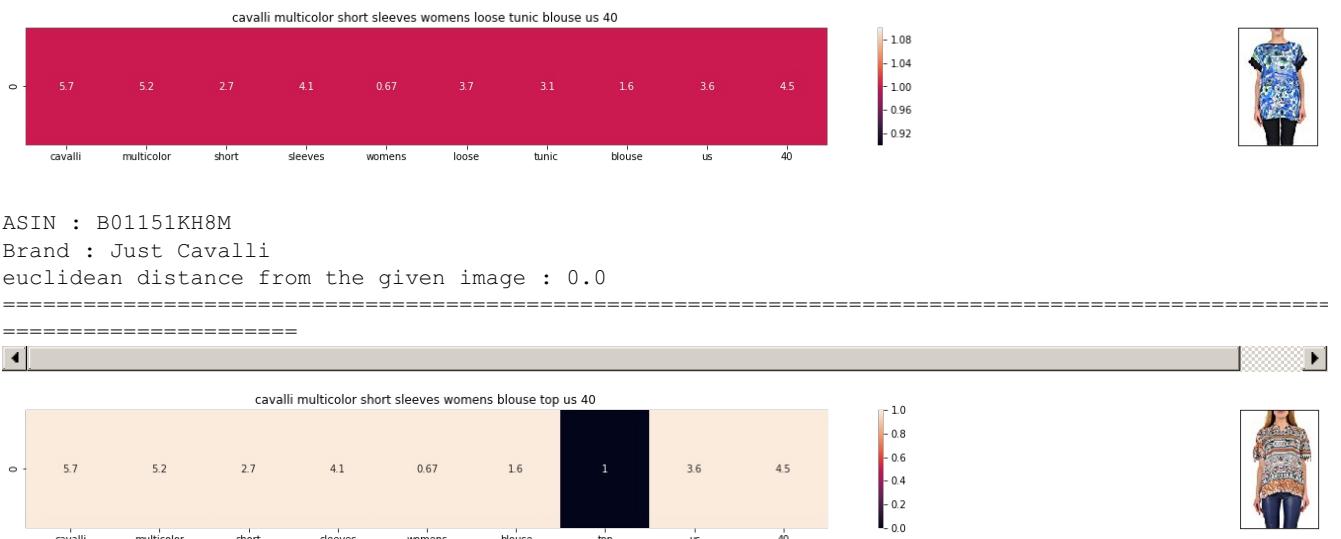
    # pairwise_dist will store the distance from given input apparel to all remaining apparels
    # the metric we used here is cosine, the coside distance is mesured as  $K(X, Y) = \langle X, Y \rangle / (\|X\| * \|Y\|)$ 
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = 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_indices[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)

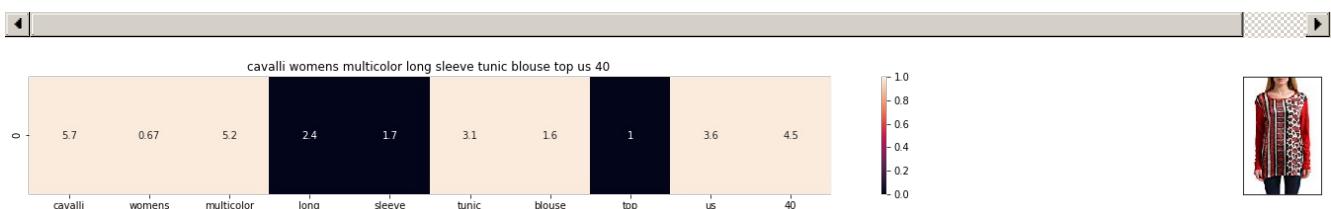
idf_model(15000,20)
# in the output heat map each value represents the idf values of the label word, the color represents the intersection with inputs title
```



ASIN : B01HWH2F42
Brand : Just Cavalli

euclidean distance from the given image : 5.280156135277855

=====

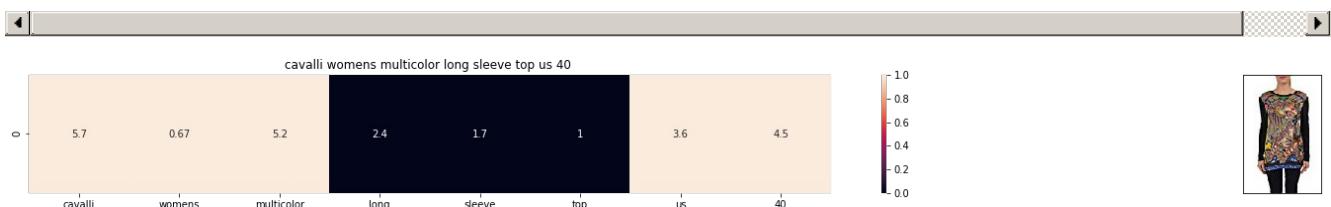


ASIN : B01CASX59M

Brand : Just Cavalli

euclidean distance from the given image : 6.883919903555224

=====

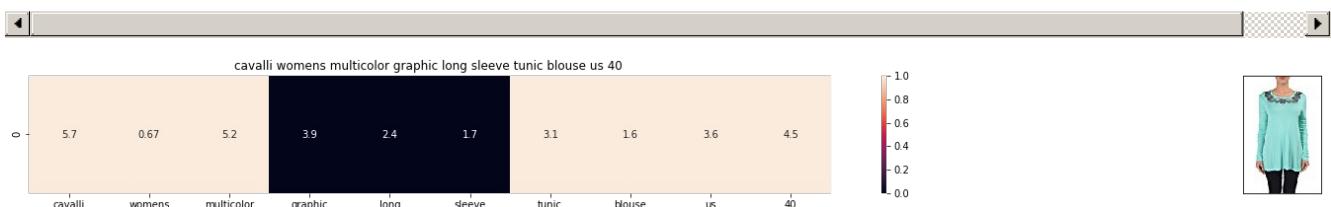


ASIN : B0175BBXY8

Brand : Just Cavalli

euclidean distance from the given image : 7.7350103284861

=====

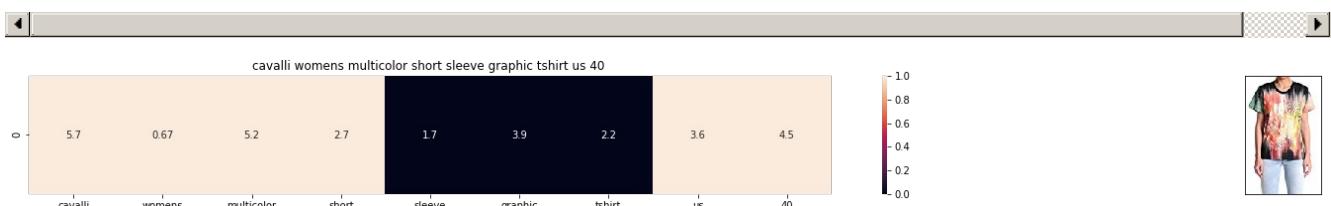


ASIN : B017ADYVZO

Brand : Just Cavalli

euclidean distance from the given image : 7.842854818326502

=====

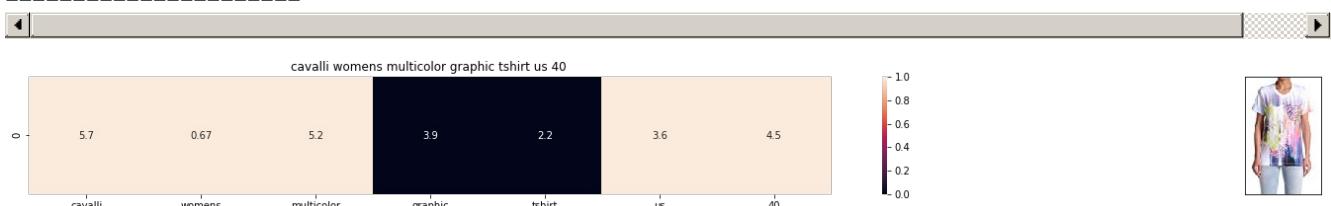


ASIN : B01FYHEEVY

Brand : Just Cavalli

euclidean distance from the given image : 8.113134916053644

=====

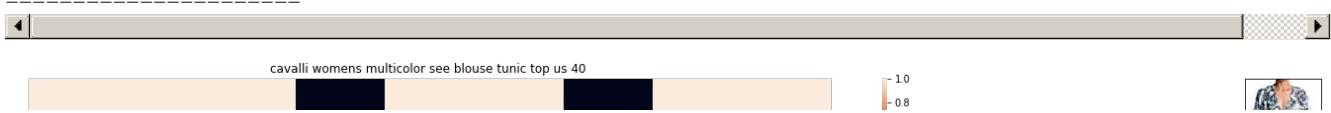


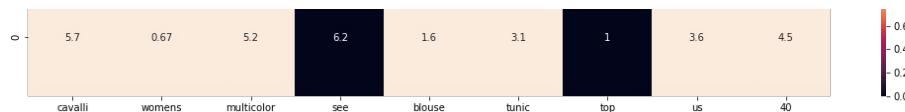
ASIN : B01FZUNBVO

Brand : Just Cavalli

euclidean distance from the given image : 8.37798404689231

=====

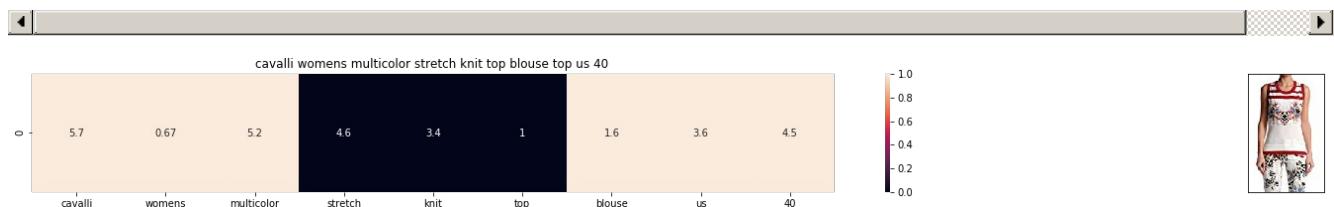




ASIN : B01FZTUK8M

Brand : Just Cavalli

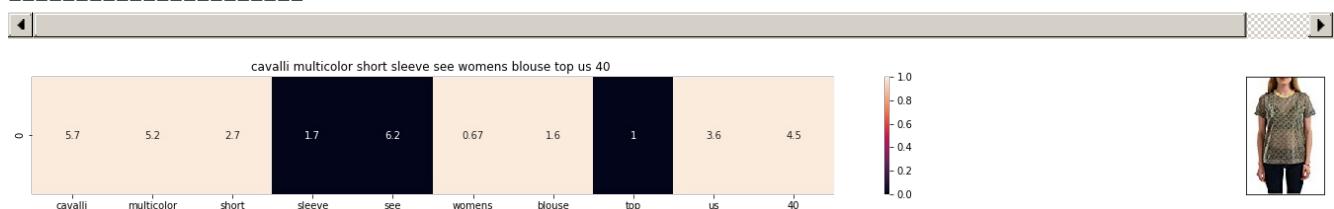
euclidean distance from the given image : 8.755383388349836



ASIN : B01FZ5QZ12

Brand : Just Cavalli

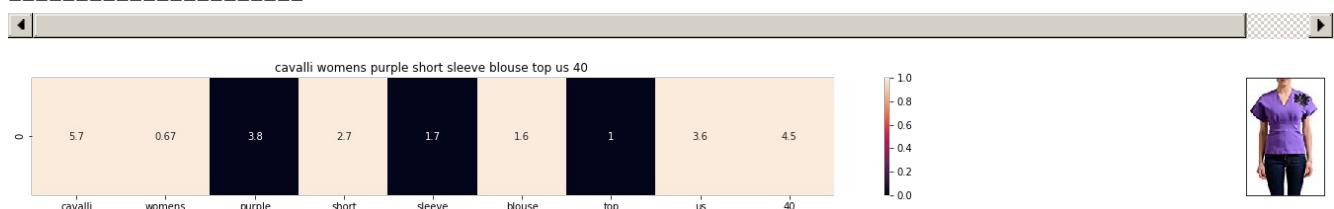
euclidean distance from the given image : 9.016413504062283



ASIN : B01HSKDJXY

Brand : Just Cavalli

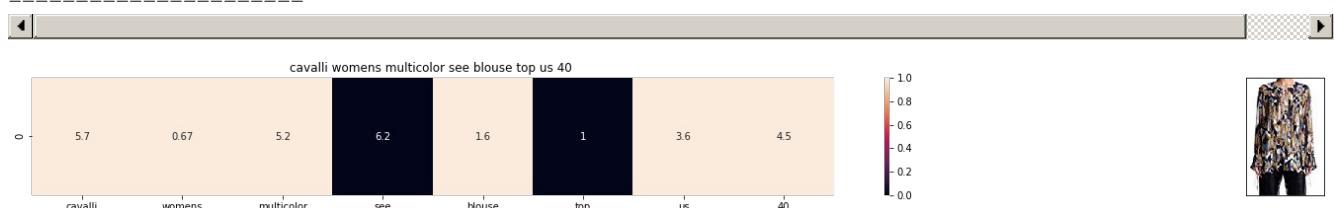
euclidean distance from the given image : 9.058325082569132



ASIN : B01CAU74LK

Brand : Just Cavalli

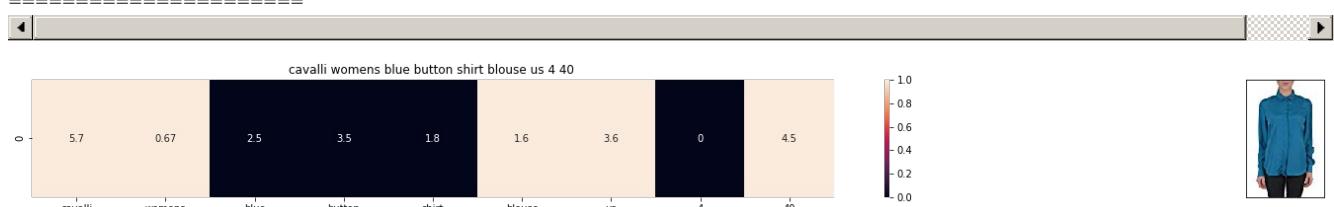
euclidean distance from the given image : 9.21328314563803



ASIN : B0180UM25M

Brand : Just Cavalli

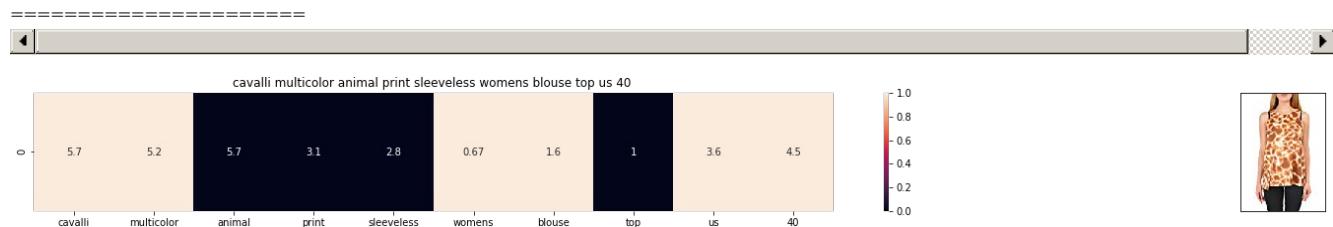
euclidean distance from the given image : 9.296284839945168



ASIN : B00TQ6OEMY

Brand : Just Cavalli

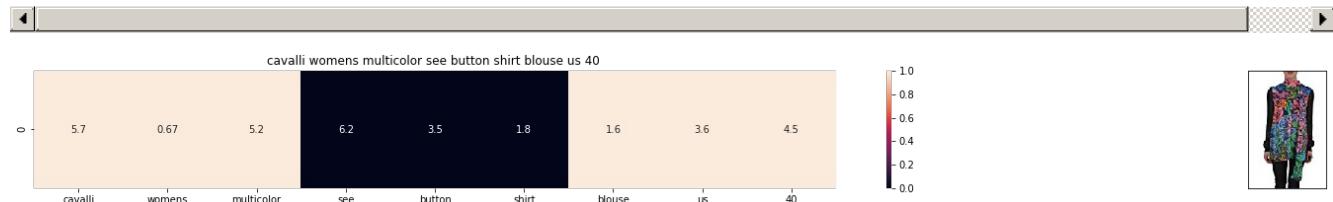
euclidean distance from the given image : 9.772884432394639



ASIN : B011AJXWA4

Brand : Just Cavalli

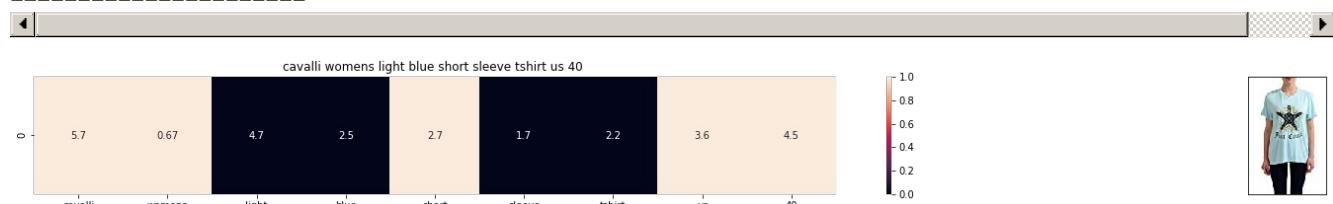
euclidean distance from the given image : 9.884336164635727



ASIN : B00TQ2LM1E

Brand : Just Cavalli

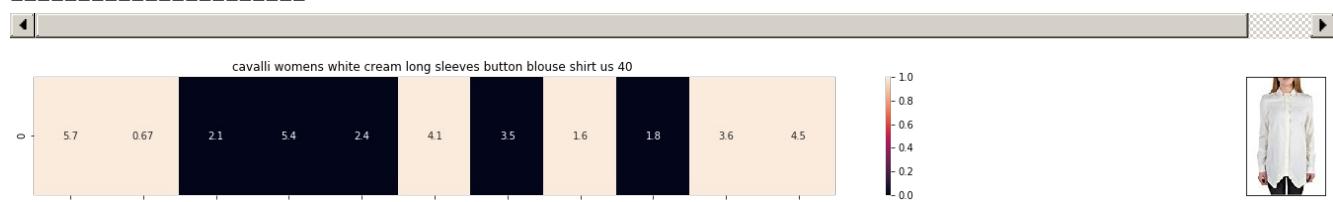
euclidean distance from the given image : 10.033144635314798



ASTN : B01GU7VHBM

ASIN : B01G0VHBM
Brand : Just Cavalli

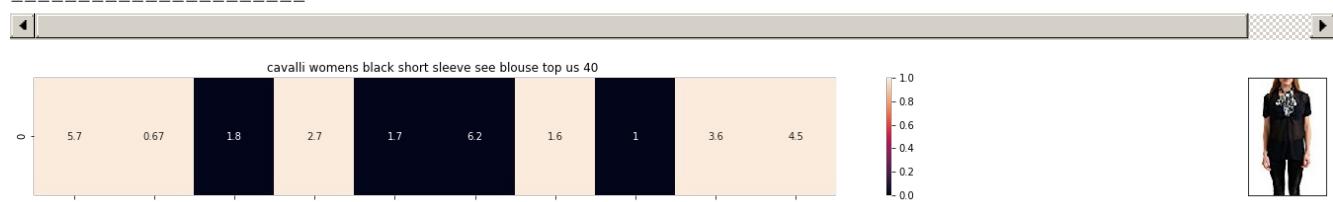
euclidean distance from the given image : 10.272163591301123



ASTN : B00117114NAG

ASIN : B000704NAG
Brand : Just Cavalli

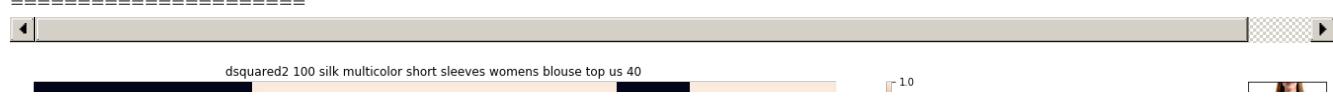
euclidean distance from the given image : 10.565876357293474

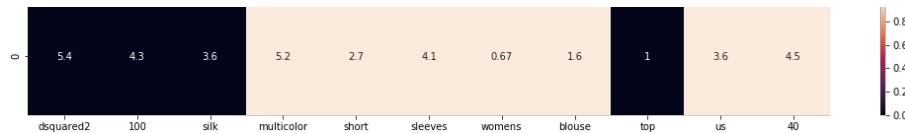


ASTIN : B01CATZDWY

ASIN : B01C4T2DWY
Brand : Just Cavalli

Brand : Just Cavalli
euclidean distance from the given image : 10.579983539012916





ASIN : B01D6F5CDK
 Brand : DSQUARED2
 euclidean distance from the given image : 10.875897624323407

[]

[9] Text Semantics based product similarity

In [0]:

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

'''
# Set values for various parameters
num_features = 300      # Word vector dimensionality
min_word_count = 1        # Minimum word count
num_workers = 4            # Number of threads to run in parallel
context = 10               # Context window size

downsampling = 1e-3       # Downsample setting for frequent words

# Initialize and train the model (this will take some time)
from gensim.models import word2vec
print ("Training model...")
model = word2vec.Word2Vec(sen_corpus, workers=num_workers, \
                           size=num_features, min_count = min_word_count, \
                           window = context)

'''
```

In [46]:

```
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 which contains a dict ,
# and it contains all our corpus 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/0B7XkCwpI5KDYNlNUTtSS21pQmM/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 [47]:

```
# 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 corpus 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 =
len(w2v_model[word])
# each row represents the word2vec representation of each word (weighted/avg) in given
sentence
return np.array(vec)

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

    final_dist = []
    # for each vector in vec1 we caluclate the distance(euclidean) to all vectors in vec2
    for i in vec1:
        dist = []
        for j in vec2:
            # np.linalg.norm(i-j) will result 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/avg) of lengt
h 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/avg) of lengt
h 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 image of appare
1
    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 [48]:

```

# 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 sentence
def build_avg_vec(sentence, num_features, doc_id, m_name):
    # sentence: its title of the apparel
    # num_features: the length 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 initialize a vector of size 300 with all zeros
# we add each word2vec(wordi) to this featureVec
nwords = 0

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

```

[9.2] Average Word2Vec product similarity.

In [49]:

```

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 [50]:

```

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 distance'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('PRODUCT :',data['product'].loc[df_indices[i]])

```

```

print('BRAND :',data['brand'].loc[ar_inaices[i]])
print ('euclidean distance from given input image :', pdists[i])
print('*'*125)

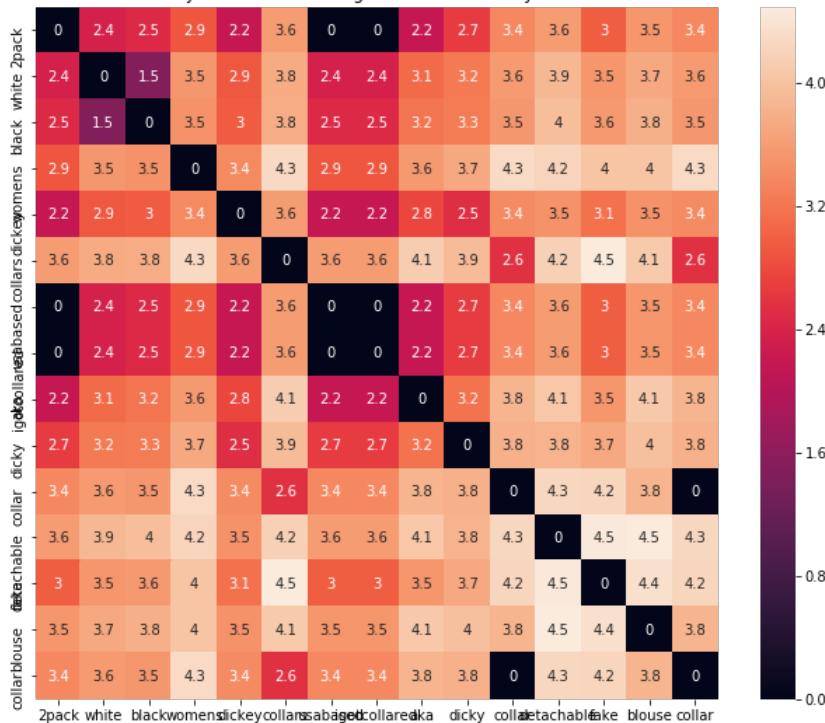
```

```

avg_w2v_model(5166, 20)
# in the give heat map, each cell contains the euclidean distance between words i, j

```

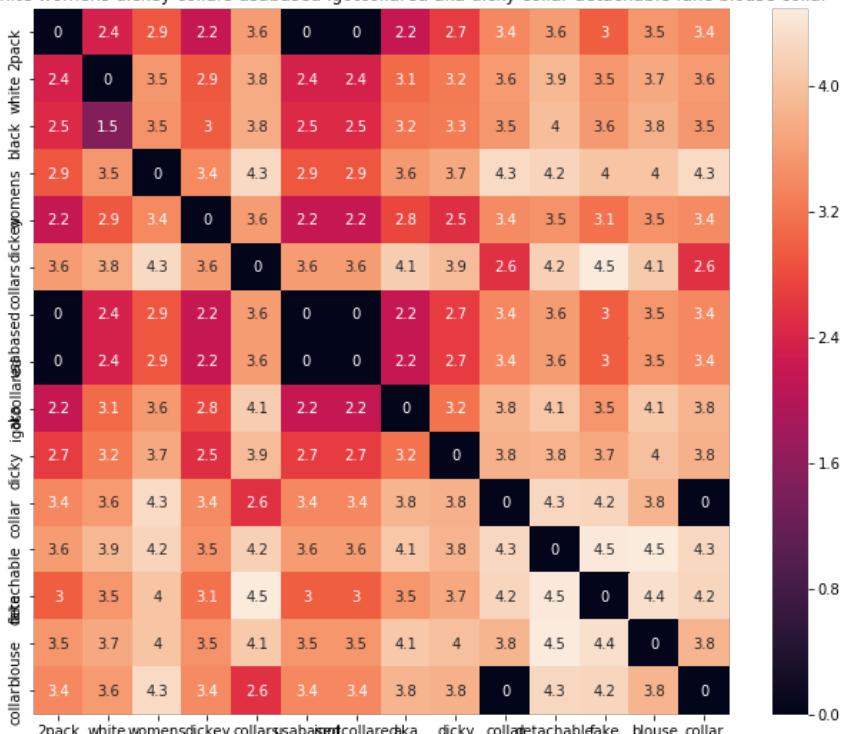
2pack white black womens dickey collars usabased igotcollared aka dicky collar detachable fake blouse collar



ASIN : B01N4GCBUM

BRAND : IGotCollared Pronounced I Got Collared
euclidean distance from given input image : 0.0

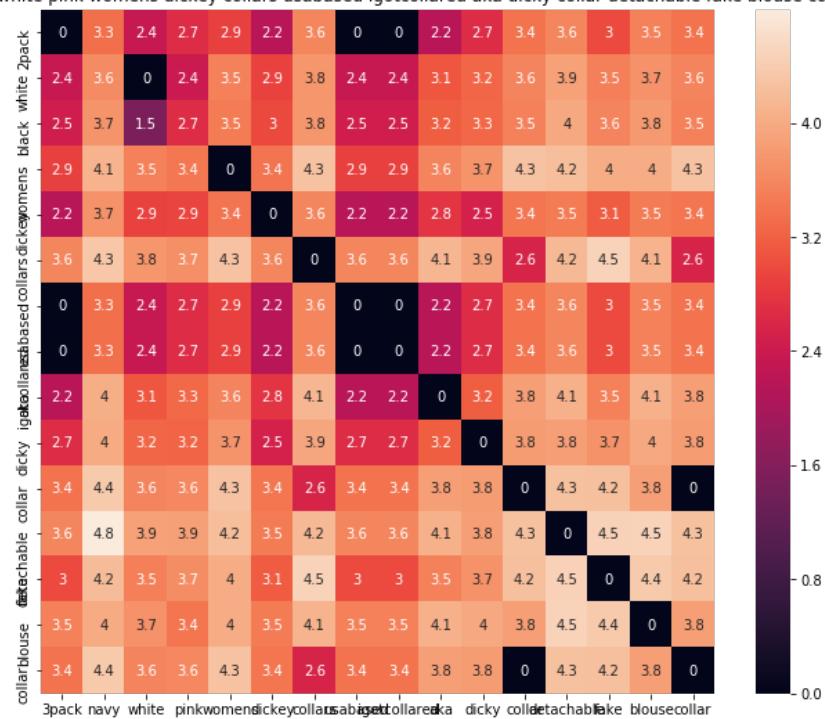
2pack white womens dickey collars usabased igotcollared aka dicky collar detachable fake blouse collar



ASIN : B01N9JYI2N

BRAND : IGotCollared Pronounced I Got Collared
euclidean distance from given input image : 0.14638343

3pack navy white pink womens dickey collars usabased igotcollared aka dicky collar detachable fake blouse collar

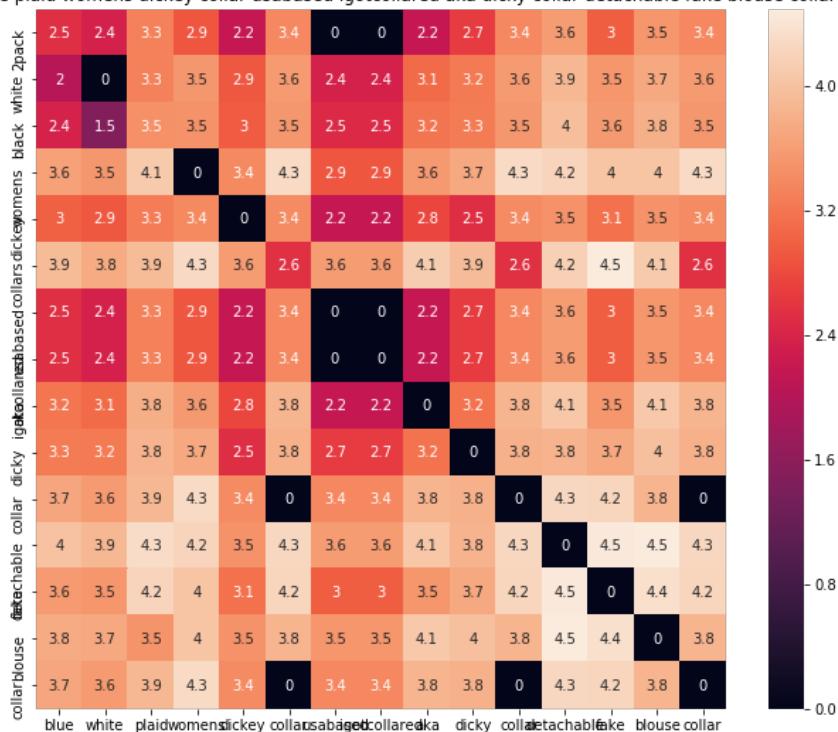


ASIN : B01MZ24OV9

BRAND : IGotCollared Pronounced I Got Collared

euclidean distance from given input image : 0.26626623

blue white plaid womens dickey collar usabased igotcollared aka dicky collar detachable fake blouse collar



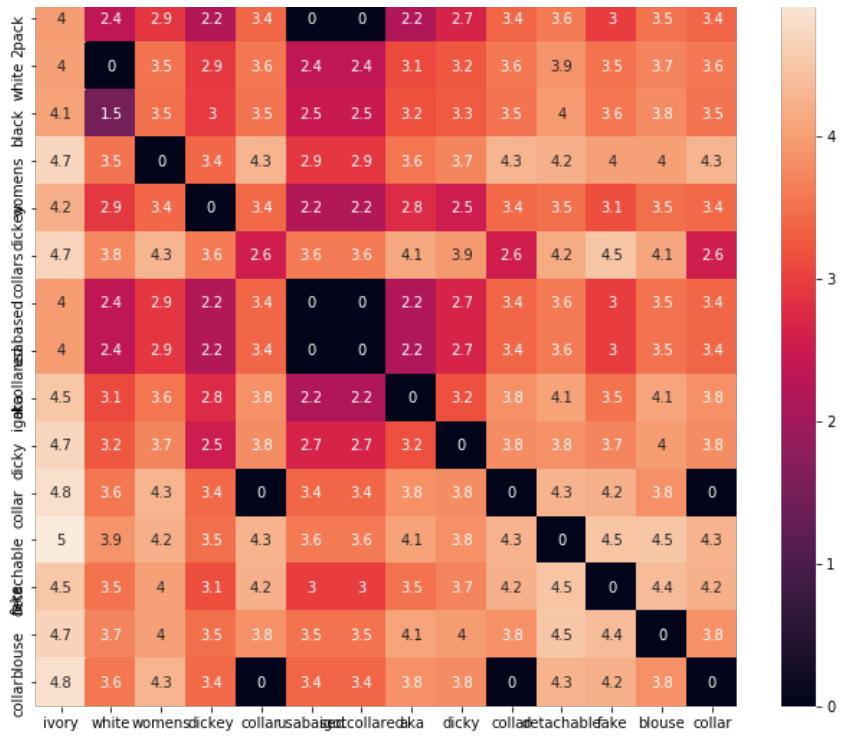
ASIN : B01MRXXYMA

BRAND : IGotCollared

euclidean distance from given input image : 0.32614702

ivory white womens dickey collar usabased igotcollared aka dicky collar detachable fake blouse collar



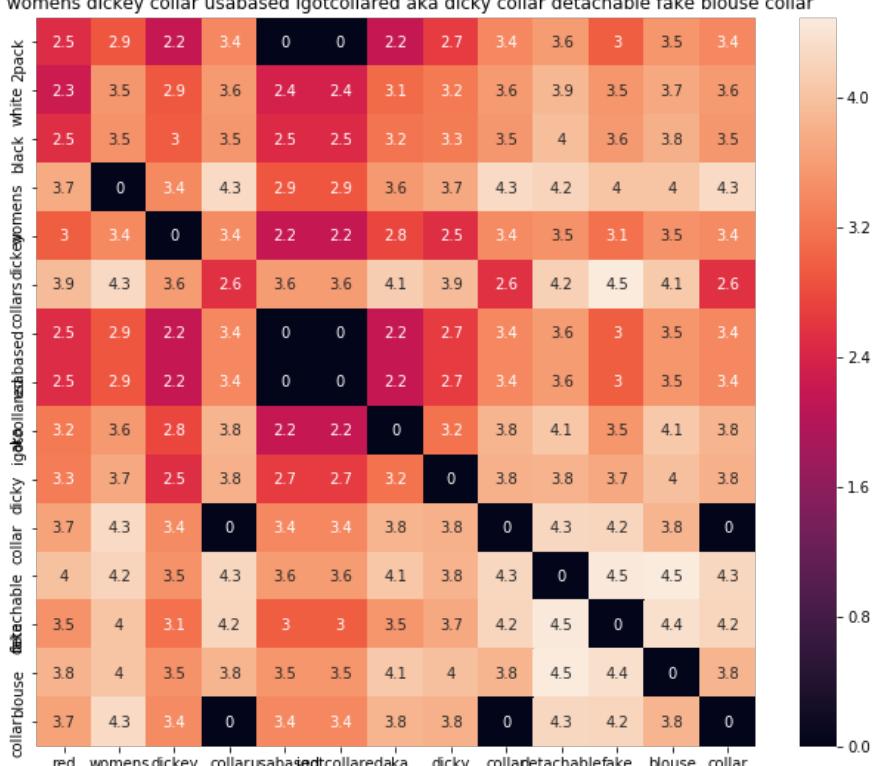


ASIN : B01MRUN3UC

BRAND : IGotCollared

euclidean distance from given input image : 0.32946828

red women's dickey collar usabased igotcollared aka dicky collar detachable fake blouse collar



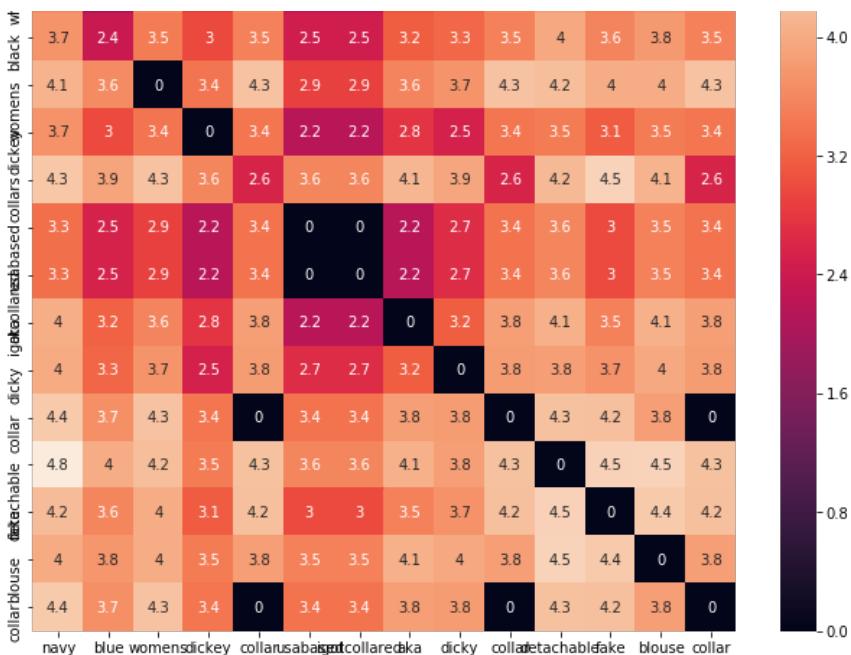
ASIN : B01MQWLG2I

BRAND : IGotCollared

euclidean distance from given input image : 0.34613767

navy blue women's dickey collar usabased igotcollared aka dicky collar detachable fake blouse collar



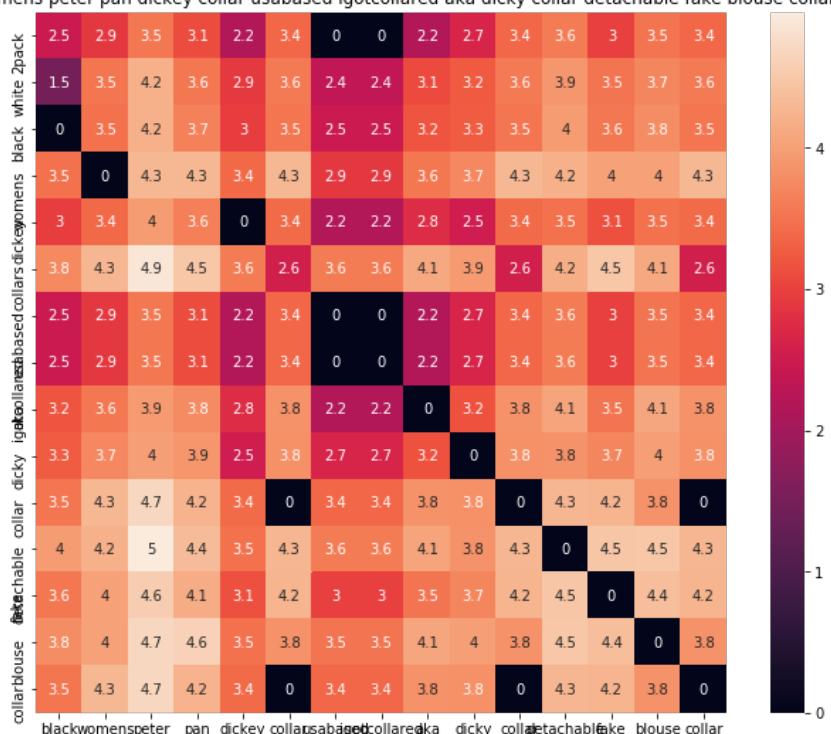


ASIN : B01NGYF4IJ

BRAND : IGotCollared

euclidean distance from given input image : 0.37557733

black womens peter pan dickey collar usabased igotcollared aka dicky collar detachable fake blouse collar

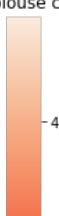
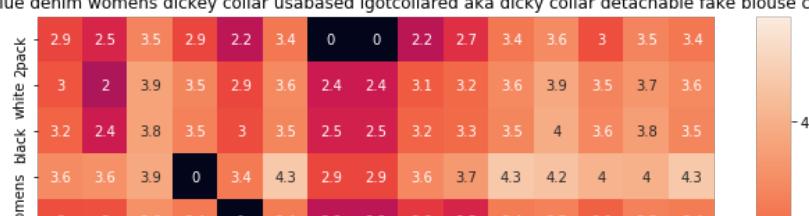


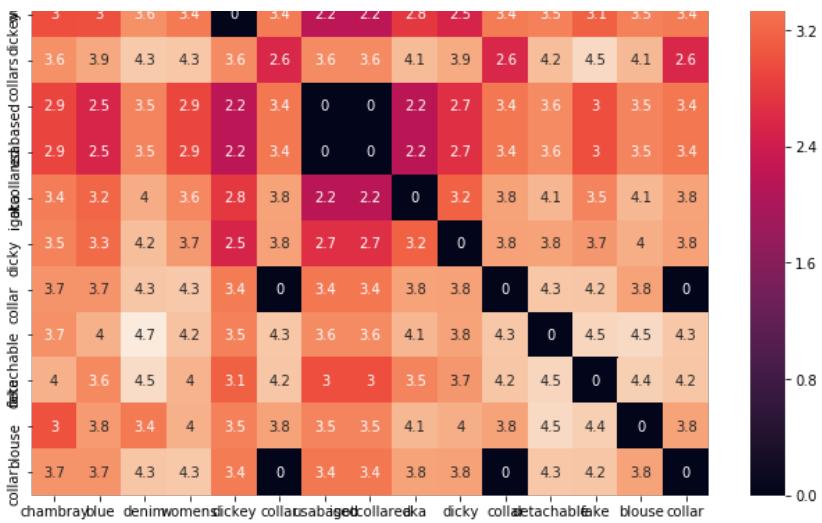
ASIN : B01MQWMK92

BRAND : IGotCollared

euclidean distance from given input image : 0.40530652

chambray blue denim womens dickey collar usabased igotcollared aka dicky collar detachable fake blouse collar



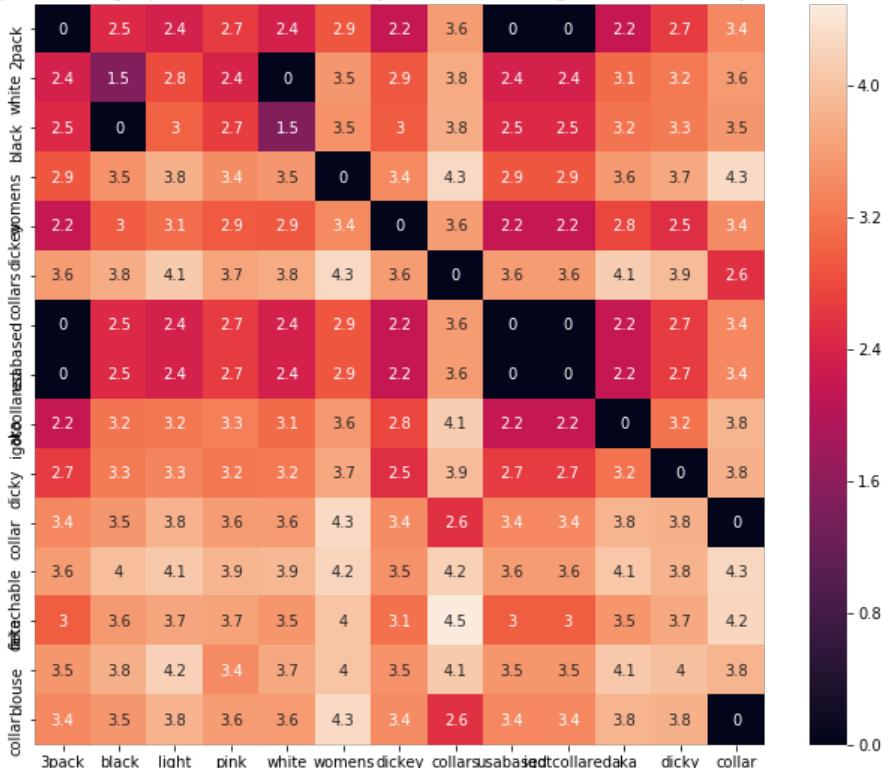


ASIN : B01MQWKWME

BRAND : IGotCollared

euclidean distance from given input image : 0.44265327

3pack black light pink white womens dickey collars usabased igotcollared aka dicky collar



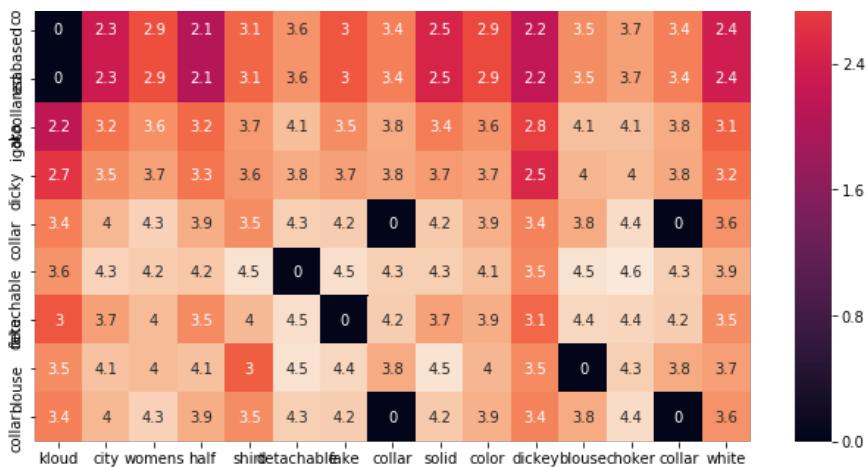
ASIN : B01N2W9P7A

BRAND : IGotCollared Pronounced I Got Collared

euclidean distance from given input image : 0.47885767

kloud city womens half shirt detachable fake collar solid color dickey blouse choker collar white



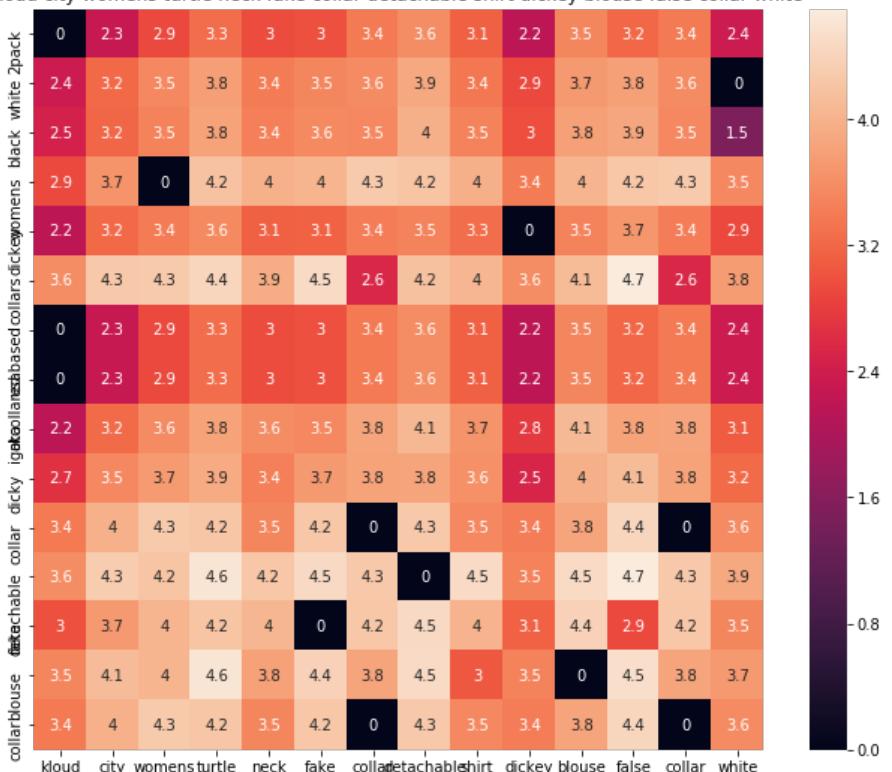


ASIN : B01MU2K4EU

BRAND : KLOUD City

euclidean distance from given input image : 0.5530486

kloud city womens turtle neck fake collar detachable shirt dickey blouse false collar white

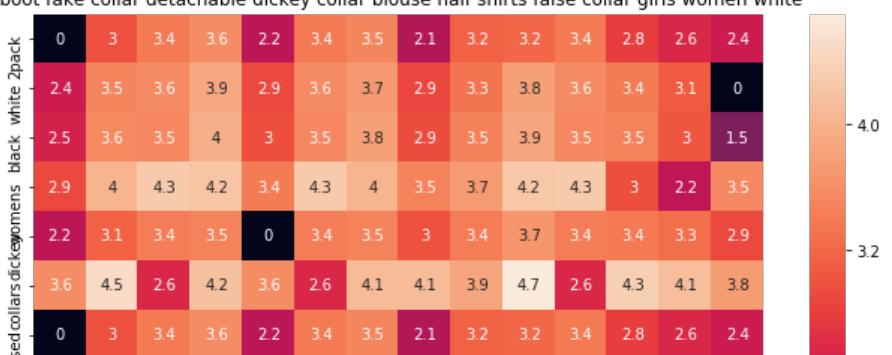


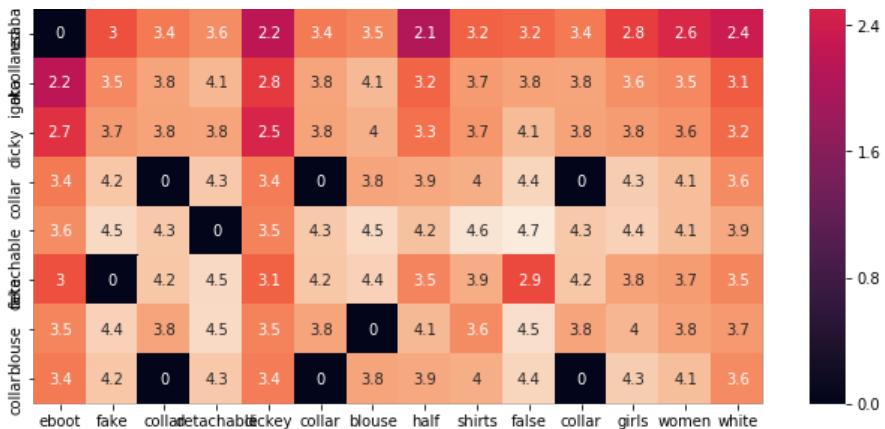
ASIN : B01N7IVVAJ

BRAND : KLOUD City

euclidean distance from given input image : 0.5642932

eboot fake collar detachable dickey collar blouse half shirts false collar girls women white



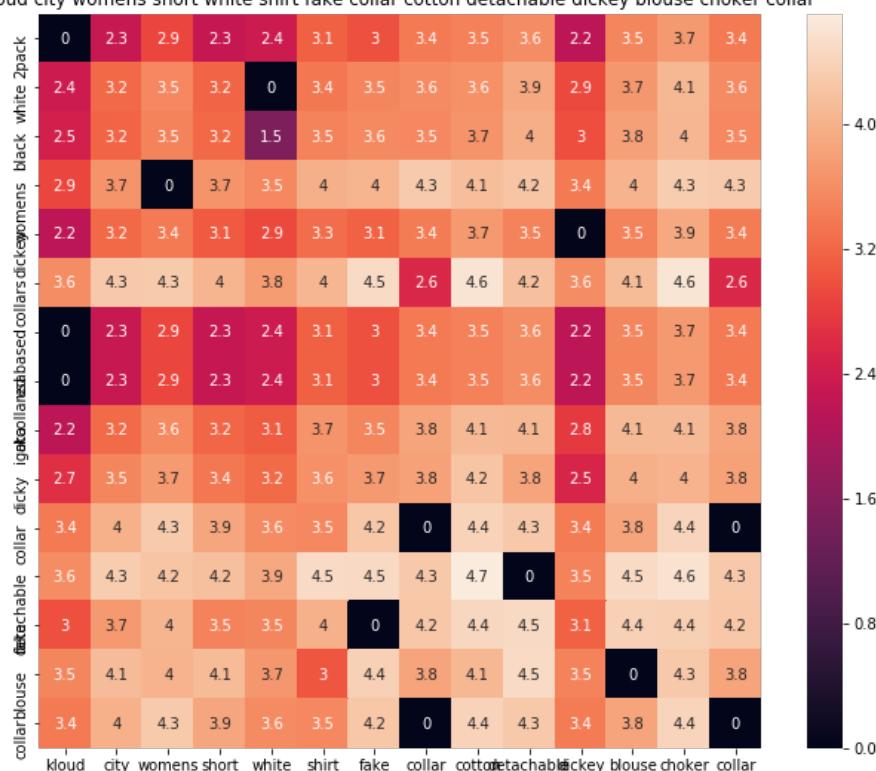


ASIN : B01N230VOY

BRAND : eBoot

euclidean distance from given input image : 0.5687726

kloud city womens short white shirt fake collar cotton detachable dickey blouse choker collar



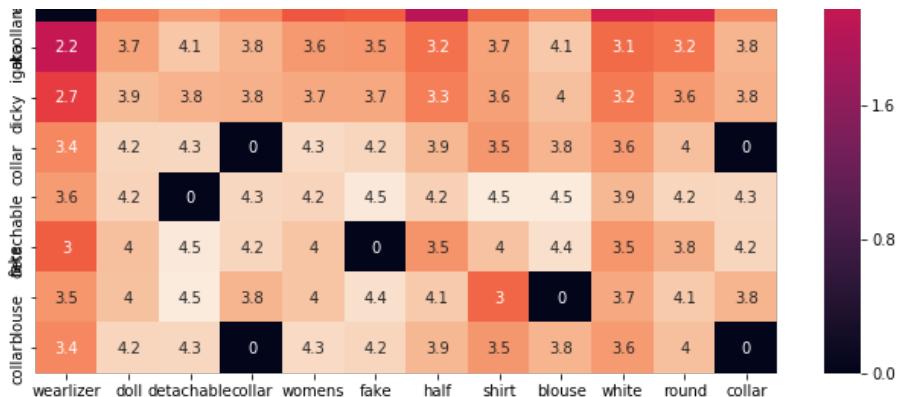
ASIN : B01MZ10MK9

BRAND : KLOUD City

euclidean distance from given input image : 0.57407403

wearlizer doll detachable collar womens fake half shirt blouse white round collar



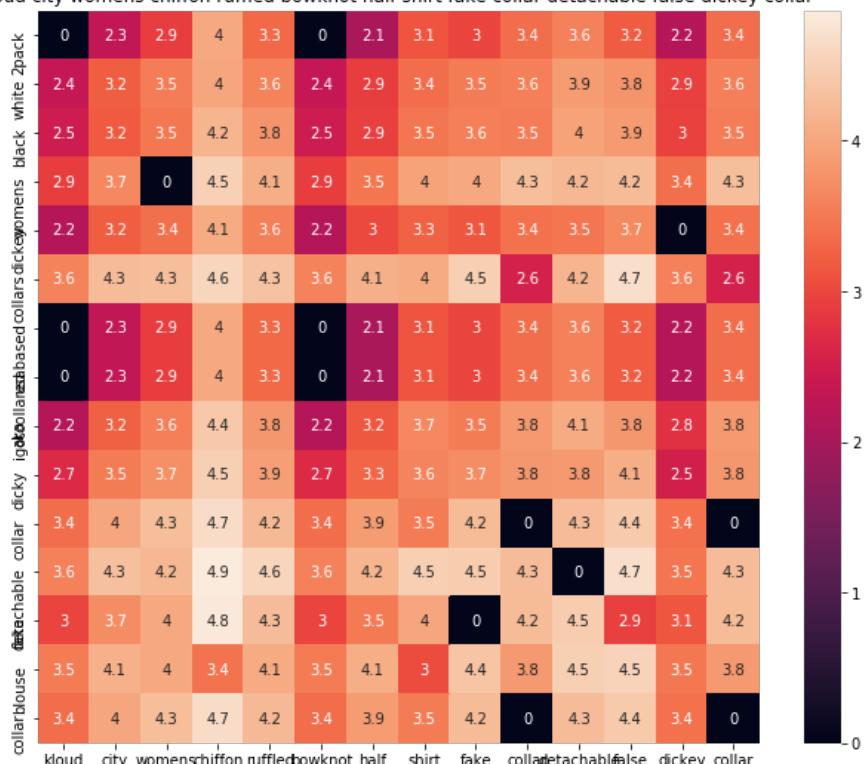


ASIN : B018HQVFFM

BRAND : Wearlizer

euclidean distance from given input image : 0.5997495

kloud city womens chiffon ruffled bowknot half shirt fake collar detachable false dickey collar

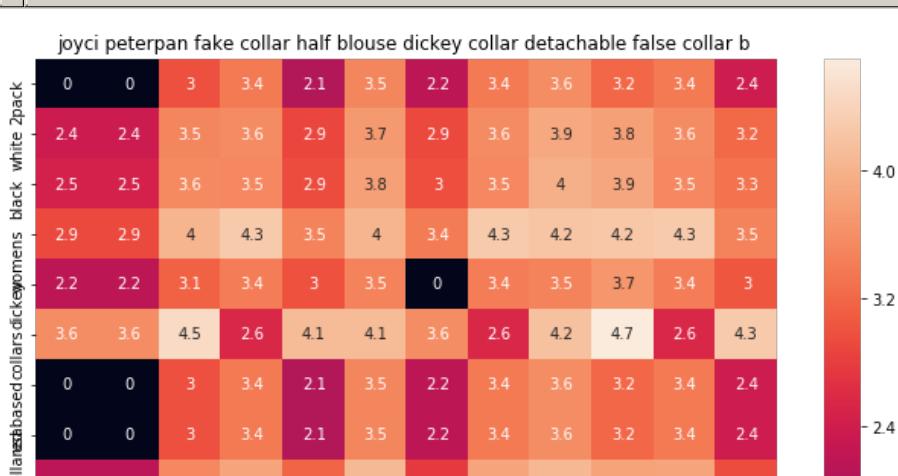


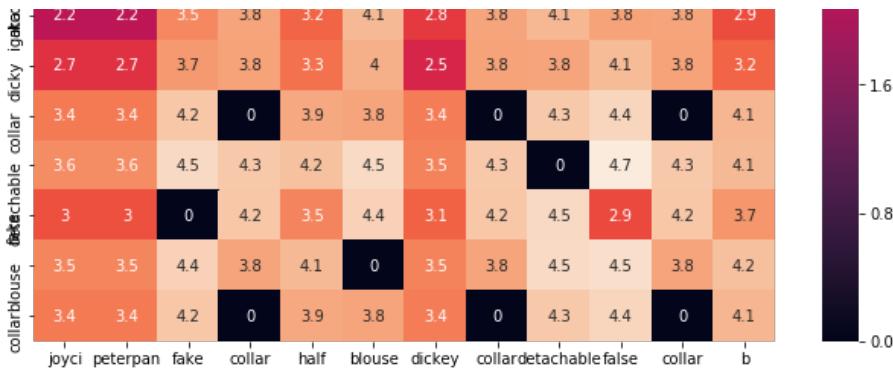
ASIN : B01N6IGKIE

BRAND : KLOUD City

euclidean distance from given input image : 0.6061027

joyci peterpan fake collar half blouse dickey collar detachable false collar b

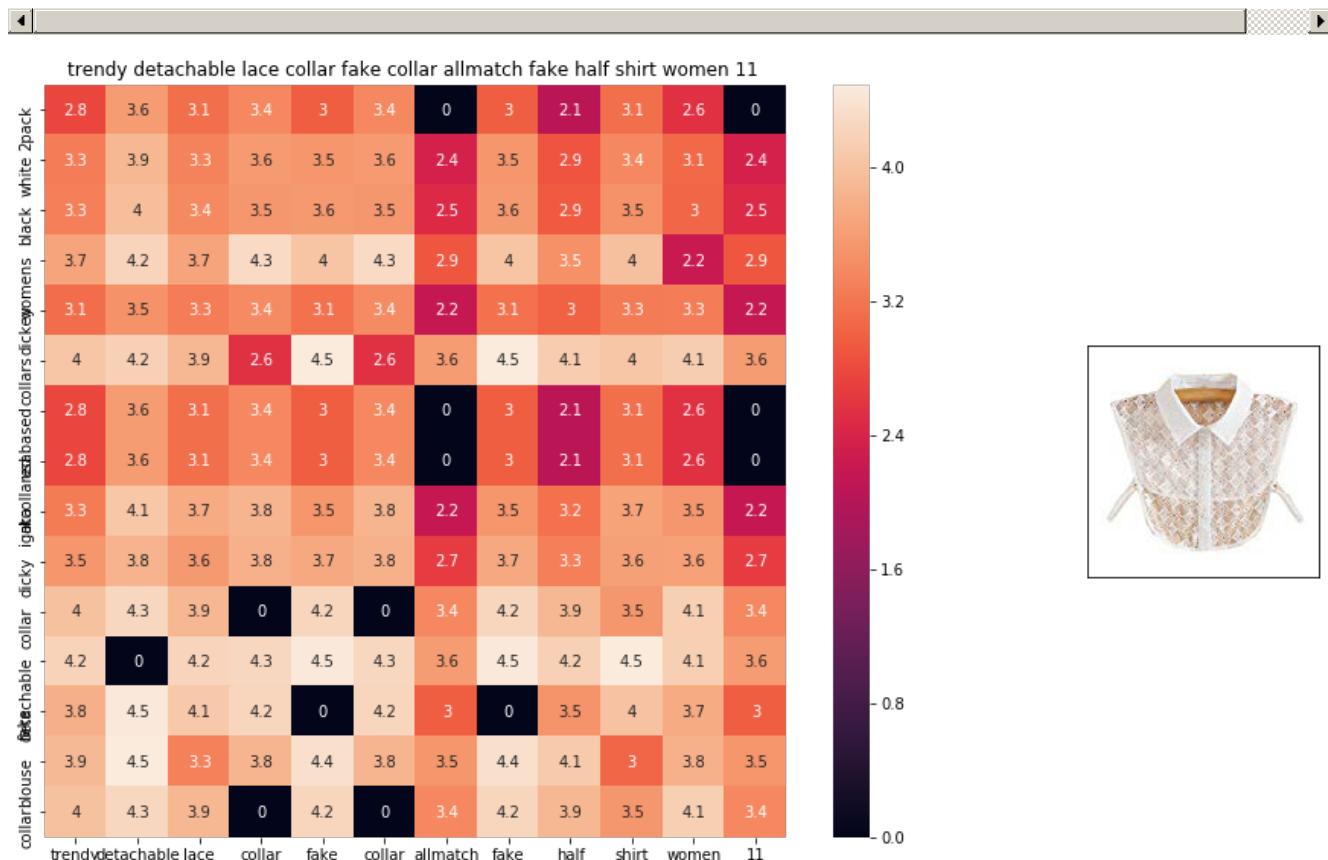




ASIN : B01M29GPK7

BRAND : Joyci

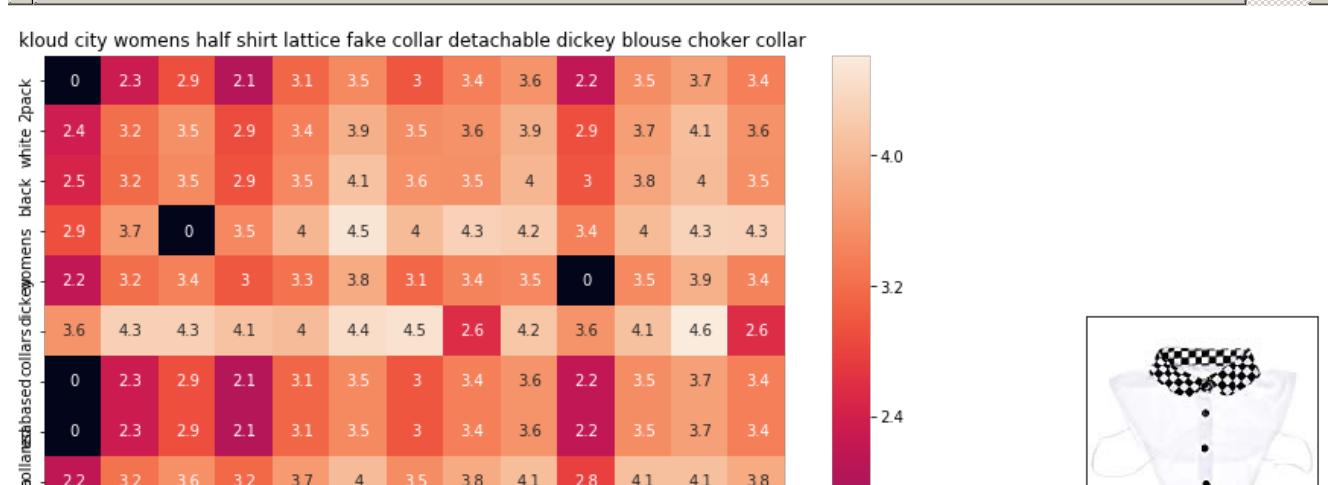
euclidean distance from given input image : 0.6093709

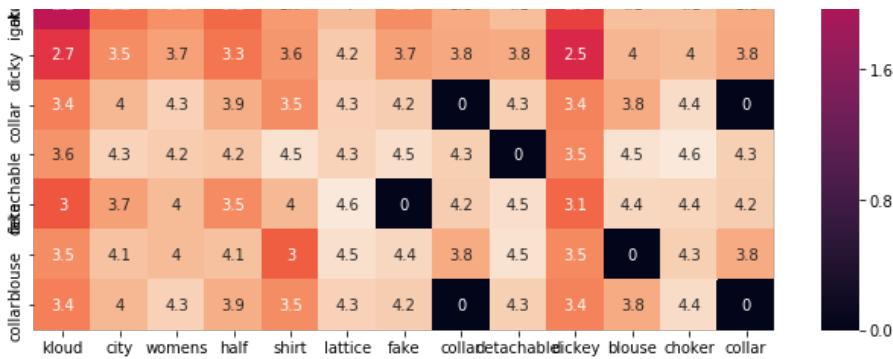


ASIN : B06X1G835Q

BRAND : George Jimmy

euclidean distance from given input image : 0.62557757



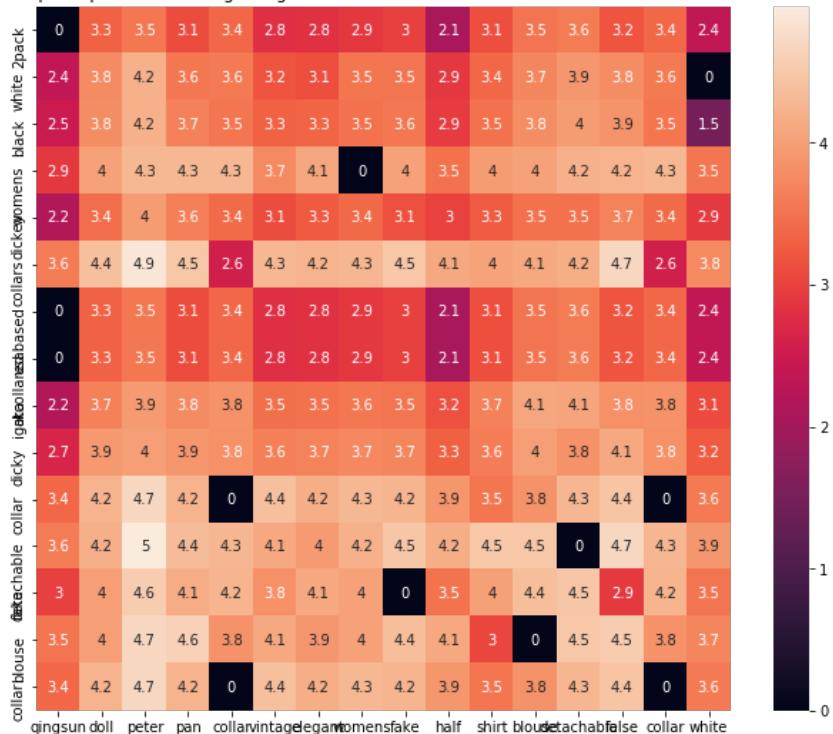


ASIN : B01MU29AKG

BRAND : KLOUD City

euclidean distance from given input image : 0.6342003

qingsun doll peter pan collar vintage elegant womens fake half shirt blouse detachable false collar white



ASIN : B06XMYD2Z1

BRAND : Qingsun

euclidean distance from given input image : 0.63936865

[9.4] IDF weighted Word2Vec for product similarity

In [51]:

```
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 [52]:

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

```

# pairwise_dist will store the distance from given input apparel to all remaining apparels
# the metric we used here is cosine, the coside distance is measured as  $K(X, Y) = \langle X, Y \rangle / (\|X\| * \|Y\|)$ 
# http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
pairwise_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))

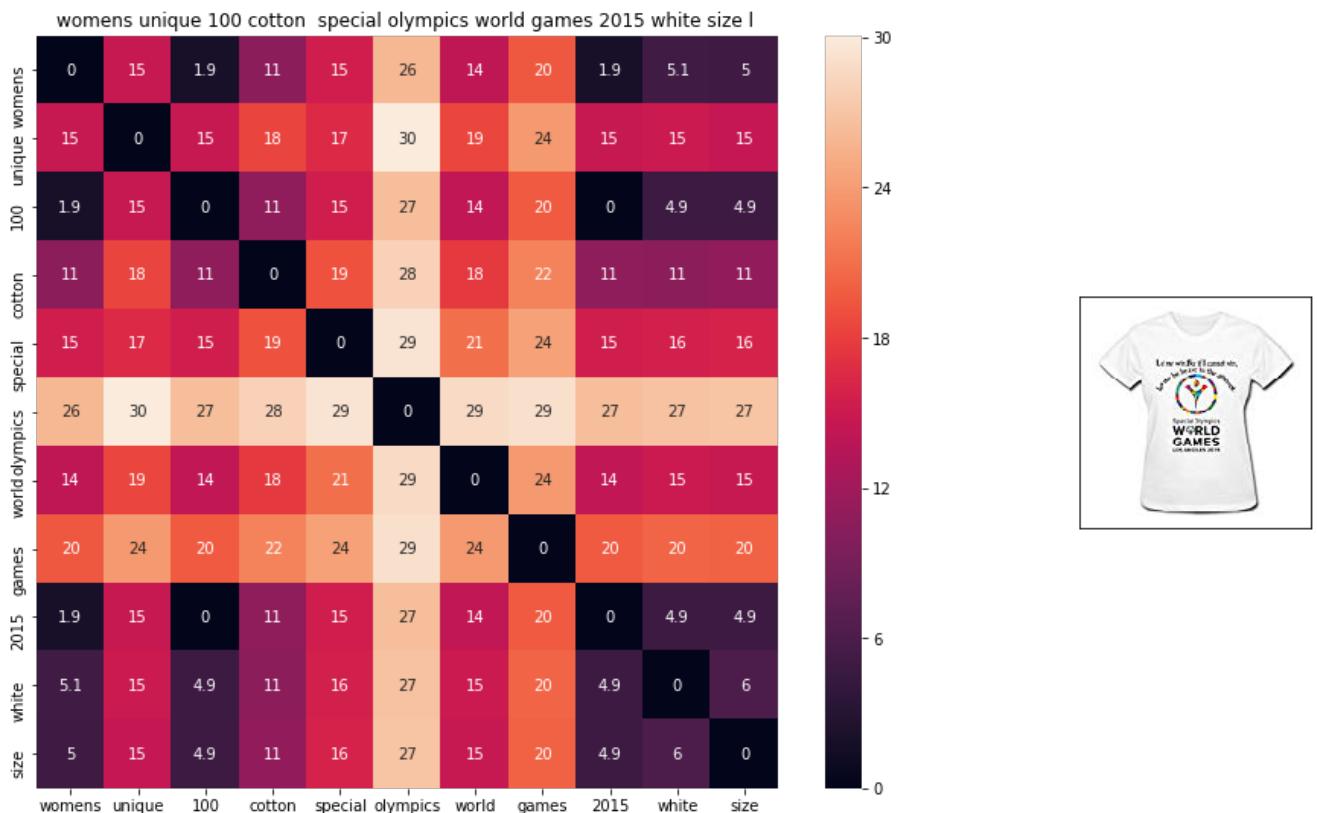
# np.argsort will return indices of 9 smallest distances
indices = np.argsort(pairwise_dist.flatten())[0:num_results]
#pdists will store the 9 smallest distances
pdists = np.sort(pairwise_dist.flatten())[0:num_results]

#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], '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)

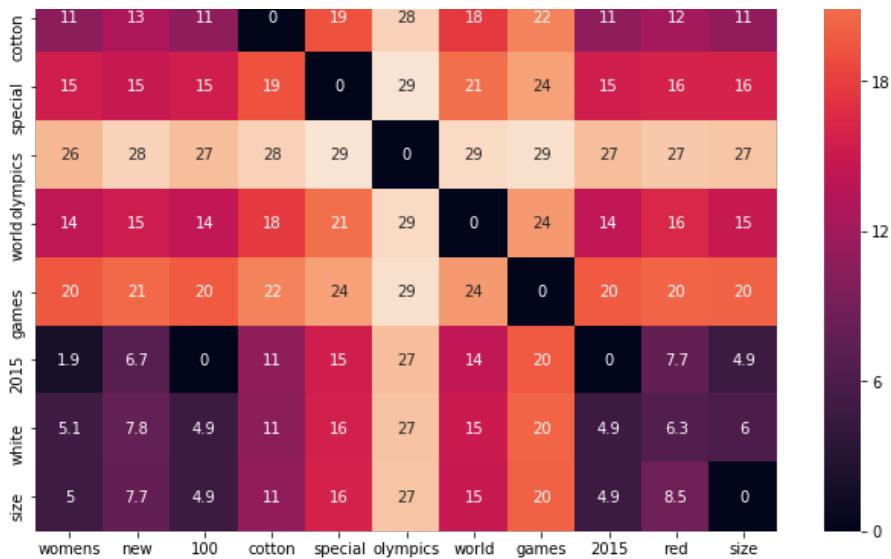
weighted_w2v_model(1, 20)
#931
#12566
# in the give heat map, each cell contains the euclidean distance between words i, j

```



ASIN : B012YX2ZPI
Brand : HX-Kingdom Fashion T-shirts
euclidean distance from input : 0.0027621358

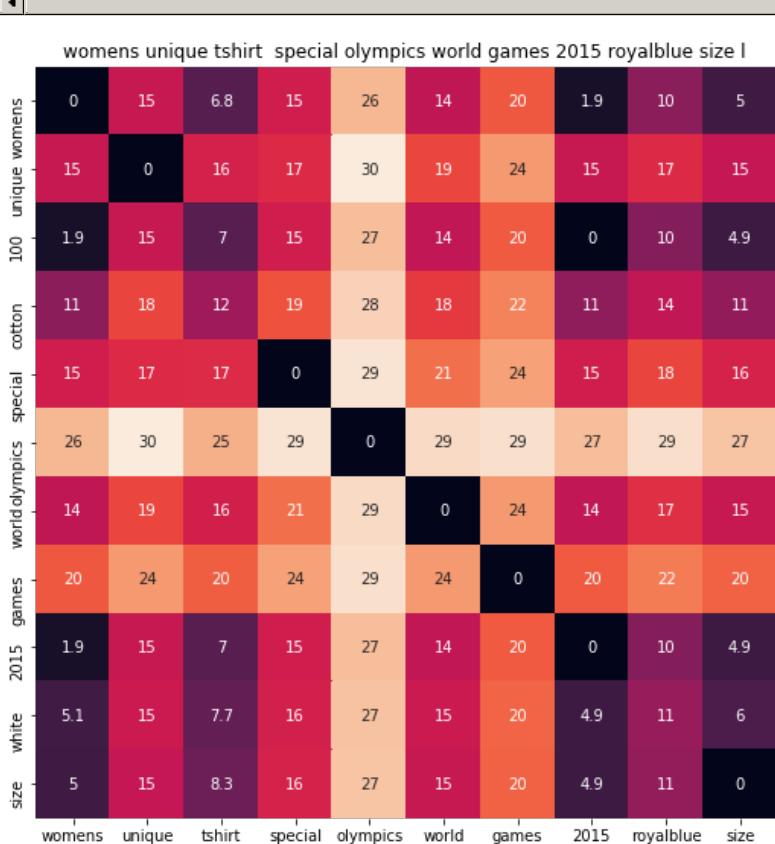




ASIN : B012YX3I5E

Brand : HX-Kingdom Fashion T-shirts

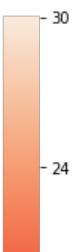
euclidean distance from input : 1.2900431

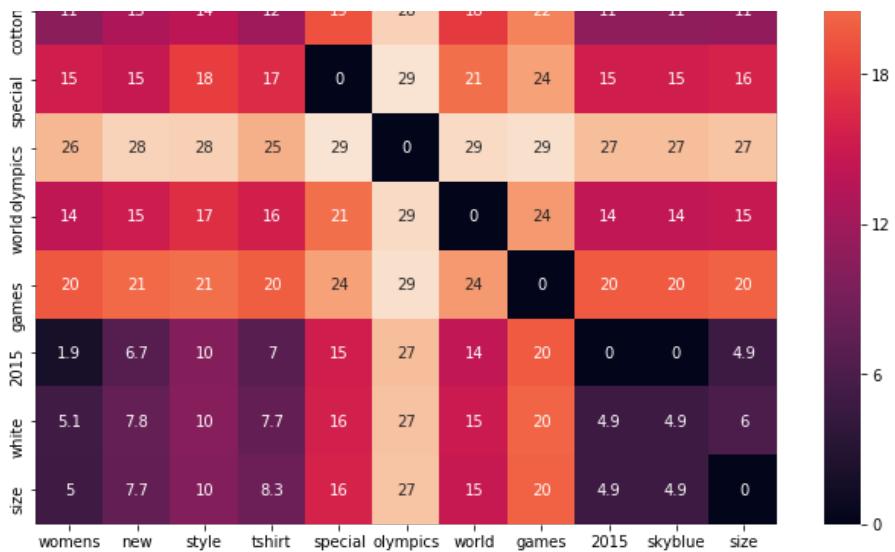


ASIN : B012YX4TVG

Brand : HX-Kingdom Fashion T-shirts

euclidean distance from input : 1.5112537

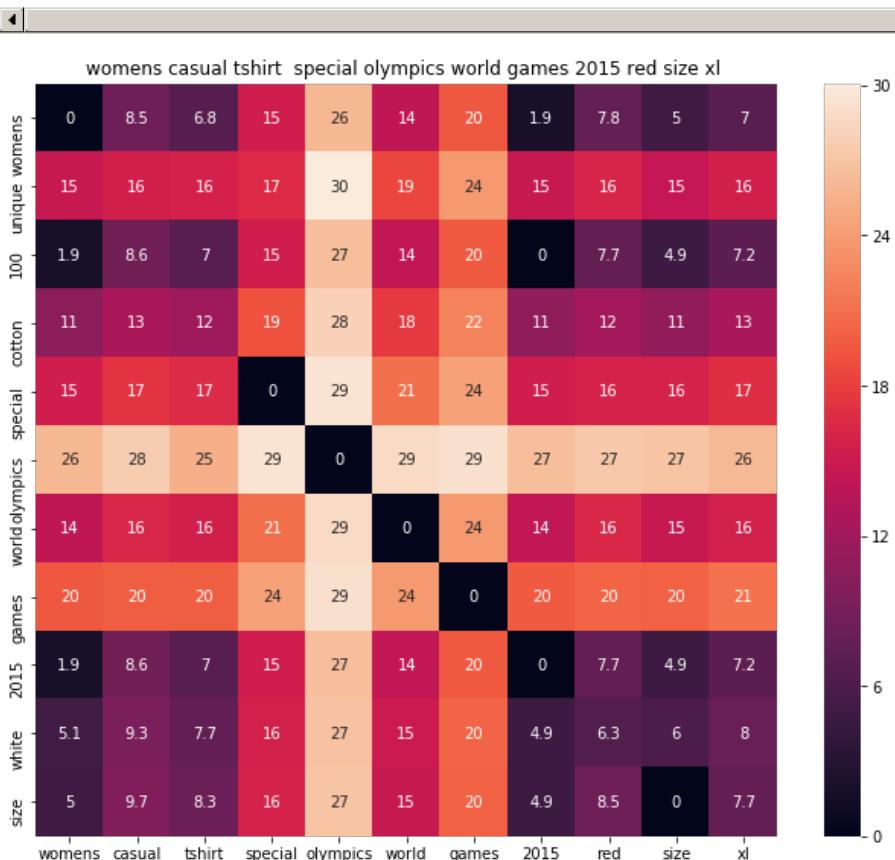




ASIN : B012YX2OJU

Brand : HX-Kingdom Fashion T-shirts

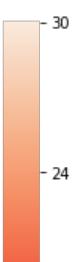
euclidean distance from input : 1.704583

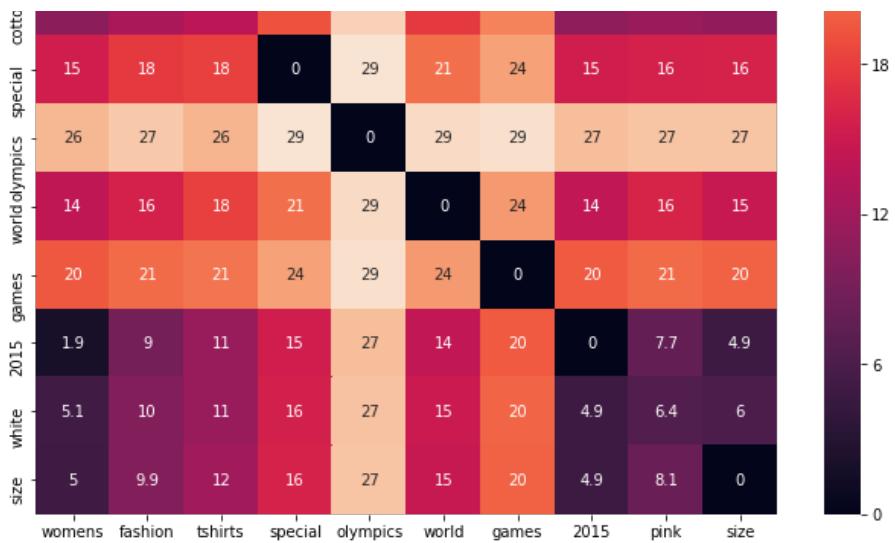


ASIN : B012YX3XTU

Brand : HX-Kingdom Fashion T-shirts

euclidean distance from input : 1.9207155

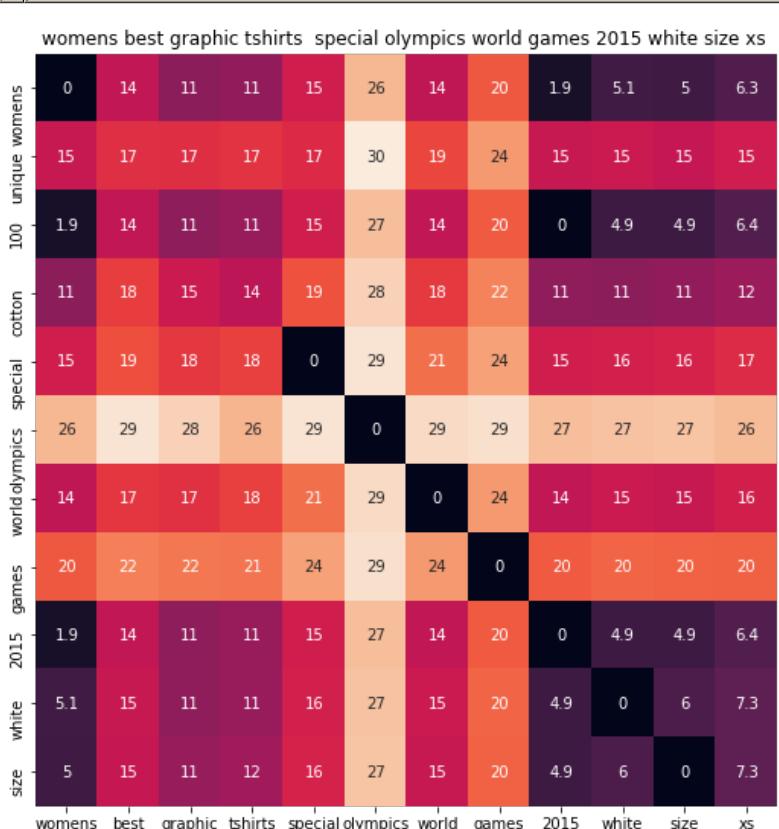




ASIN : B012YX26Q6

Brand : HX-Kingdom Fashion T-shirts

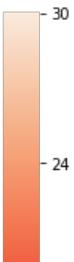
euclidean distance from input : 2.0728219

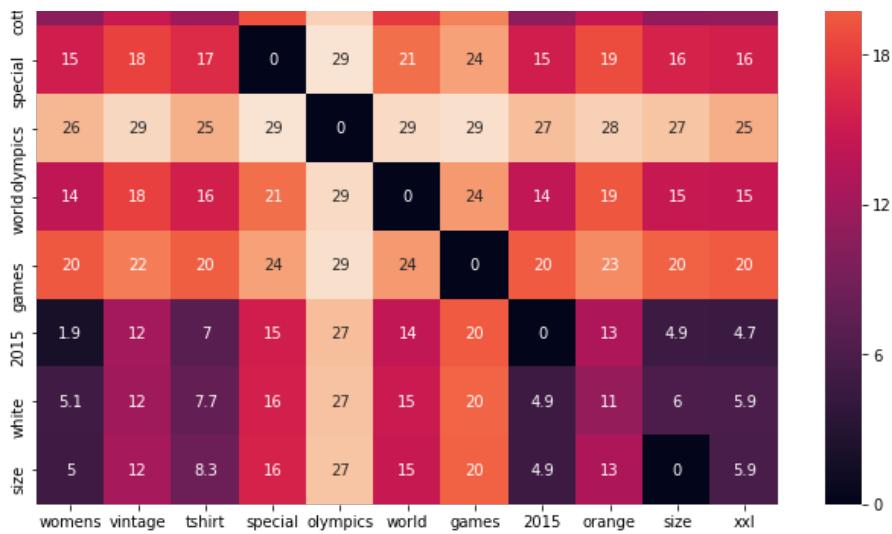


ASIN : B012YX2M3I

Brand : HX-Kingdom Fashion T-shirts

euclidean distance from input : 2.1508017

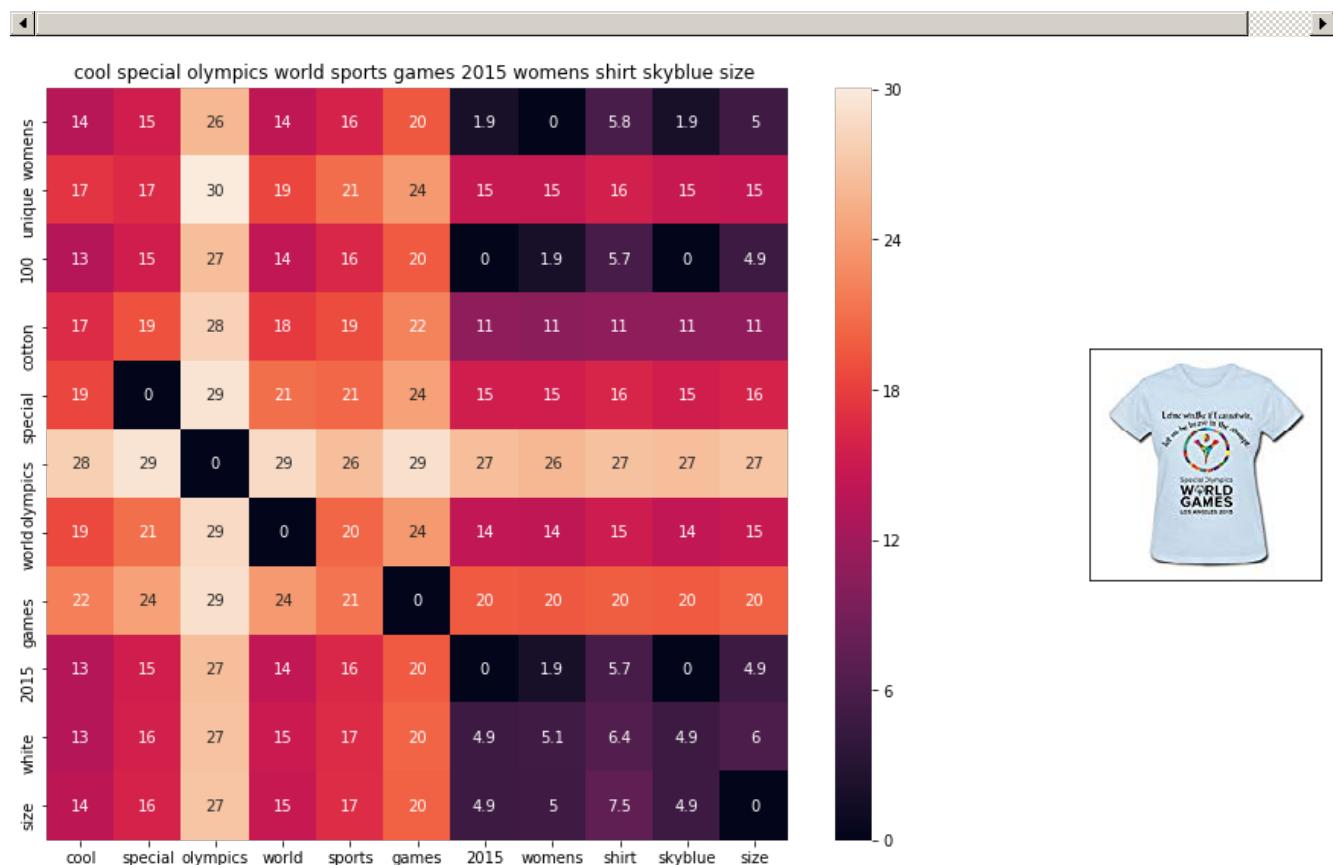




ASIN : B012YX4LVE

Brand : HX-Kingdom Fashion T-shirts

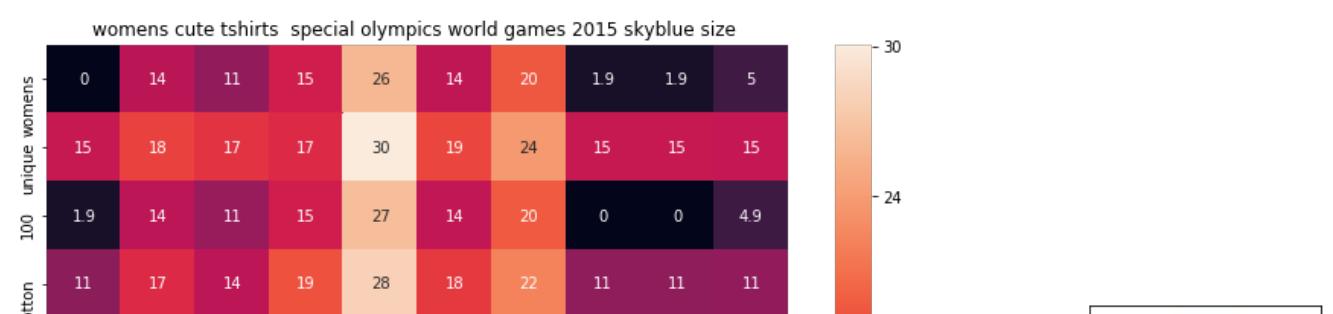
euclidean distance from input : 2.2502222

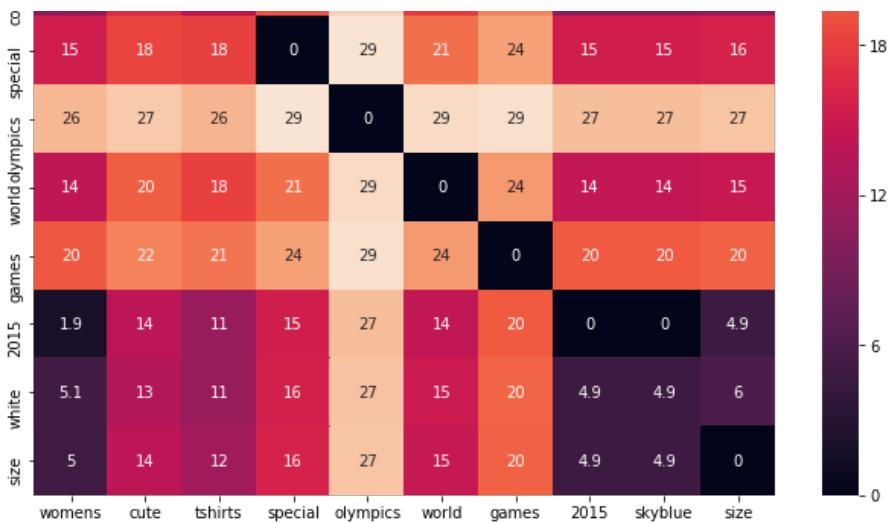


ASIN : B01338LWAM

Brand : MAM2 Arts

euclidean distance from input : 2.4271648

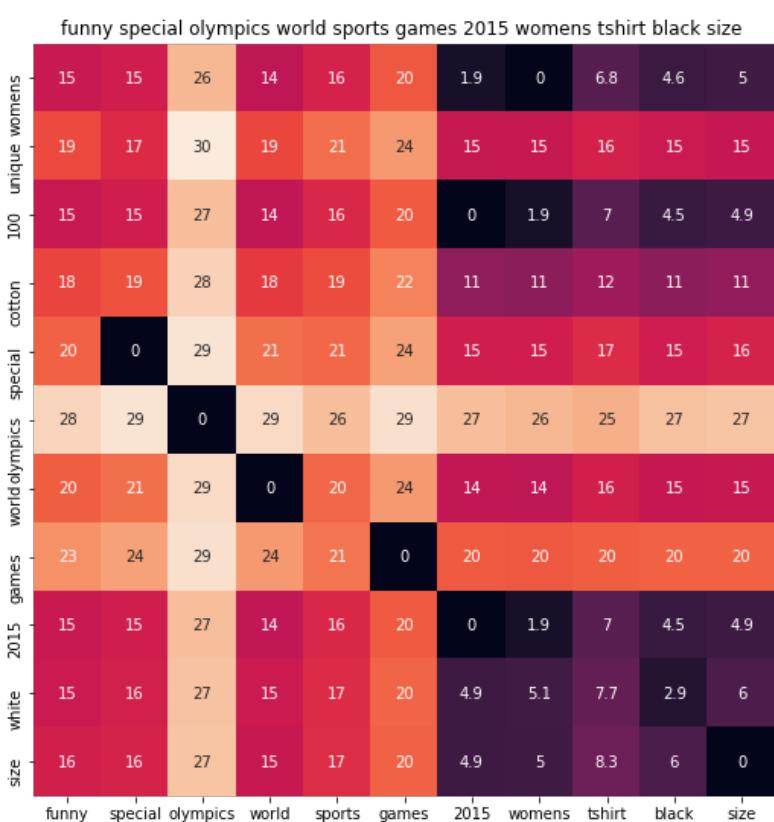




ASIN : B012YX30T8

Brand : HX-Kingdom Fashion T-shirts

euclidean distance from input : 2.436382



ASIN : B01338MTHM

Brand : MAM2 Arts

euclidean distance from input : 2.6622996

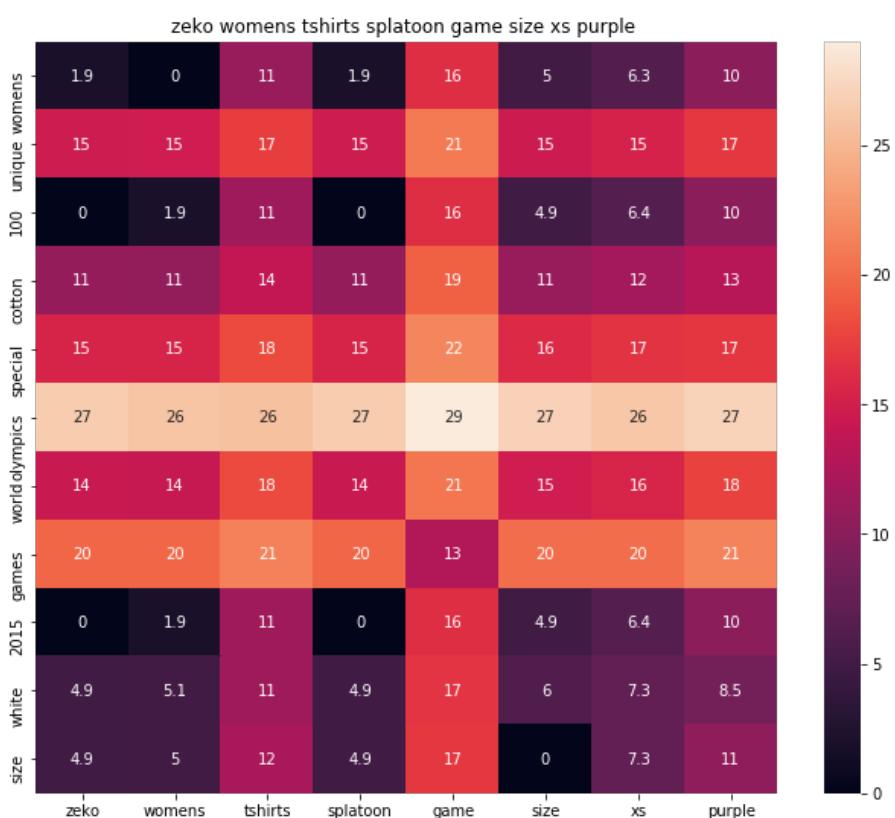




ASIN : B0167UU5ZQ

Brand : ZEKO

euclidean distance from input : 3.7380502

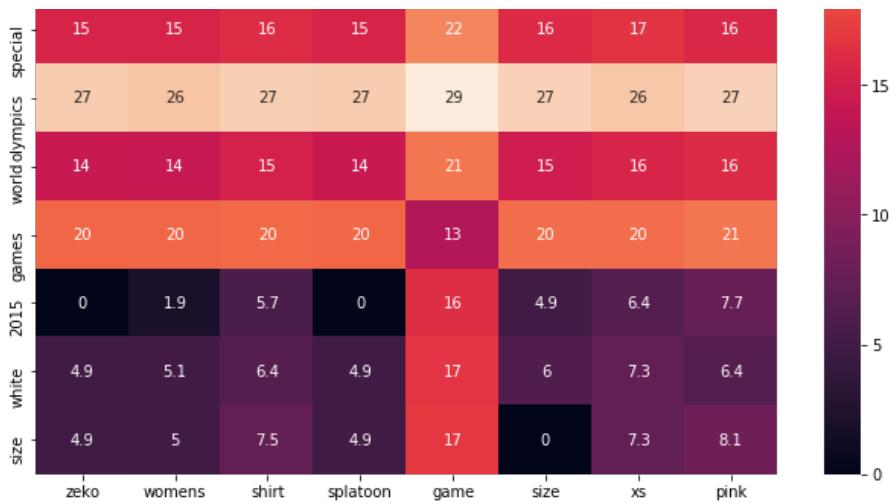


ASIN : B0167UURXQ

Brand : ZEKO

euclidean distance from input : 3.815717

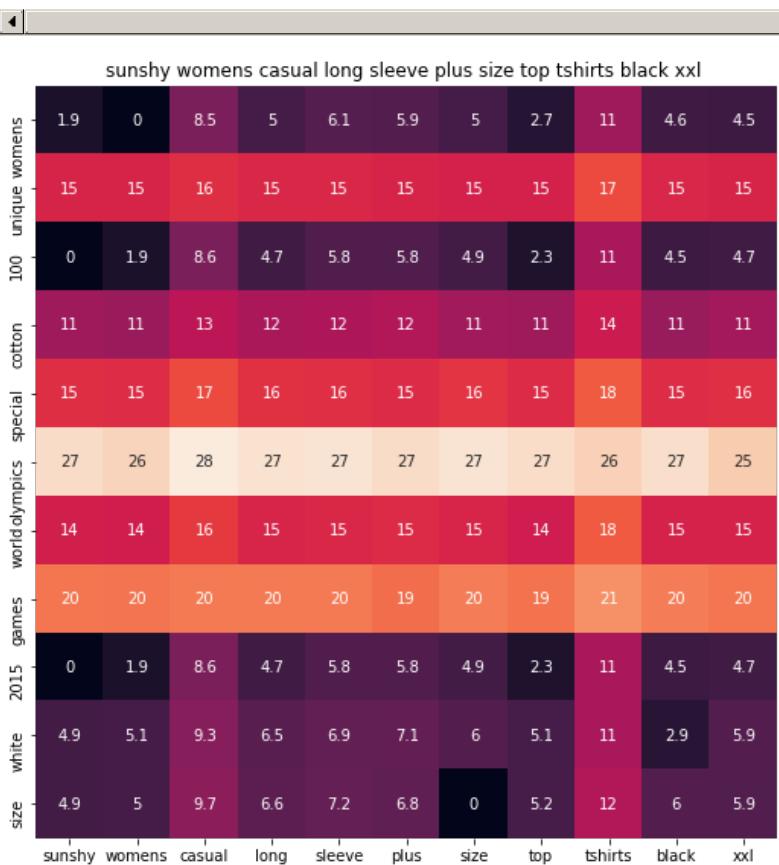




ASIN : B0167UTZ4I

Brand : ZEKO

euclidean distance from input : 3.8381796

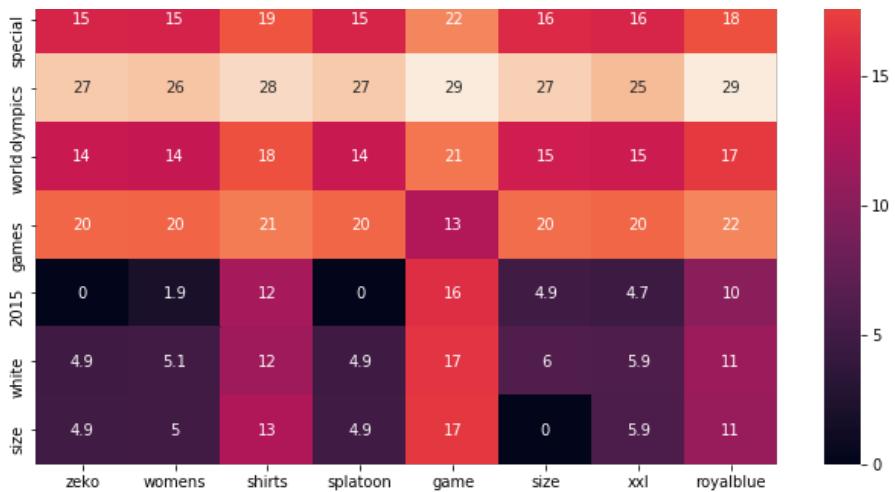


ASIN : B018EB3Q34

Brand : Juntong

euclidean distance from input : 3.8647017



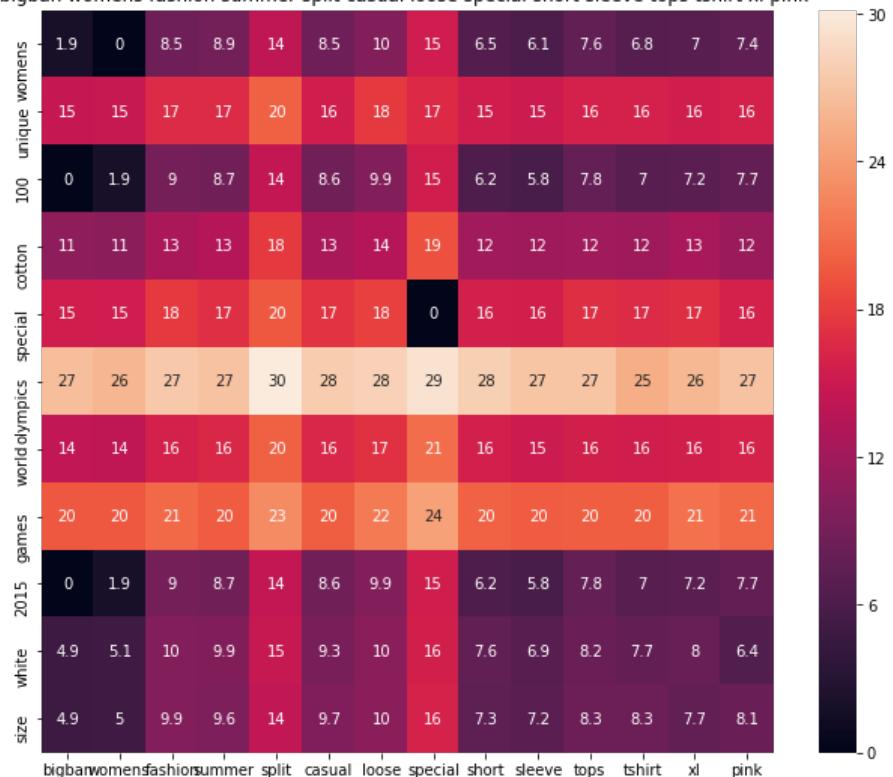


ASIN : B0167UVCGW

Brand : ZEKO

euclidean distance from input : 3.9313798

bigban womens fashion summer split casual loose special short sleeve tops tshirt xl pink



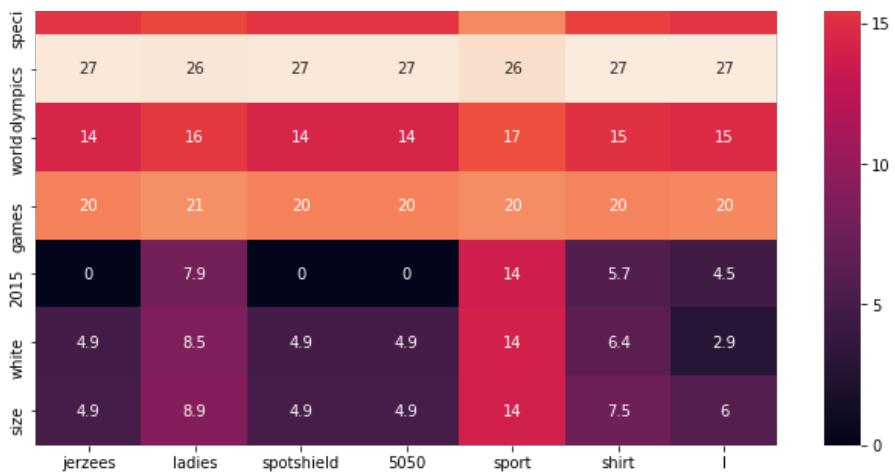
ASIN : B01HXCS9BO

Brand : Bigban

euclidean distance from input : 3.9710717

jerzees ladies spotshield 5050 sport shirt l black

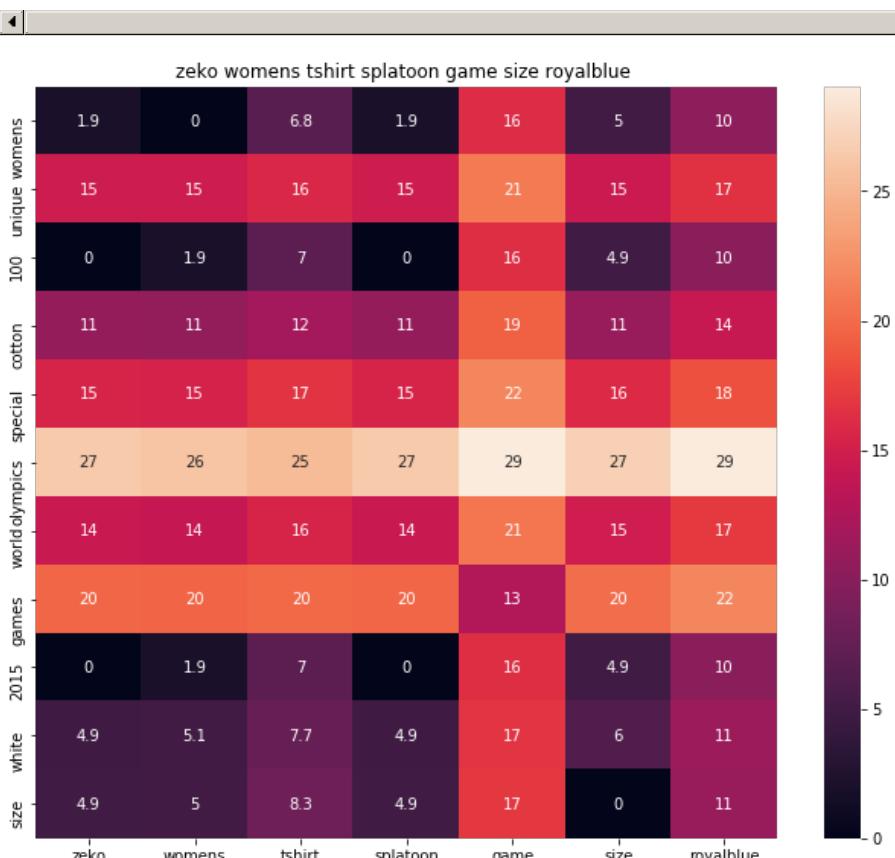




ASIN : B00FK49HDW

Brand : Jerzees

euclidean distance from input : 3.9770153

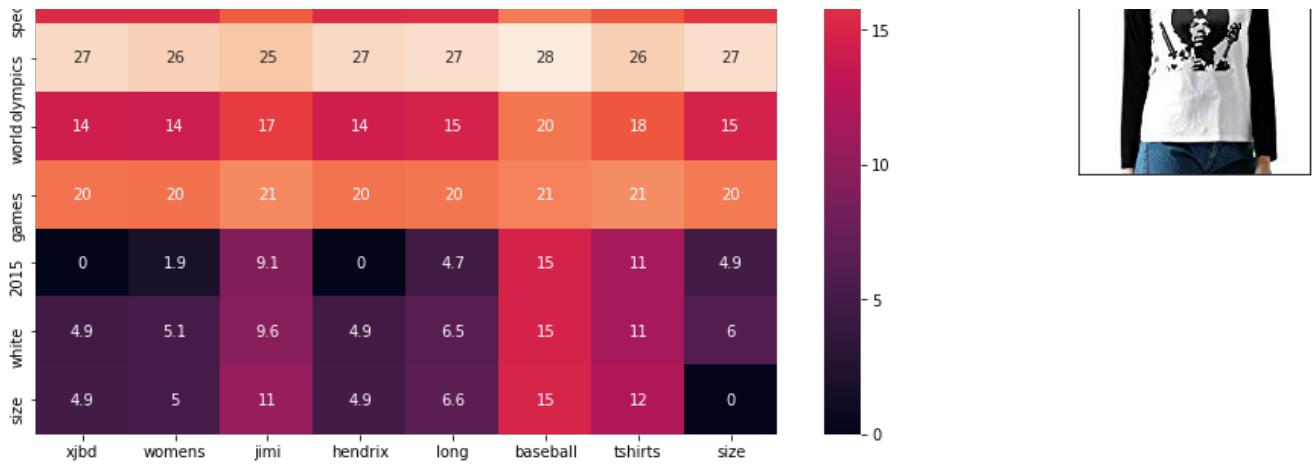


ASIN : B0167UV4KQ

Brand : ZEKO

euclidean distance from input : 3.9830794





ASIN : B01MF5BN69

Brand : XJBD

euclidean distance from input : 3.9987867

[9.6] Weighted similarity using brand and color.

In [74]:

```
# 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 hyphen
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 [75]:

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

    # sentance1 : title1, input apparel
    # sentance2 : title2, recommended apparel
    # url: apparel image url
    # doc_id1: document id of input apparel
    # doc_id2: document id of recommended 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/avg) 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/avg) 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'],
                  [url, doc_id1, doc_id2, df_id1, df_id2, model]]
```

```

        [data['asin'].loc[df_id1],brands[doc_id1], colors[doc_id1], types[doc_id1]], # input
apparel's features
        [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2], types[doc_id2]]] # recommended apparel's features

colorscale = [[0, '#1d004d'], [.5, '#f2e5ff'], [1, '#f2e5d1']] # to color the headings 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])
# plotting 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 [56]:

```

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 apparels
    # the metric we used here is cosine, the coside distance is mesured as  $K(X, Y) = \langle X, Y \rangle / (\|X\| * \|Y\|)$ 
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    idf_w2v_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(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)

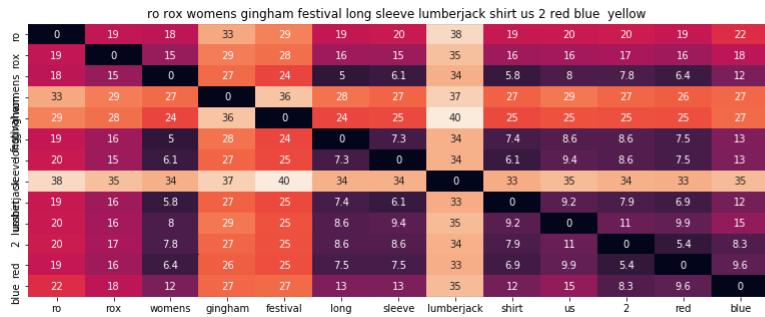
    # np.argsort will return indices of 9 smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists = np.sort(pairwise_dist.flatten())[0:num_results]

    #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_indices[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(4000, 5, 5, 20)
# in the give heat map, each cell contains the euclidean distance between words i, j

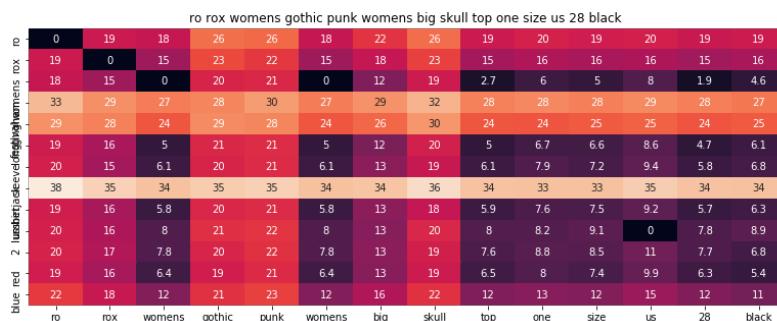
```



ASIN : B00THPW4VM

Brand : Ro Rox

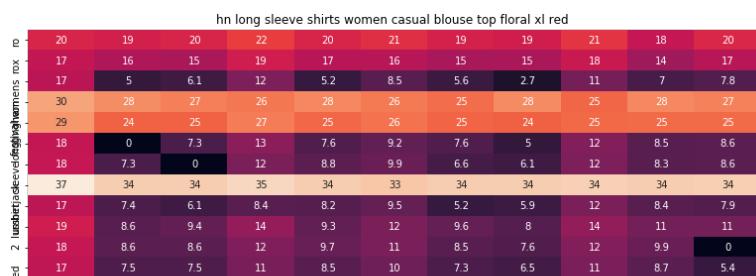
euclidean distance from input : 0.001381067931652069



ASIN : B00VOE195Q

Brand : Ro Rox

euclidean distance from input : 3.2149560928344725





ASIN : B0711KH9DX

Brand : H&N

euclidean distance from input : 3.2256261825561525

=====



yellow lao laotian sleeveless floral white lace blouse tops size 36 sl36a

color	10	18	18	19	21	21	19	21	19	20	19	19	19	19
size	22	18	15	15	17	18	16	17	15	16	16	15	15	15
item	18	17	5.4	1.9	10	11	5.1	9.4	5.6	7.6	5	1.9	1.9	1.9
2	12	16	27	27	28	25	25	27	24	25	27	28	28	28
3	27	27	24	24	26	25	25	25	25	25	25	24	24	24
4	13	16	7.1	4.7	11	12	6.5	11	7.6	8.5	6.6	4.7	4.7	4.7
5	13	16	7.7	5.8	10	12	6.9	9.6	6.6	9.2	7.2	5.8	5.8	5.8
6	35	36	34	34	34	34	33	33	34	33	33	34	34	34
7	12	16	7.4	5.7	9.5	12	6.4	9.7	5.2	8.7	7.5	5.7	5.7	5.7
8	15	17	8.4	7.8	13	14	9	12	9.6	11	9.1	7.8	7.8	7.8
9	8.3	17	8.7	7.7	12	12	6.3	11	8.5	9.3	8.5	7.7	7.7	7.7
10	9.6	16	7.7	6.3	11	11	4.8	9.9	7.3	8.8	7.4	6.3	6.3	6.3
11	0	19	13	12	15	14	10	14	12	13	12	12	12	12
12	yellow	lao	laos	laotian	sleeveless	floral	white	lace	blouse	tops	size	36	sl36a	



ASIN : B074J781N9

Brand : Nanon

euclidean distance from input : 3.2533508375047773

=====

=====



tommy hilfiger womens small plaid button shirt blue

color	10	19	18	19	24	22	19	19	19
size	19	15	15	16	19	17	16	16	16
item	15	1.9	0	5.2	14	13	5.8	6.4	6.4
2	28	28	27	28	21	28	27	26	26
3	28	24	24	25	27	28	25	25	25
4	16	4.7	5	6.2	15	13	7.4	7.5	7.5
5	16	5.8	6.1	7.7	14	12	6.1	7.5	7.5
6	33	34	34	34	31	35	33	33	33
7	16	5.7	5.8	7.5	13	12	0	6.9	6.9
8	17	7.8	8	8.8	16	14	9.2	9.9	9.9
9	17	7.7	7.8	8.4	14	13	7.9	5.4	5.4
10	16	6.3	6.4	7.4	13	13	6.9	0	0
11	19	12	12	12	16	16	12	9.6	9.6
12	tommy	hilfiger	womens	small	plaid	button	shirt	blue	



ASIN : B0759NSBCL

Brand : Tommy Hilfiger

euclidean distance from input : 3.256801489128405

=====

=====

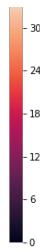


yellow lao laotian silk 34 sleeve button shirt wrap blouse tops sz 42 xl s42d

color	22	18	18	21	19	20	22	19	24	19	20	19	19	18	19
2	22	18	18	21	19	20	22	19	24	19	20	19	19	18	19



blue red	2 laotian	laos	laotian	silk	blend	sleeve	button	shirt	wrap	blouse	tops	sz	42	xl	s42d
18	17	15	18	15	15	17	16	20	15	16	16	15	14	15	
12	16	5.4	11	1.9	6.1	13	5.8	15	5.6	7.6	12	1.9	7	1.9	
27	30	27	25	28	27	28	27	29	25	27	27	28	28	28	
27	27	24	25	24	25	28	25	28	25	25	27	24	25	24	
13	16	7.1	12	4.7	7.3	13	7.4	15	7.6	8.5	14	4.7	8.5	4.7	
13	16	7.7	12	5.8	0	12	6.1	14	6.6	9.2	13	5.8	8.3	5.8	
35	36	34	34	34	34	35	33	36	34	33	35	34	34	34	
12	16	7.4	11	5.7	6.1	12	0	15	5.2	8.7	13	5.7	8.4	5.7	
15	17	8.4	14	7.8	9.4	14	9.2	17	9.6	11	16	7.8	11	7.8	
8.3	17	8.7	12	7.7	8.6	13	7.9	16	8.5	9.3	14	7.7	9.9	7.7	
9.6	16	7.7	11	6.3	7.5	13	6.9	15	7.3	8.8	13	6.3	8.7	6.3	
0	19	13	14	12	13	16	12	19	12	13	16	12	13	12	



ASIN : B0718YTQT5

Brand : Nanon

euclidean distance from input : 3.277779395854387



blue red	2 laotian	laos	laotian	silk	blend	sleeve	button	shirt	wrap	blouse	tops	sz	42	xl	
22	18	18	19	21	24	19	20	22	19	19	20	19	19	18	
18	17	15	15	18	20	15	15	17	16	15	16	16	15	14	
12	16	5.4	1.9	11	15	1.9	6.1	13	5.8	5.6	7.6	12	1.9	7	
27	30	27	28	25	29	28	27	28	27	25	27	27	28	28	
27	27	24	24	25	27	24	25	28	25	25	25	27	24	25	
13	16	7.1	4.7	12	15	4.7	7.3	13	7.4	7.6	8.5	14	4.7	8.5	
13	16	7.7	5.8	12	15	5.8	0	12	6.1	6.6	9.2	13	5.8	8.3	
35	36	34	34	34	36	34	34	35	33	34	33	35	34	34	
12	16	7.4	5.7	11	15	5.7	6.1	12	0	5.2	8.7	13	5.7	8.4	
15	17	8.4	7.8	14	16	7.8	9.4	14	9.2	9.6	11	16	7.8	11	
8.3	17	8.7	7.7	12	16	7.7	8.6	13	7.9	8.5	9.3	14	7.7	9.9	
9.6	16	7.7	6.3	11	15	6.3	7.5	13	6.9	7.3	8.8	13	6.3	8.7	
0	19	13	12	14	18	12	13	16	12	12	13	16	12	13	



ASIN : B01JFLTO9W

Brand : Nanon

euclidean distance from input : 3.287353523051652



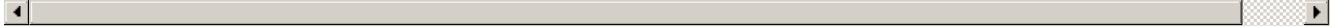
blue red	2 laotian	laos	laotian	silk	blend	sleeve	button	shirt	wrap	blouse	tops	sz	40	l	
22	18	18	21	19	20	22	19	24	19	20	19	19	19	19	
18	17	15	18	15	15	17	16	20	15	16	16	15	15	15	
12	16	5.4	11	1.9	6.1	13	5.8	15	5.6	7.6	12	1.9	1.9	1.9	
27	30	27	25	28	27	28	27	29	25	27	27	28	28	28	
27	27	24	25	24	25	28	25	28	25	25	27	24	24	24	
13	16	7.1	12	4.7	7.3	13	7.4	15	7.6	8.5	14	4.7	4.7	4.7	
13	16	7.7	12	5.8	0	12	6.1	14	6.6	9.2	13	5.8	5.8	5.8	
35	36	34	34	34	35	33	36	34	33	35	34	34	34	34	
12	16	7.4	11	5.7	6.1	12	0	15	5.2	8.7	13	5.7	5.7	5.7	
15	17	8.4	14	7.8	9.4	14	9.2	17	9.6	11	16	7.8	7.8	7.8	
8.3	17	8.7	12	7.7	8.6	13	7.9	16	8.5	9.3	14	7.7	7.7	7.7	
9.6	16	7.7	11	6.3	7.5	13	6.9	15	7.3	8.8	13	6.3	6.3	6.3	
0	19	13	14	12	13	16	12	19	12	13	16	12	12	12	

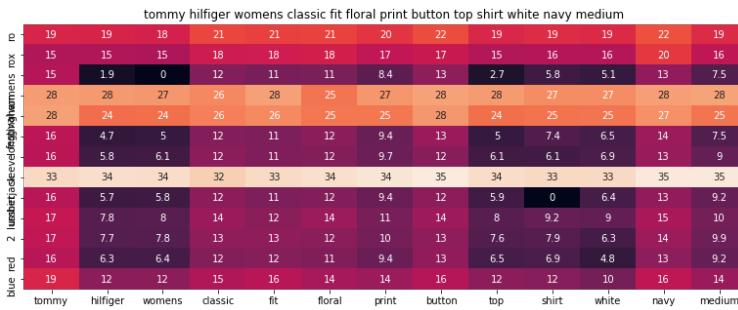


ASIN : B06XHCK59R

Brand : Nanon

euclidean distance from input : 3.2986614301561445





ASIN : B001980FYEY

Brand : Ashworth

euclidean distance from input : 3.3072891309618084

=====



ASIN : B071HBD7LY

Brand : Nanon

euclidean distance from input : 3.307752998149309

=====

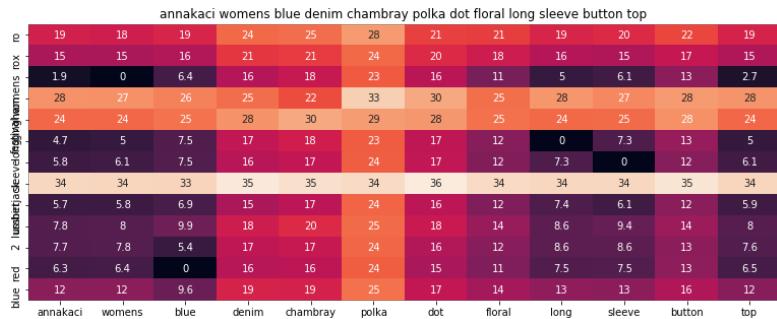


ASIN : B01K5BRKGK

Brand : RRL

euclidean distance from input : 3.3118551328538985

=====

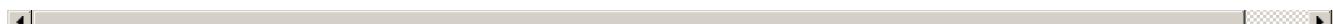


ASIN : B008SMIFN6

Brand : Anna-Kaci

euclidean distance from input : 3.3135590345995967

=====

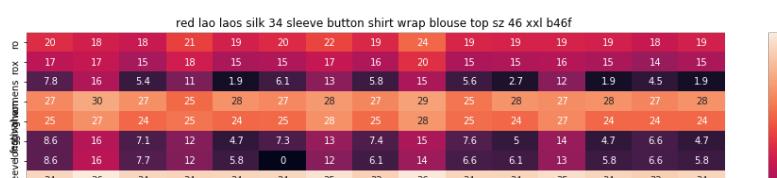


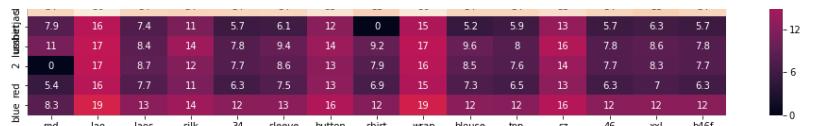
ASIN : B01MS7H2R5

Brand : Shouhengda

euclidean distance from input : 3.3166797712205978

=====





ASIN : B071YMD8QN

Brand : Nanon

euclidean distance from input : 3.3183124616502853



ASIN : B074QTKXPP

Brand : Chloe K.

euclidean distance from input : 3.3226463392137617

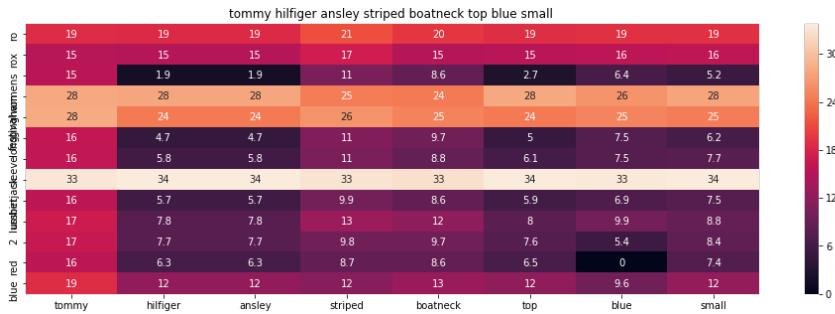


ASIN : B06XG6HCZJ

Brand : Nanon

euclidean distance from input : 3.326060493266496

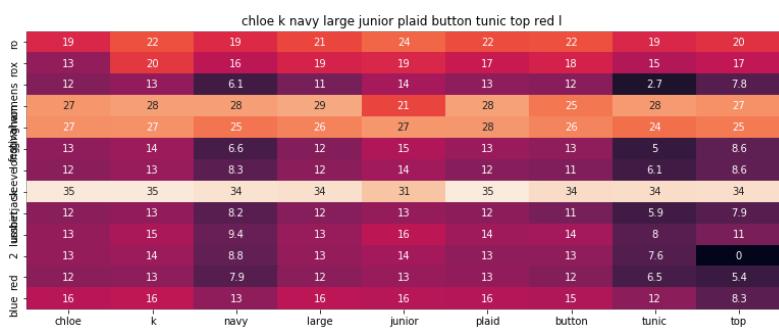




ASIN : B01LYYU8BB

Brand : Tommy Hilfiger

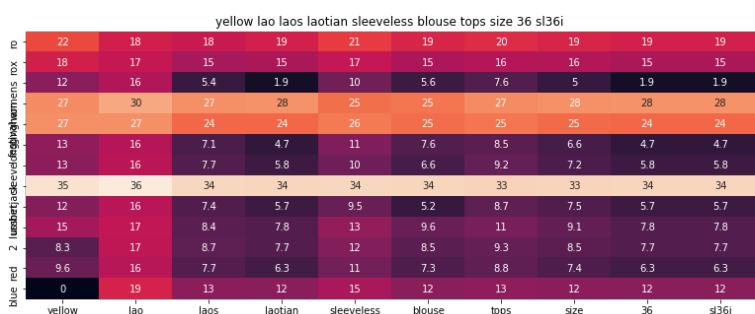
euclidean distance from input : 3.328615453972155



ASIN : B074MJSJX8

Brand : Chloe K.

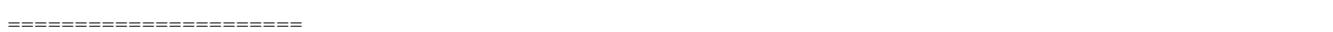
euclidean distance from input : 3.3397876813768477

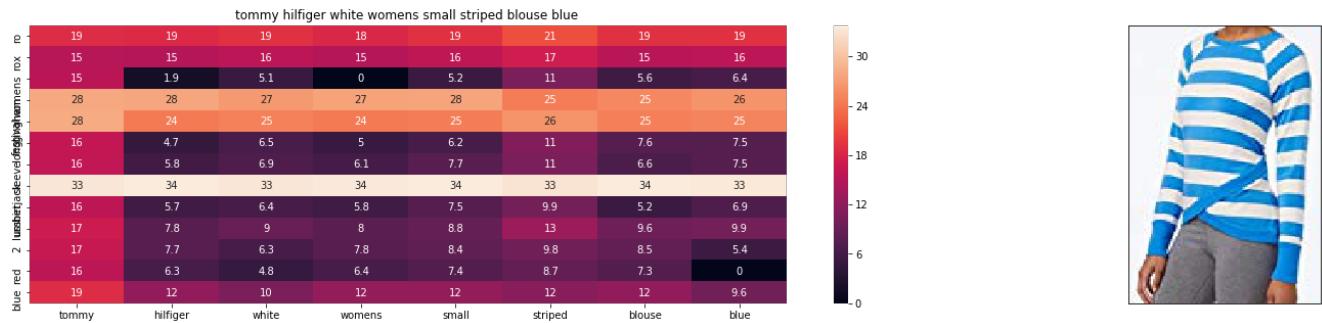
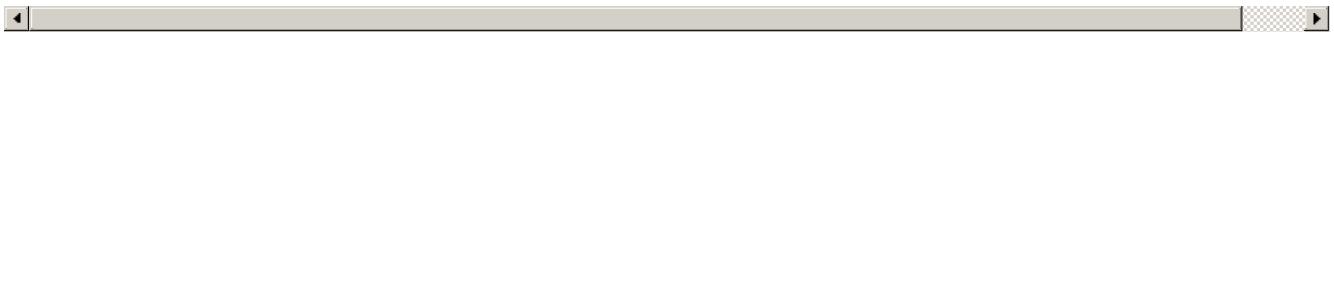


ASIN : B074HXKRSG

Brand : Nanon

euclidean distance from input : 3.3486057355760663





ASIN : B0759NSBV6

Brand : Tommy Hilfiger

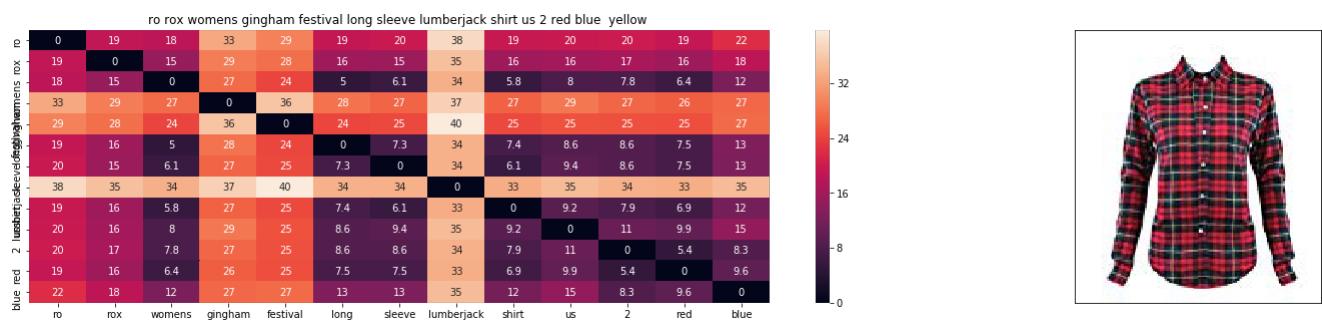
euclidean distance from input : 3.359269407524401



In [57]:

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

idf_w2v_brand(4000, 5, 50, 20)
```

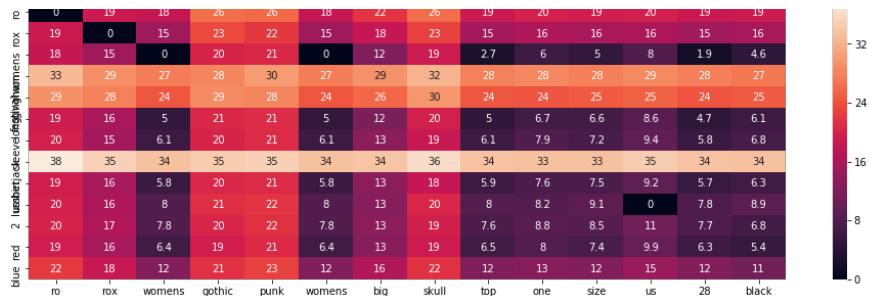


ASIN : B00THPW4VM

Brand : Ro Rox

euclidean distance from input : 0.0002511032603003762

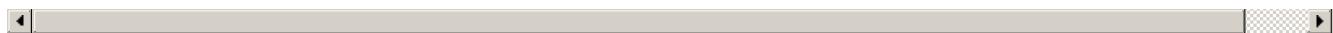




ASIN : B00VOE195Q

Brand : Ro Rox

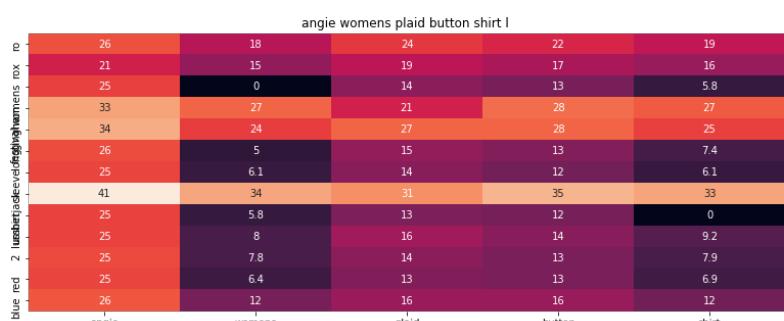
euclidean distance from input : 2.220901107788086



ASIN : B0711KH9DX

Brand : H&N

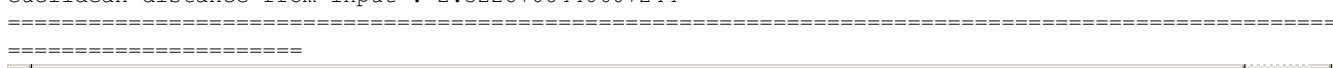
euclidean distance from input : 2.2228411241011186

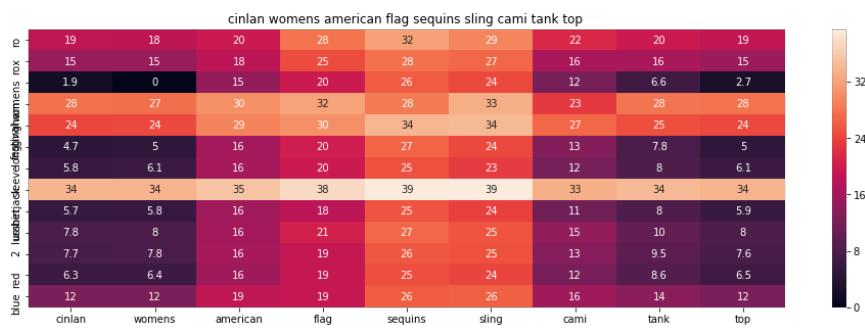


ASIN : B06XKHQ5P5

Brand : Angie

euclidean distance from input : 2.3228708440607244





ASIN : B01AUJPCP8

Brand : Cinlan

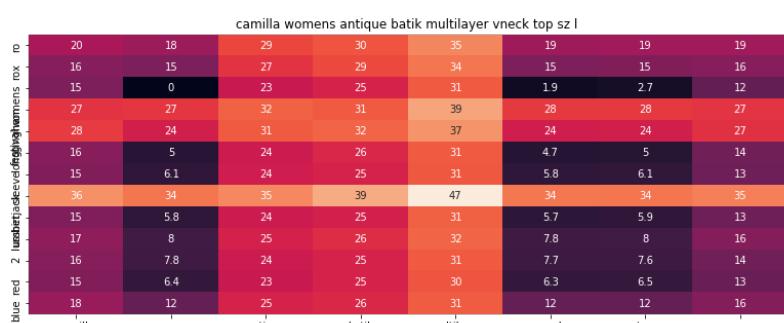
euclidean distance from input : 2.3829784740101205



ASIN : B074J781N9

Brand : Nanon

euclidean distance from input : 2.4210284975006964



camilla womens antique batik multilayer vneck top sz

ASIN : B073YK11HQ

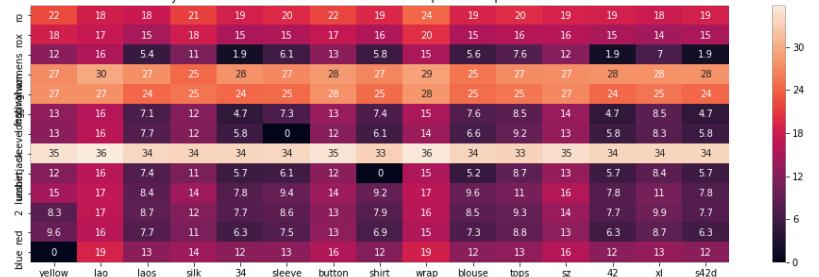
Brand : Camilla

euclidean distance from input : 2.424445134943182

=====



yellow lao laos silk 34 sleeve button shirt wrap blouse tops sz 42 xl s42d



ASIN : B0718YTQ75

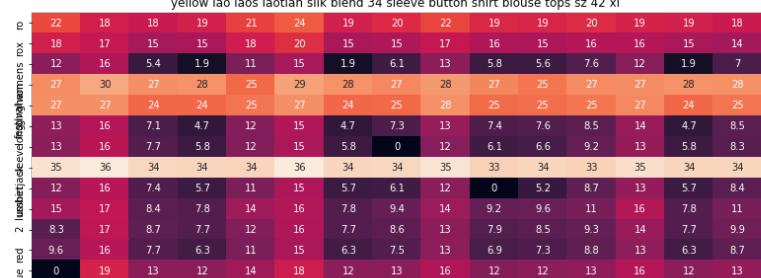
Brand : Nanon

euclidean distance from input : 2.4254700535642617

=====



yellow lao laos laotian silk blend 34 sleeve button shirt blouse tops sz 42 xl

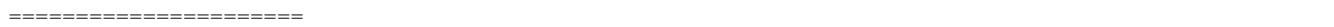


ASIN : B01JFLT09W

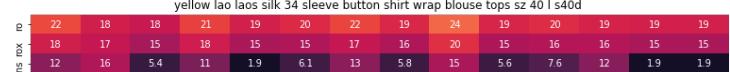
Brand : Nanon

euclidean distance from input : 2.4272108039637645

=====



yellow lao laos silk 34 sleeve button shirt wrap blouse tops sz 40 l



	27	30	27	25	20	27	20	27	27	25	25	27	27	20	20
blue red	27	27	24	25	24	25	28	25	28	25	25	27	24	24	24
	13	16	7.1	12	4.7	7.3	13	7.4	15	7.6	8.5	14	4.7	4.7	4.7
	13	16	7.7	12	5.8	0	12	6.1	14	6.6	9.2	13	5.8	5.8	5.8
	35	36	34	34	34	35	33	36	34	33	35	34	34	34	34
	12	16	7.4	11	5.7	6.1	12	0	15	5.2	8.7	13	5.7	5.7	5.7
	15	17	8.4	14	7.8	9.4	14	9.2	17	9.6	11	16	7.8	7.8	7.8
	8.3	17	8.7	12	7.7	8.6	13	7.9	16	8.5	9.3	14	7.7	7.7	7.7
	9.6	16	7.7	11	6.3	7.5	13	6.9	15	7.3	8.8	13	6.3	6.3	6.3
	0	19	13	14	12	13	16	12	19	12	13	16	12	12	12
yellow	lao	laos	silk	34	sleeve	button	shirt	wrap	blouse	tops	sz	40	1		



ASIN : B06XHCK59R

Brand : Nanon

euclidean distance from input : 2.4292667870736726

=====

[<] [] [>]

	19	19	18	21	21	21	20	22	19	19	19	22	19		
blue red	15	15	15	18	18	18	17	17	15	16	16	20	16		
	15	1.9	0	12	11	11	8.4	13	2.7	5.8	5.1	13	7.5		
	28	28	27	26	28	25	27	28	28	27	27	28	28		
	28	24	24	26	26	25	25	28	24	25	25	27	25		
	16	4.7	5	12	11	12	9.4	13	5	7.4	6.5	14	7.5		
	16	5.8	6.1	12	11	12	9.7	12	6.1	6.1	6.9	13	9		
	33	34	34	32	33	34	34	35	34	33	33	35	35		
	16	5.7	5.8	12	11	12	9.4	12	5.9	0	6.4	13	9.2		
	17	7.8	8	14	12	14	11	14	8	9.2	9	15	10		
	17	7.7	7.8	13	13	12	10	13	7.6	7.9	6.3	14	9.9		
	16	6.3	6.4	12	12	11	9.4	13	6.5	6.9	4.8	13	9.2		
	19	12	12	15	16	14	14	16	12	12	10	16	14		
tommy	hilfiger	womens	classic	fit	floral	print	button	top	shirt	white	navy	medium			



ASIN : B001980FEY

Brand : Ashworth

euclidean distance from input : 2.4308354599474296

=====

[<] [] [>]

	19	18	18	21	19	20	22	19	19	19	19	18	19		
blue red	16	17	15	18	15	15	17	16	15	16	15	14	15		
	6.4	16	5.4	11	1.9	6.1	13	5.8	5.6	12	1.9	4.5	1.9		
	26	30	27	25	28	27	28	27	25	27	28	27	28		
	25	27	24	25	24	25	28	25	25	27	24	24	24		
	7.5	16	7.1	12	4.7	7.3	13	7.4	7.6	14	4.7	6.6	4.7		
	7.5	16	7.7	12	5.8	0	12	6.1	6.6	13	5.8	6.6	5.8		
	33	36	34	34	34	35	35	33	34	35	34	33	34		
	6.9	16	7.4	11	5.7	6.1	12	0	5.2	13	5.7	6.3	5.7		
	9.9	17	8.4	14	7.8	9.4	14	9.2	9.6	16	7.8	8.6	7.8		
	5.4	17	8.7	12	7.7	8.6	13	7.9	8.5	14	7.7	8.3	7.7		
	0	16	7.7	11	6.3	7.5	13	6.9	7.3	13	6.3	7	6.3		
	9.6	19	13	14	12	13	16	12	12	16	12	12	12		
blue	lao	laos	silk	34	sleeve	button	shirt	blouse	sz	46	xxl	s46f			



ASIN : B071HBD7LY

Brand : Nanon

euclidean distance from input : 2.430919799436066

=====

[<] [] [>]

ralph lauren rrl double rl womens long sleeve striped work shirt blue white size 1 small																			
e	17	17	19	23	22	18	19	20	21	22	19	19	19	19	19	19	19	19	
blue red	2	luster	devoid	feeling	mens	ox													
	17	13	15	20	22	15	16	15	17	20	16	16	16	16	16	16	16	16	
	11	12	19	15	23	0	5	6.1	11	12	5.8	6.4	5.1	5	5.2				
	28	28	28	29	36	27	28	27	25	31	27	26	27	28	28	28	28	28	
	26	26	24	28	33	24	24	25	26	27	25	25	25	25	25	25	25	25	
	12	13	4.7	15	24	5	0	7.3	11	12	7.4	7.5	6.5	6.6	6.2				
	13	13	5.8	15	24	6.1	7.3	0	11	13	6.1	7.5	6.9	7.2	7.7				
	34	34	34	36	40	34	34	34	33	36	33	33	33	33	33	34	34	34	
	12	12	5.7	15	24	5.8	7.4	6.1	9.9	13	0	6.9	6.4	7.5	7.5				
	13	13	7.8	16	25	8	8.6	9.4	13	13	9.2	9.9	9	9.1	8.8				
	13	14	7.7	15	24	7.8	8.6	8.6	9.8	15	7.9	9.4	6.3	8.5	8.4				
	12	13	6.3	15	24	6.4	7.5	7.5	8.7	14	6.9	0	4.8	7.4	7.4				
	15	16	12	17	27	12	13	13	12	17	12	9.6	10	12	12				
	ralph	lauren	ml	double	rl	womens	long	sleeve	striped	work	shirt	blue	white	size	1				



ASIN : B01K5BRKGK

Brand : RRL

euclidean distance from input : 2.431665642109628

=====

=====

=====

shouhengda women long sleeve loose blouse ladies solid color casual ol shirt tops xxl																			
e	19	20	19	20	21	19	19	21	22	21	25	19	20	18	19	20	18	19	
blue red	2	luster	devoid	feeling	mens	ox													
	19	17	16	15	17	15	16	18	18	16	21	16	16	14					
	1.9	5.2	5	6.1	10	5.6	7.1	10	12	8.5	21	5.8	7.6	4.5					
	28	28	28	27	28	25	27	29	27	26	30	27	27	27	27	27	27	27	
	24	25	24	25	26	25	25	26	26	26	32	25	25	24					
	4.7	7.6	0	7.3	10	7.6	9	9.8	13	9.2	21	7.4	8.5	6.6					
	5.8	8.8	7.3	0	11	6.6	9.6	11	12	9.9	21	6.1	9.2	6.6					
	34	34	34	34	34	34	33	35	35	33	33	33	33	33	33	33	33	33	
	5.7	8.2	7.4	6.1	11	5.2	9.1	11	12	9.5	22	0	8.7	6.3					
	7.8	9.3	8.6	9.4	12	9.6	9.7	12	14	12	21	9.2	11	8.6					
	7.7	9.7	8.6	8.6	12	8.5	10	12	11	11	22	7.9	9.3	8.3					
	6.3	8.5	7.5	7.5	11	7.3	9.6	11	10	10	21	6.9	8.8	7					
	12	14	13	13	15	12	14	15	13	15	23	12	13	12					
	shouhengda	women	long	sleeve	loose	blouse	ladies	solid	color	casual	ol	shirt	tops	xxl					



ASIN : B01MS7H2R5

Brand : Shouhengda

euclidean distance from input : 2.432542849085391

=====

=====

=====

red lao laos silk 34 sleeve button shirt wrap blouse top sz 46 xxl b46f																			
e	20	18	18	21	19	20	22	19	24	19	19	19	19	18	19	19	19	19	
blue red	2	luster	devoid	feeling	mens	ox													
	20	17	15	18	15	15	17	16	20	15	15	15	16	15	14	15	15	15	
	7.8	16	5.4	11	1.9	6.1	13	5.8	15	5.6	2.7	12	1.9	4.5	1.9				
	27	30	27	25	28	27	28	27	29	25	28	27	28	27	28	27	28	28	
	25	27	24	25	24	25	28	25	28	25	24	27	24	24	24	24	24	24	
	8.6	16	7.1	12	4.7	7.3	13	7.4	15	7.6	5	14	4.7	6.6	4.7				
	8.6	16	7.7	12	5.8	0	12	6.1	14	6.6	6.1	13	5.8	6.6	5.8				
	34	36	34	34	34	34	35	33	36	34	34	35	34	33	34				
	7.9	16	7.4	11	5.7	6.1	12	0	15	5.2	5.9	13	5.7	6.3	5.7				
	11	17	8.4	14	7.8	9.4	14	9.2	17	9.6	8	16	7.8	8.6	7.8				
	0	17	8.7	12	7.7	8.6	13	7.9	16	8.5	7.6	14	7.7	8.3	7.7				
	5.4	16	7.7	11	6.3	7.5	13	6.9	15	7.3	6.5	13	6.3	7	6.3				
	8.3	19	13	14	12	13	16	12	19	12	12	16	12	12	12				
	red	lao	laos	silk	34	sleeve	button	shirt	wrap	blouse	top	sz	46	xxl	b46f				



ASIN : B071YMD8QN

Brand : Nanon

euclidean distance from input : 2.432839701890789

=====

=====



chloe k white womens striped colorblock blouse blue xs

19	19	18	21	20	19	19	17
13	16	15	17	16	15	16	14
12	5.1	0	11	8.5	5.6	6.4	6.3
27	27	27	25	25	25	26	27
27	25	24	26	24	25	25	25
13	6.5	5	11	10	7.6	7.5	7.7
12	6.9	6.1	11	9.2	6.6	7.5	8
35	33	34	33	33	34	33	34
12	6.4	5.8	9.9	9	5.2	6.9	7.8
13	9	8	13	11	9.6	9.9	10
13	6.3	7.8	9.8	9.6	8.5	5.4	9.4
12	4.8	6.4	8.7	8.6	7.3	0	7.9
16	10	12	12	13	12	9.6	13
chloe	k	white	womens	striped	colorblock	blouse	blue

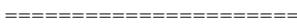


ASIN : B074QTKXPP

Brand : Chloe K.

euclidean distance from input : 2.433627679629603

=====



red black lao laos silk 34 sleeve button shirt blouse wrap sz 34 s340

20	19	18	18	21	19	20	22	19	19	24	19	19	19
17	16	17	15	18	15	15	17	16	15	20	16	15	15
7.8	4.6	16	5.4	11	19	6.1	13	5.8	5.6	15	12	1.9	1.9
27	27	30	27	25	28	27	28	27	25	29	27	28	28
25	25	27	24	25	24	25	28	25	25	28	27	24	24
8.6	6.1	16	7.1	12	4.7	7.3	13	7.4	7.6	15	14	4.7	4.7
8.6	6.8	16	7.7	12	5.8	0	12	6.1	6.6	14	13	5.8	5.8
34	34	36	34	34	34	34	35	33	34	36	35	34	34
7.9	6.3	16	7.4	11	5.7	6.1	12	0	5.2	15	13	5.7	5.7
11	8.9	17	8.4	14	7.8	9.4	14	9.2	9.6	17	16	7.8	7.8
0	6.8	17	8.7	12	7.7	8.6	13	7.9	8.5	16	14	7.7	7.7
5.4	5.4	16	7.7	11	6.3	7.5	13	6.9	7.3	15	13	6.3	6.3
8.3	11	19	13	14	12	13	16	12	12	19	16	12	12
red	black	lao	laos	silk	34	sleeve	button	shirt	blouse	wrap	sz	34	s340



ASIN : B06XG6HCZJ

Brand : Nanon

euclidean distance from input : 2.4342484349119182

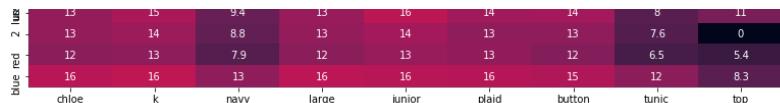
=====



chloe k navy large junior plaid button tunic top red l

19	22	19	21	24	22	22	19	20
13	20	16	19	19	17	18	15	17
12	13	6.1	11	14	13	12	2.7	7.8
27	28	28	29	21	28	25	28	27
27	27	25	26	27	28	26	24	25
13	14	6.6	12	15	13	13	5	8.6
12	13	8.3	12	14	12	11	6.1	8.6
35	35	34	34	31	35	34	34	34
12	13	8.2	12	13	12	11	5.9	7.9
13	15	8.1	13	15	14	14	6	13
shirts	deals	vests	high	mens	ro	tops	blouses	tunics





ASIN : B074MJSJX8

Brand : Chloe K.

euclidean distance from input : 2.4367442872956184

=====

=====

=====

=====

yellow lao laotian sleeveless blouse tops size 36 sl36i

re	22	18	18	19	21	19	20	19	19	19
blue red	18	17	15	15	17	15	16	16	15	15
2	12	16	5.4	1.9	10	5.6	7.6	5	1.9	1.9
lao	27	30	27	28	25	25	27	28	28	28
laos	27	27	24	24	26	25	25	25	24	24
laotian	13	16	7.1	4.7	11	7.6	8.5	6.6	4.7	4.7
sleeveless	13	16	7.7	5.8	10	6.6	9.2	7.2	5.8	5.8
blouse	35	36	34	34	34	34	33	33	34	34
tops	12	16	7.4	5.7	9.5	5.2	8.7	7.5	5.7	5.7
size	15	17	8.4	7.8	13	9.6	11	9.1	7.8	7.8
36	8.3	17	8.7	7.7	12	8.5	9.3	8.5	7.7	7.7
sl36i	9.6	16	7.7	6.3	11	7.3	8.8	7.4	6.3	6.3
featherlite	0	19	13	12	15	12	13	12	12	12
ladies										



ASIN : B074HXKRSG

Brand : Nanon

euclidean distance from input : 2.4383475698772945

=====

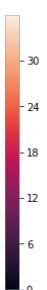
=====

=====

=====

featherlite ladies long sleeve oxford shirt light blue xlarge

re	19	19	19	20	22	19	22	19	19
blue red	15	16	16	15	20	16	18	16	15
2	1.9	7.1	5	6.1	18	5.8	11	6.4	1.9
lao	28	27	28	27	25	27	29	26	28
laos	24	25	24	25	30	25	27	25	24
laotian	4.7	9	0	7.3	18	7.4	11	7.5	4.7
sleeve	5.8	9.6	7.3	0	18	6.1	12	7.5	5.8
oxford	34	33	34	34	36	33	35	33	34
shirt	5.7	9.1	7.4	6.1	17	0	12	6.9	5.7
light	7.8	9.7	8.6	9.4	19	9.2	13	9.9	7.8
blue	7.7	10	8.6	8.6	18	7.9	11	5.4	7.7
xlarge	6.3	9.6	7.5	7.5	17	6.9	11	0	6.3
featherlite	12	14	13	13	20	12	14	9.6	12
ladies									



ASIN : B003BSQPW0

Brand : FeatherLite

euclidean distance from input : 2.4403316979623444

=====

=====

=====

=====

[10.2] Keras and Tensorflow to extract features

In [58]:

```
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
```

Using TensorFlow backend.

In [0]:

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

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

# This code 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()

'''
```

[10.3] Visual features based product similarity.

In [59]:

```
#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_apperial_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_train[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.amazon.com/dp/' + asins[indices[i]])
```

get_similar_products_cnn(12345, 20)



Product Title: mahogany girl tshirt swarovski purple clear crystals
Euclidean Distance from input image: 0.07654656
Amazon Url: www.amazon.com/dp/B01K5Q5PGW



Product Title: district juniors vintage wash vneck teem black dt4501
Euclidean Distance from input image: 44.158707
Amazon Url: www.amazon.com/dp/B00TSNTQI2



Product Title: district juniors vintage wash vneck teexl warm grey dt4501
Euclidean Distance from input image: 44.929653
Amazon Url: www.amazon.com/dp/B00TSNY902



Product Title: levis womens womens perfect graphic tee pins print jet black tshirt
Euclidean Distance from input image: 46.451305
Amazon Url: www.amazon.com/dp/B071V93RRN



Product Title: district juniors vintage wash vneck tee4xl deep turquoise dt4501
Euclidean Distance from input image: 47.209816
Amazon Url: www.amazon.com/dp/B00TSNVHZC



Product Title: district womens comfortable triblend crewneck tshirt mari hecha he3xl
Euclidean Distance from input image: 47.58396
Amazon Url: www.amazon.com/dp/B01BKCZTPW



Product Title: port company womens soft perfect pique polo shirtyellowmedium
Euclidean Distance from input image: 48.71448
Amazon Url: www.amazon.com/dp/B00GYZM6Z2





Product Title: authentic pigment ladies best summer pocket tank nautical red xlarge
Euclidean Distance from input image: 49.52847
Amazon Url: www.amazon.com/dp/B01GESXJ3G



Product Title: womens basic short sleeve vneck tee shirt cotton blend junior plus sizes charcoal small
Euclidean Distance from input image: 49.762405
Amazon Url: www.amazon.com/dp/B00JPT2PRS



Product Title: two vince camuto womens striped tribal beat shirt ultra white small
Euclidean Distance from input image: 49.779545
Amazon Url: www.amazon.com/dp/B01H24W0IG



Product Title: harriton womens breathable double needle polo shirt black xxlarge
Euclidean Distance from input image: 50.382065
Amazon Url: www.amazon.com/dp/B0038JFOU6



Product Title: devon jones womens ycollar polo shirt espresso xxxlarge
Euclidean Distance from input image: 50.438766
Amazon Url: www.amazon.com/dp/B01GGACDZ2





Product Title: levis womens womens slim crew neck slasher jet black tshirt
Euclidean Distance from input image: 50.58456
Amazon Url: www.amazon.com/dp/B071YMRVSP



Product Title: levis womens womens perfect graphic tee wink jet black tshirt
Euclidean Distance from input image: 50.737858
Amazon Url: www.amazon.com/dp/B072L2HS3V



Product Title: nili lotan womens normandy blouse black xsmall
Euclidean Distance from input image: 50.823578
Amazon Url: www.amazon.com/dp/B0736D8BVV



Product Title: miraclebody womens jersey slimming tunic top black
Euclidean Distance from input image: 51.06024
Amazon Url: www.amazon.com/dp/B0059GPDDE



Product Title: fox womens flipped tank top size small color black
Euclidean Distance from input image: 51.328533
Amazon Url: www.amazon.com/dp/B00MONRNOM





Product Title: elizabeth james womens kim gathered front tank top purple xsmall
Euclidean Distance from input image: 51.410038
Amazon Url: www.amazon.com/dp/B0717B9WYS



Product Title: dna rhea long sleeve knit top sm 1012 currant purple
Euclidean Distance from input image: 51.618755
Amazon Url: www.amazon.com/dp/B0755KBYW3



Product Title: authentic pigment ladies true spirit raglan tshirt nautical red xsmall
Euclidean Distance from input image: 51.621075
Amazon Url: www.amazon.com/dp/B01GESXQ8E

Assignment

[9.6] Weighted similarity using Brand , Color , Image and Text

In [68]:

```
# 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 hyphen  
brands = [x.replace(" ", "-") for x in data['brand'].values]  
colors = [x.replace(" ", "-") for x in data['color'].values]  
  
brand_vectorizer = CountVectorizer()  
brand_features = brand_vectorizer.fit_transform(brands)  
  
color_vectorizer = CountVectorizer()  
color_features = color_vectorizer.fit_transform(colors)
```

In [69]:

```
def heat_map(sentance1, sentance2, url, doc_id1, doc_id2, df_id1, df_id2, model):  
  
    # sentance1 : title1, input apparel  
    # sentance2 : title2, recommended apparel  
    # url: apparel image url  
    # doc_id1: document id of input apparel  
    # doc_id2: document id of recommended 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/avg) of length
h 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/avg) of length
h 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_id1]], # input
apparel's features
               [data['asin'].loc[df_id2], brands[doc_id2], colors[doc_id2], types[doc_id2]]] # recommended apparel's features

colorscale = [[0, '#1d004d'], [.5, '#f2e5ff'], [1, '#f2e5d1']] # to color the headings 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])
# plotting 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 lens 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()

```

Weights assigned to features

1. Text feature - 5
2. Color feature - 10
3. Brand feature - 15
4. Image feature - 20

In [81]:

```

def idf_w2v(doc_id, w1, w2, w3, w4, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for w2v features
    # w2: weight for brand and color features

    # pairwise_dist will store the distance from given input apparel to all remaining apparels
    # 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
    text_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))
    color_dist = pairwise_distances(color_features, color_features[doc_id])

```

```

brand_dist = pairwise_distances(brand_features, brand_features[doc_id])
image_dist = pairwise_distances(bottleneck_features_train, bottleneck_features_train[doc_id].reshape(1,-1))
pairwise_dist = (w1 * text_dist + w2 * color_dist + w3 * brand_dist + w4 * image_dist)/float(w1 + w2 + w3 + w4)

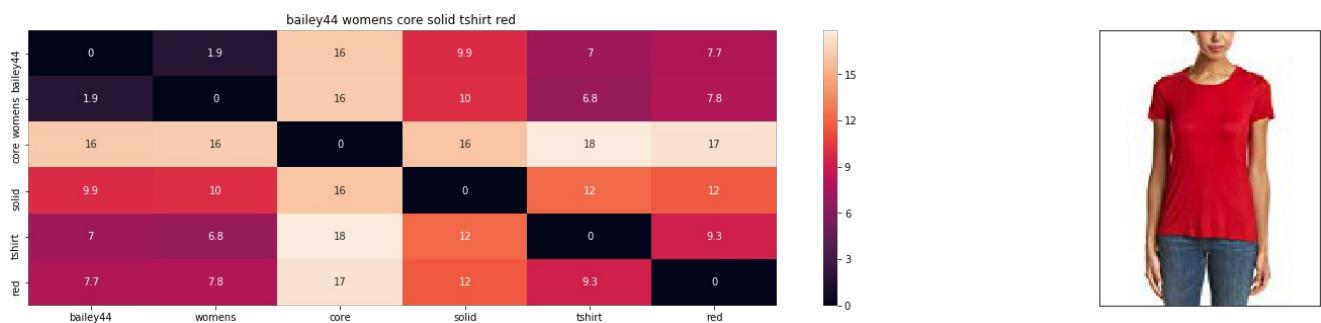
# np.argsort will return indices of 9 smallest distances
indices = np.argsort(pairwise_dist.flatten())[0:num_results]
#pdists will store the 9 smallest distances
pdists = np.sort(pairwise_dist.flatten())[0:num_results]

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

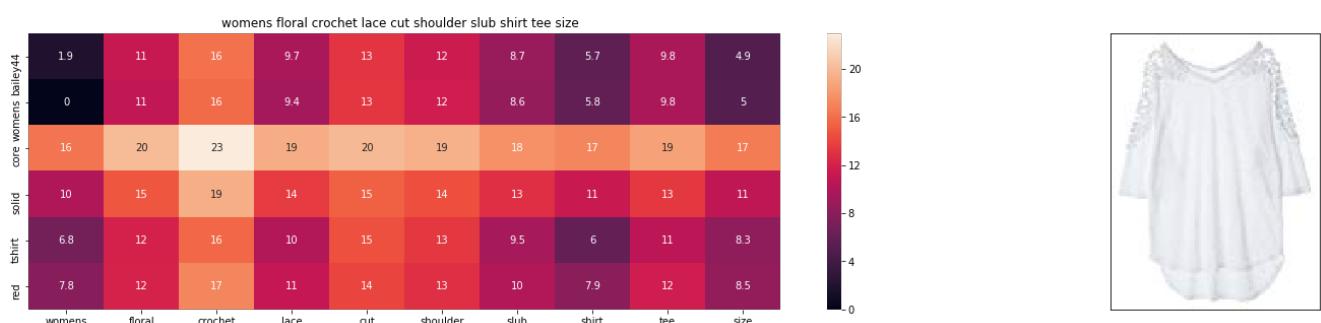
for i in range(0, len(indices)):
    heat_map(data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i], df_indices[0], df_indices[i], 'weighted')
    print('ASIN :', data['asin'].loc[df_indices[i]])
    print('Brand :', data['brand'].loc[df_indices[i]])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

idf_w2v(444, 5, 10, 15, 20, 20)
# in the give heat map, each cell contains the euclidean distance between words i, j

```



ASIN : B06Y5PL72B
 Brand : Bailey 44
 euclidean distance from input : 0.009034147262573242

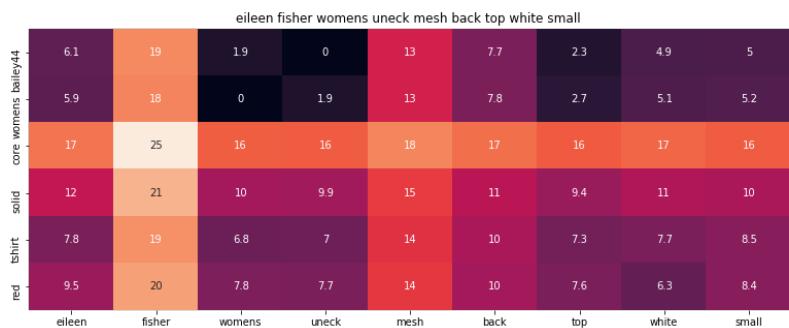


ASIN : B00YC8PK7Y

Brand : Kiki

euclidean distance from input : 15.065632393221845

=====

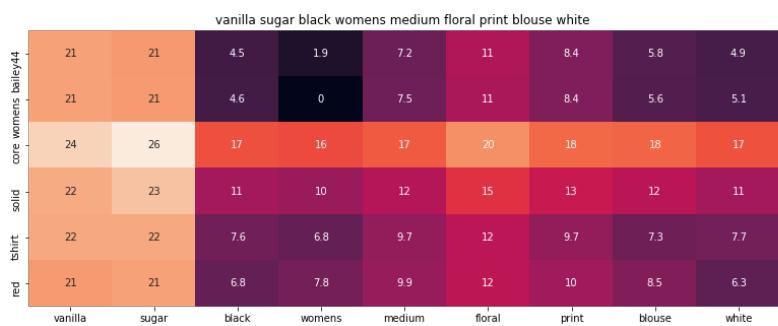


ASIN : B0728J4S62

Brand : Eileen Fisher

euclidean distance from input : 15.8746301270254

=====

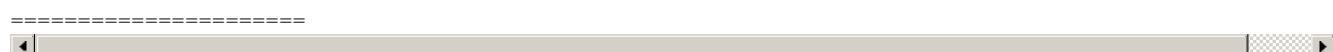


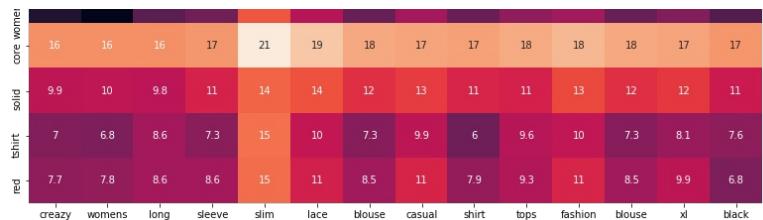
ASIN : B072K3CZW6

Brand : Vanilla Sugar

euclidean distance from input : 15.929152832103526

=====

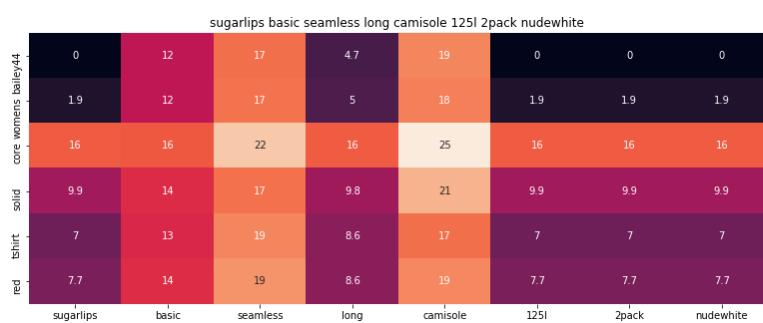




ASIN : B01EV1NKNW

Brand : Creazy

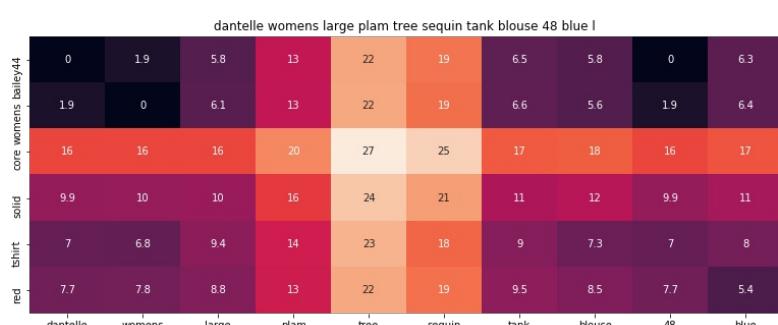
euclidean distance from input : 16.071655647616375



ASIN : B01D9B49LW

Brand : Sugar Lips

euclidean distance from input : 16.111947326660157

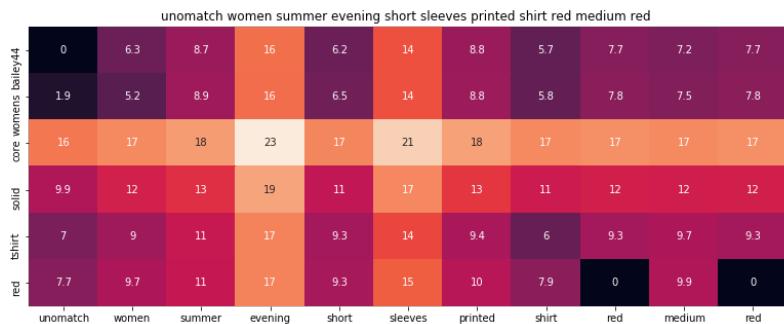


ASIN : B074P85Y4R

Brand : Dantelle

euclidean distance from input : 16.13022116885661





ASIN : B06WP5F7VW

Brand : Unomatch

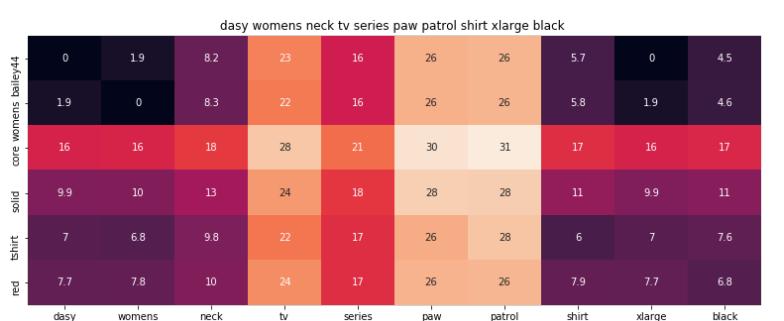
euclidean distance from input : 16.20692145564957



ASIN : B01NCV1TKU

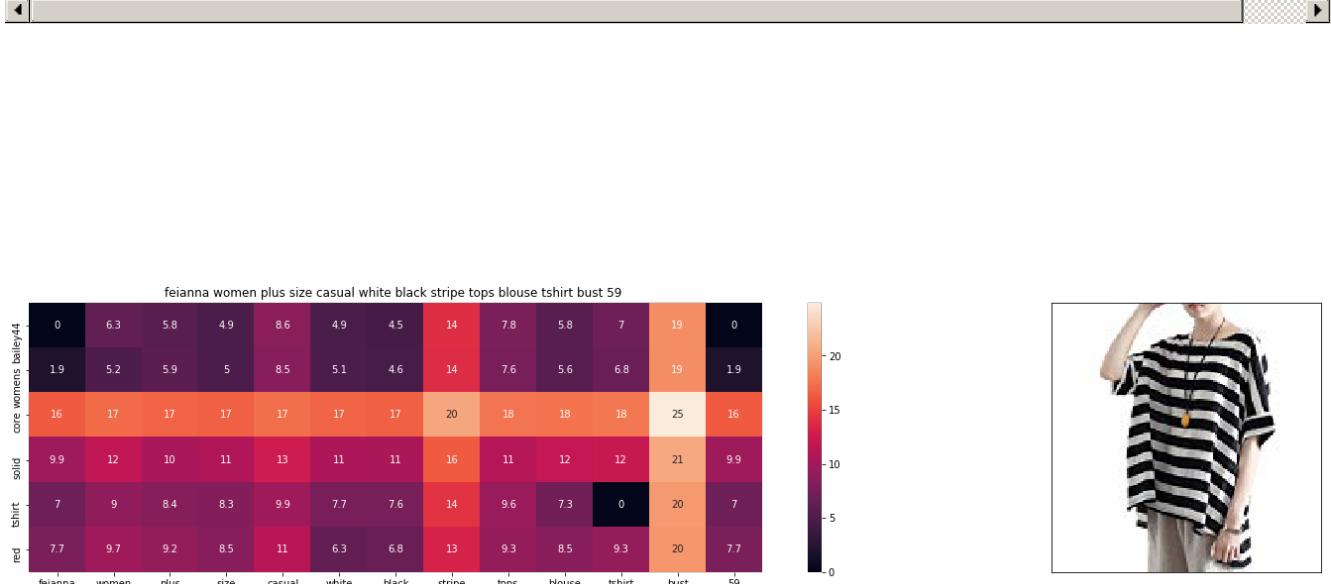
Brand : WIIPU

euclidean distance from input : 16.305874931673994

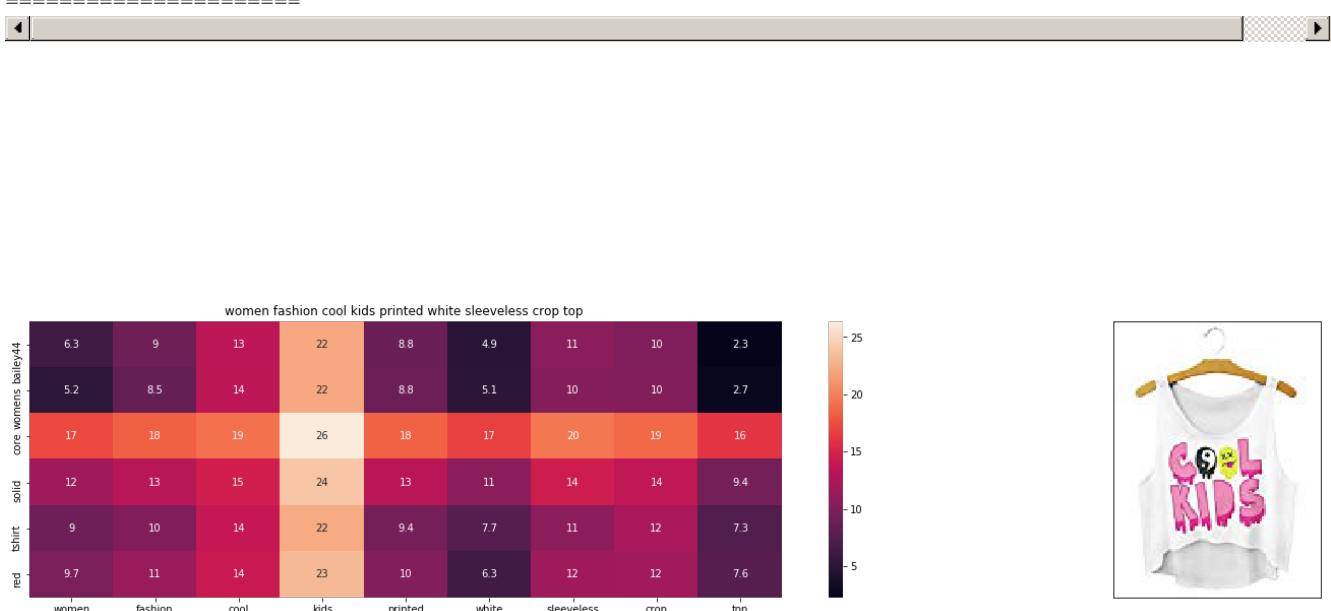


ASIN : B015J2XE56

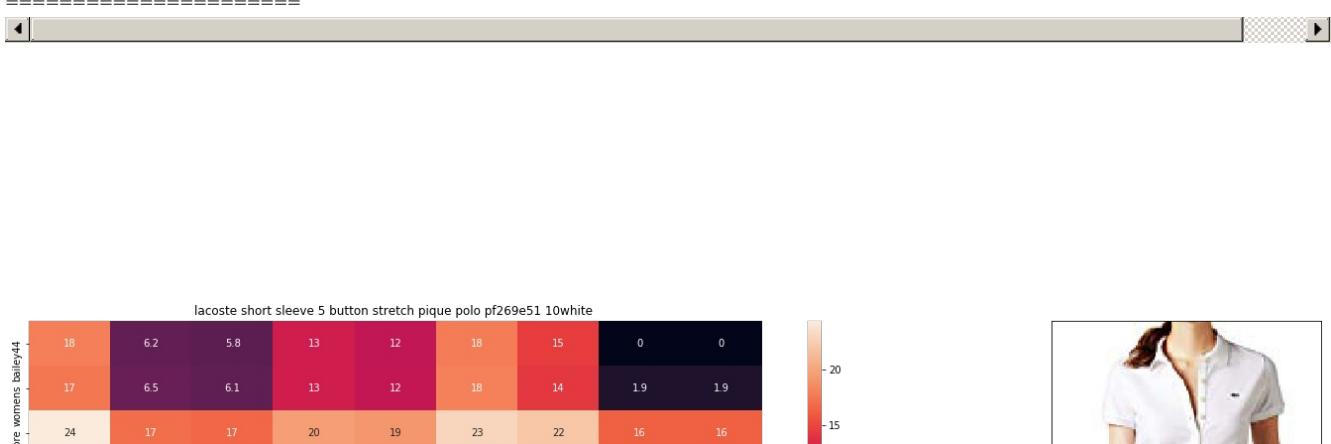
Brand : Dasy
euclidean distance from input : 16.35471305118083

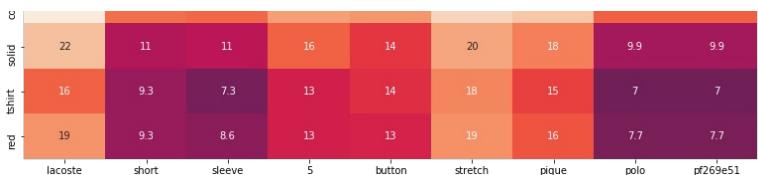


ASIN : B072WQ86QJ
Brand : FEIANNA
euclidean distance from input : 16.631052009848787



ASIN : B017UCA5GE
Brand : MKP Crop Top
euclidean distance from input : 16.636476371562395



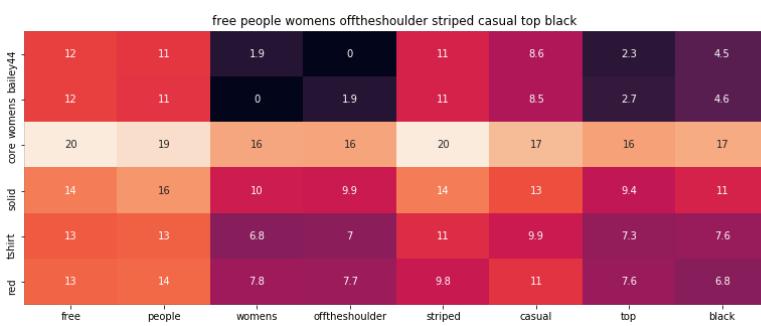


ASIN : B006WNYWT4

Brand : Lacoste

euclidean distance from input : 16.65859690890788

=====

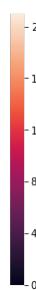
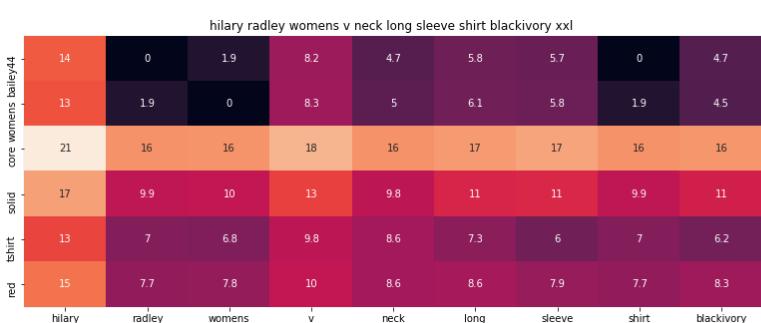


ASIN : B01HCZLYW8

Brand : Free People

euclidean distance from input : 16.703107338023937

=====



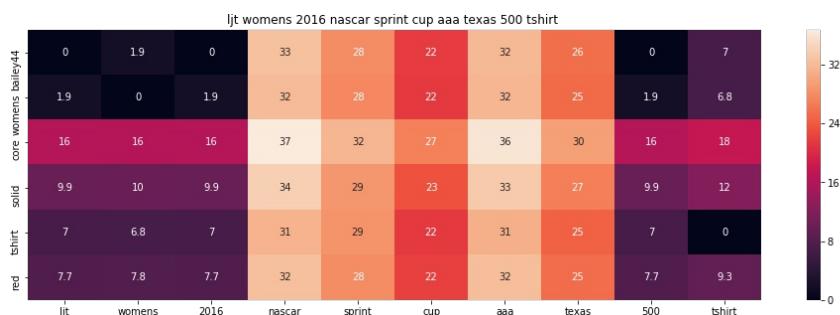
ASIN : B01MR25ZVJ

Brand : Hilary Radley

euclidean distance from input : 16.707399655837506

=====

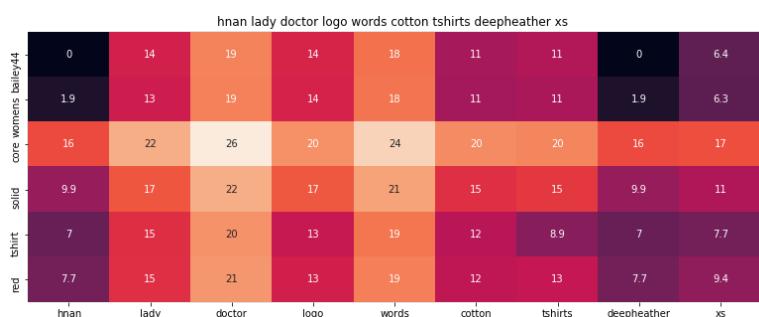




ASIN : B01IBEXL48

Brand : LaJiTong

euclidean distance from input : 16.728553077964023



ASIN : B0142LT93Q

Brand : H'nan

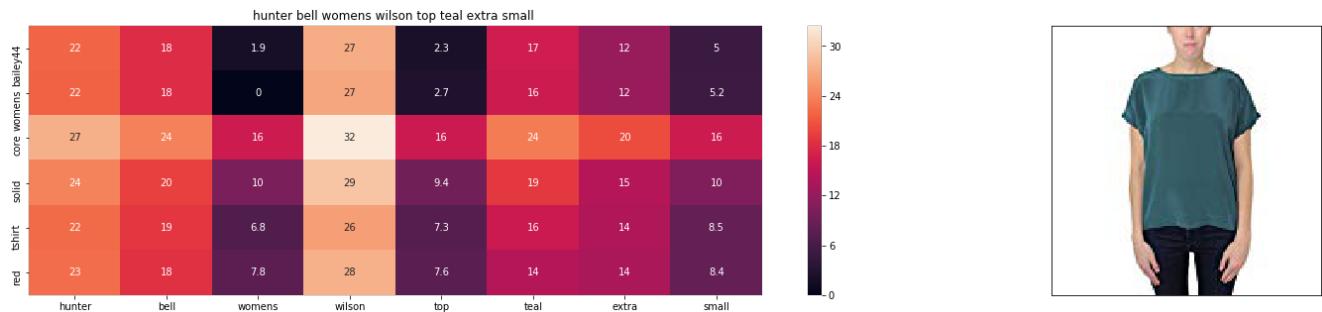
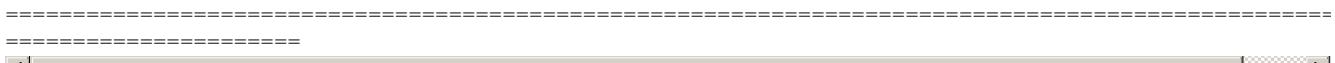
euclidean distance from input : 16.753663208346357



ASIN : B074XG4RZJ

Brand : Jess & Jane

euclidean distance from input : 16.820871429515634



ASIN : B01N2OSO76

Brand : Hunter Bell

euclidean distance from input : 16.879165496898448



ASIN : B07229JP6M

Brand : MSK

euclidean distance from input : 16.92327536807536

Step by Step procedure

- Our objective was to Recommend Apparels to user based on their current selection of Apparel on a E-commerce website
- I got the data from the Amazon website for womens shirt
- Number of data points : 183138 and Number of features/variables: 19
- There were 72 unique Product type and the most frequent type was shirt
- There were 10577 unique Brand and the most frequent Brand was zago
- I eliminated the data points which were containing NULL value for Price and color and the no of data points got reduced to 28K
- I removed data points where the item was same and the difference was just because of size or color
- I removed the data points where the titles differ only in the last few words.
- In the previous step, I sorted whole data in alphabetical order of titles. Then, I removed titles which are adjacent and very

similar title But there are some products whose titles are not adjacent but very similar, so in this step I removed those data points

10. I did some text preprocessing of text title and removed the stop words from them
11. I did text based product similarity using Bag of Words on the title data
12. Then I did text based product similarity using TF-IDF on the title data
13. Then I did text based product similarity using IDF on the title data
14. Then I did text based product similarity using Average Word2Vec on the title data
15. Then I did text based product similarity using IDF weighted Word2Vec on the title data
16. Then I did weighted Brand and Color based product similarity
17. Then I did visual features based product similarity
18. Then I combine all the above and did weighted Brand , Color, Text and Image based product similarity
19. To find the product similarity in all the above I was using Euclidean distance from the Input Data or Query Data
20. I used the function `Pairwise_distances` to compute the Euclidean distance