Amazon Apparel Recommendations

In [12]:

```
# Importing necessary Libraries

import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
import pickle
import re
import requests
from PIL import Image
from io import BytesIO

import math
import time
import os
import nltk
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize
from gensim.models import Word2Vec

from sklearn.feature_extraction.text import CountVectorizer, 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)
```

In [3]:

```
# loading the json data using pandas 'read_json' command.
data = pd.read_json('tops_fashion.json')
data.head()
```

Out[3]:

	asin	author	availability	availability_type	brand	color	editorial_reivew	editorial_review	formatted_price	large_image_url	man
0	B016I2TS4W	None	None	None	FNC7C	None	NaN	Minions Como Superheroes Ironman Women's O Nec...	None	https://images-na.ssl-images-amazon.com/images...	
1	B01N49AI08	None	None	None	FIG Clothing	None	NaN	Sizing runs on the small side. FIG® recommends...	None	https://images-na.ssl-images-amazon.com/images...	
2	B01JDPCOHO	None	None	None	FIG Clothing	None	NaN	Sizing runs on the small side. FIG® recommends...	None	https://images-na.ssl-images-amazon.com/images...	
3	B01N19U5H5	None	None	None	Focal18	None	NaN	100% Brand New & Fashion Quantity: 1 Piece...	None	https://images-na.ssl-images-amazon.com/images...	
4	B004GSI2OS	None	Usually ships in 6-10 business days	now	FeatherLite	Onyx Black/ Stone	NaN		$26.26	https://images-na.ssl-images-amazon.com/images...	

In [4]:

```
print('The shape of the data is:', data.shape)
```

```
print('\nThe columns in the data are:\n',data.columns)
```

The shape of the data is: (183138, 19)

The columns in the data are:
Index(['asin', 'author', 'availability', 'availability_type', 'brand', 'color',
'editorial_reivew', '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 apperal, ex: SHIRT/TSHIRT)
5. medium_image_url (url of the image)
6. title (title of the product.)
7. formatted_price (price of the product)

In [5]:

```
data = data[['asin', 'brand', 'color', 'medium_image_url', 'product_type_name', 'title', 'formatted_price']]
data.head(3)
```

Out[5]:

	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

Stats for product type

In [6]:

```
# count matches total count and hence no missing values
data['product_type_name'].describe()
```

Out[6]:

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

In [7]:

```
print('Shirts count for a whopping ', str((167794/183138)* 100) ,'%')
```

Shirts count for a whopping 91.62161867007393 %

In [8]:

```
unique_products = data['product_type_name'].unique()
unique_products
```

Out[8]:

array(['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'], dtype=object)

In [9]:

```
# top 10 products in terms of value
# series to list and then applying counter and getting top 10

Counter(data['product_type_name']).most_common(10)
```

Out[9]:

```
[('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)]
```

Stats for Brand

In [10]:

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

Out[10]:

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

In [11]:

```
missing_values = data.shape[0] - data['brand'].count()
missing_valuesmissing_values = data.shape[0] - data['brand'].count()
missing_values
```

Out[11]:

```
151
```

In [12]:

```
# top 10 brand types

Counter(data['brand']).most_common(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)]

Stats for color

In [13]:

```
data['color'].describe()
```

Out[13]:

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

In [14]:

```
missing_values = data.shape[0] - data['color'].count()
missing_values
```

Out[14]:

```
118182
```

In [15]:

```
# top 10 colors types

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

Out[15]:

```
[(None, 118182),
 ('Black', 13207),
 ('White', 8616),
 ('Blue', 3570),
 ('Red', 2289),
 ('Pink', 1842),
 ('Grey', 1499),
 ('*', 1388),
 ('Green', 1258),
 ('Multi', 1203)]
```

In [16]:

```
print('Black acccounts for a count of: {0:.2f}%'.format((13207/183138)* 100))
```

```
Black acccounts for a count of: 7.21%
```

Stats for formatted price

In [17]:

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

Out[17]:

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

In [18]:

```
print('Price info provided: {0:.2f}%'.format((28395/183138)* 100))
```

```
Price info provided: 15.50%
```

In [19]:

```
# top 10 price types

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

Out[19]:

```
[(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)]
```

Stats for titles

In [20]:

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

Out[20]:

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

In [21]:

```
# The number of datapoints where all are present.

data[data['formatted_price'].notnull()].count()
```

Out[21]:

```
asin                28395
brand               28302
color               28385
medium_image_url    28395
product_type_name   28395
title               28395
formatted_price     28395
dtype: int64
```

In [22]:

```
print('A total of {0:.2f}% of datapoints are null in one or the other features'.format(((183138-28395) / 183138)* 100))
```

A total of 84.50% of datapoints are null in one or the other features

Preprocessing the missing values in color and considering the datapoints with color feature

In [22]:

```
# You can download all these 183k 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[22]:

```
"\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('images/28k_images/'+row['asin']+'.jpeg')\n\n\n"
```

In [23]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183138 entries, 0 to 183137
Data columns (total 7 columns):
asin                183138 non-null object
brand               182987 non-null object
color               64956 non-null object
medium_image_url    183138 non-null object
product_type_name   183138 non-null object
title               183138 non-null object
formatted_price     28395 non-null object
dtypes: object(7)
memory usage: 9.8+ MB
```

In [24]:

```
data[['color']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183138 entries, 0 to 183137
Data columns (total 1 columns):
color    64956 non-null object
dtypes: object(1)
memory usage: 1.4+ MB
```

In [25]:

```
# filling the datapoints with a word 'None'

data1 = data.fillna(value = {'color':'None'})
```

In [26]:

```
# considering the datapoints without the word 'None'

data1 = data1[data1['color'] != 'None']
data1.head()
```

Out[26]:

	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
5	B00TAEHGGS	Fitscloth	Grape	https://images-na.ssl-images-amazon.com/images...	SHIRT	[Fits Cloth] Grape Solid Modern Long Sleeve Pl...	None
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
8	B06Y2LCC5S	Fashion2ne1	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	Standing on His Promises Rhinestones T-Shirt R...	None
11	B001LOUGE4	Fitness Etc.	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	Ladies Cotton Tank 2x1 Ribbed Tank Top	$11.99

In [27]:

```
data1.shape
```

Out[27]:

```
(64956, 7)
```

Deduplication

In [28]:

```
# returns boolean True and False. here True is considered 1 and False as 0, hece summation of all 1's
sum(data1['title'].duplicated())
```

Out[28]:

3277

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.

Part - I

In [29]:

```
data1['title'].head()
```

Out[29]:

```
4     Featherlite Ladies' Long Sleeve Stain Resistan...
5     [Fits Cloth] Grape Solid Modern Long Sleeve Pl...
6     Women's Unique 100% Cotton T - Special Olympic...
8     Standing on His Promises Rhinestones T-Shirt R...
11              Ladies Cotton Tank 2x1 Ribbed Tank Top
Name: title, dtype: object
```

In [30]:

```
# removing titles whose length of words are less than 5 as they are useless.

data1_sorted = data1[data1['title'].apply(lambda x: len(x.split()) > 4)]
```

In [31]:

```
data1_sorted['title'].head()
```

Out[31]:

```
4     Featherlite Ladies' Long Sleeve Stain Resistan...
5     [Fits Cloth] Grape Solid Modern Long Sleeve Pl...
6     Women's Unique 100% Cotton T - Special Olympic...
8     Standing on His Promises Rhinestones T-Shirt R...
11              Ladies Cotton Tank 2x1 Ribbed Tank Top
Name: title, dtype: object
```

In [32]:

```
data1_sorted.shape
```

Out[32]:

(63374, 7)

In [33]:

```
data1_sorted.sort_values(by = 'title', inplace = True)
data1_sorted.head(3)
```

Out[33]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
118987	B008D30AGK	Out+of+Print+Clothing	Multicolored	https://images-na.ssl-images-amazon.com/images...	SHIRT	"1984" Retro Book Cover Women's SLim Fit T-Shi...	$7.51
149224	B01E0XLYHA	GreaterGood	Blue	https://images-na.ssl-images-amazon.com/images...	SHIRT	"Ask Me About My Granddog" T-Shirt	None
78827	B003IDE8XQ	Maggie's Organics	Grey	https://images-na.ssl-images-amazon.com/images...	HOME	"Camisoles Grey - Medium Fair Labor, 1 pc"	$18.99

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 [34]:

```
"""iterrows means separation of a row in two parts of (index, rest of row) in series
and here we are appending all indexes in indicies list.(in short, picking all indexes)"""

indices = []
for i, j in data1_sorted.iterrows():
    indices.append(i)

indices[:5]
```

Out[34]:

[118987, 149224, 78827, 50056, 36310]

In [35]:

```
import itertools

stage1_dedupe_asins = []
i = 0
j = 0
num_data_points = data1_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 = data1['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 = data1['title'].loc[indices[j]].split()

        # store the maximum length of two strings
        length = max(len(a), len(b))
```

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

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

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

    # if the number of words in which both strings differ are > 2 , we are considering it as those two apparels are
    # different
    # if the number of words in which both strings differ are < 2 , we are considering it as those two apparels are
    # same, hence we are ignoring (removing) them.
    if (length - count) > 2: # number of words in which both sentences differ
        # if both strings differ by more than 2 words we include the 1st string index
        stage1_dedupe_asins.append(data1_sorted['asin'].loc[indices[i]])

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

In [36]:

```
data1 = data1.loc[data1['asin'].isin(stage1_dedupe_asins)]
data1.head()
```

Out[36]:

	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
5	B00TAEHGGS	Fitscloth	Grape	https://images-na.ssl-images-amazon.com/images...	SHIRT	[Fits Cloth] Grape Solid Modern Long Sleeve Pl...	None
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
16	B00L3C0O2S	FEA Merchandising	Light	https://images-na.ssl-images-amazon.com/images...	SHIRT	FEA Merchandising Juniors Florida Georgia Line...	None
17	B00XIM7A1Y	FIFA	Grey	https://images-na.ssl-images-amazon.com/images...	SHIRT	World Cup USA Heather Grey Juniors Soft Tank Top	None

In [37]:

```
data1.shape
```

Out[37]:

(48792, 7)

Part - II

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 [38]:

```
# 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 data1.iterrows():
    indices.append(i)

stage2_dedupe_asins = []
while len(indices) != 0:
    i = indices.pop()
    stage2_dedupe_asins.append(data1['asin'].loc[i])
    # consider the first apperal's title
    a = data1['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 = data1['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 appened None in case of
        # unequal strings
        # example: a =['a', 'b', 'c', 'd']
        # b = ['a', 'b', 'd']
        # itertools.zip_longest(a,b): will give [('a','a'), ('b','b'), ('c','d'), ('d', None)]
        for k in itertools.zip_longest(a,b):
            if (k[0]==k[1]):
                count += 1

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

In [39]:

```
data1 = data1.loc[data1['asin'].isin(stage2_dedupe_asins)]

print('Number of data points after stage two of dedupe: ',data1.shape[0])
```

Number of data points after stage two of dedupe: 44585

- The data has been reduced from 48792 to 44585, and saving the data onto pickle file.

In [40]:

```
# Storing these products in a pickle file

data1.to_pickle('67k_apparel_data')
```

Importing data1 as data

In [2]:

```
data = pd.read_pickle('16k_apparel_data')
print(data.shape)
data.head(3)
```

(16042, 7)

Out[2]:

	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
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	FeatherLite Ladies' Moisture Free Mesh Sport S...	$20.54

Text pre-processing

In [3]:

```
import contractions
import re
from nltk.corpus import stopwords

sw = set(stopwords.words('english'))
len(sw)
```

Out[3]:

179

In [4]:

```
# Defining our function to remove punctuations and spl chars.

def clean_sentence(text):
    text = BeautifulSoup(text, 'lxml').get_text() # removes html tags such as <br />
    text = text.lower() # converts text to lower case
    text = contractions.fix(text) # converts (don't) to (do not)
    text = text.replace('\\r\\n', '').replace('nan', '')
    text = re.sub('\W+',' ',text) # removes all special chars, punc
    text = ' '.join(e for e in text.split() if e not in sw) # removes stopwords
    text = ''.join([i for i in text if not i.isdigit()]) # removes numbers
    text = text.split()
    text1 = []
    for i in text:
        if "_" in i:
            del i # removes string if it contains '_'
        else:
            text1.append(i)
    text = " ".join(text1)
    text = ' '.join([i for i in text.split() if len(i)>1]) # https://stackoverflow.com/a/32705991/10219869

    return text
```

In [5]:

```
data['title'].head()
```

Out[5]:

```
4     Featherlite Ladies' Long Sleeve Stain Resistan...
6     Women's Unique 100% Cotton T - Special Olympic...
15     FeatherLite Ladies' Moisture Free Mesh Sport S...
27     Supernatural Chibis Sam Dean And Castiel O Nec...
46     Fifth Degree Womens Gold Foil Graphic Tees Jun...
Name: title, dtype: object
```

In [6]:

```
title = [clean_sentence(i) for i in data['title']]
title[:5]
```

Out[6]:

```
['featherlite ladies long sleeve stain resistant tapered twill shirt xl onyx black stone',
 'women unique cotton special olympics world games white size',
 'featherlite ladies moisture free mesh sport shirt white xxx large',
 'supernatural chibis sam dean castiel neck shirts female purple',
 'fifth degree womens gold foil graphic tees junior top short sleeve printed shirt']
```

In [7]:

```
data['title'] = title
data['title'].head()
```

Out[7]:

```
4     featherlite ladies long sleeve stain resistant...
6     women unique cotton special olympics world gam...
15    featherlite ladies moisture free mesh sport sh...
27    supernatural chibis sam dean castiel neck shir...
46    fifth degree womens gold foil graphic tees jun...
Name: title, dtype: object
```

Text based product similarity

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

In [8]:

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

Out[8]:

	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

In [9]:

```
# Display an image
# https://stackoverflow.com/a/23489503/10219869
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 [w1, w3, w2....]
        # values: len(values) ==  len(keys), values(i) represents the occurence of the word keys(i) [1, 2, 1,1, 0, 3....]
        # labels: len(labels) == len(keys), the values of labels depends on the model we are using like
                # [all words correspondingly]
                # if model == 'bag of words': labels(i) = values(i)
                # if model == 'tfidf weighted bag of words':labels(i) = tfidf(keys(i))
                # if model == 'idf weighted bag of words':labels(i) = idf(keys(i))
        # url : apparel's url

        # we will devide the whole figure into two parts
        gs = gridspec.GridSpec(2, 2, width_ratios=[4,1], height_ratios=[4,1])
        fig = plt.figure(figsize=(25,3))

        # 1st, ploting heat map that represents the count of commonly ocurred words in title2
        ax = plt.subplot(gs[0])

        # it displays a cell in white color if the word is intersection(lis of words of title1 and list of words of title2), in black if not
        ax = sns.heatmap(np.array([values]), annot=np.array([labels]))
        ax.set_xticklabels(keys) # set that axis labels as the words of title
        ax.set_title(text) # apparel title
```

```
    # 2nd, plotting image of the the apparel
    ax = plt.subplot(gs[1])

    # we don't want any grid lines for image and no labels on x-axis and y-axis
    ax.grid(False)
    ax.set_xticks([])
    ax.set_yticks([])

    # we call dispaly_img based with 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, 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 recommended 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(title_features[doc_id, idf_title_vectorizer.vocabulary_[x]])
            else:
                labels.append(0)

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


# 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 occurence of each word in list, it returns dict type object {word1:count}


def get_result(doc_id, text1, text2, url, model):
```

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

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

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

Bag of Words (BoW) on product titles.

In [10]:

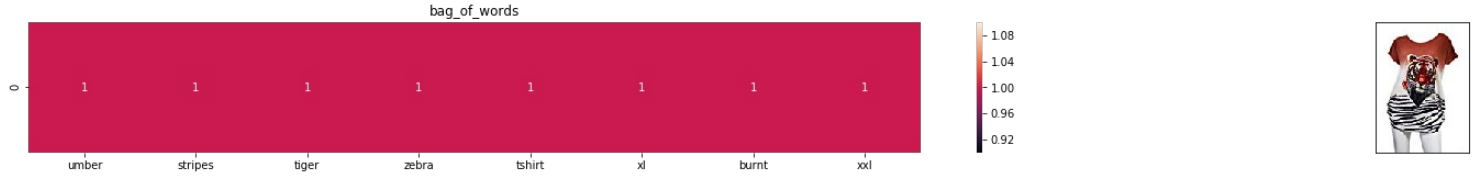
```
title_features = CountVectorizer()
title_features = title_features.fit_transform(data['title'])
title_features.get_shape()
```

Out[10]:

(16042, 12609)

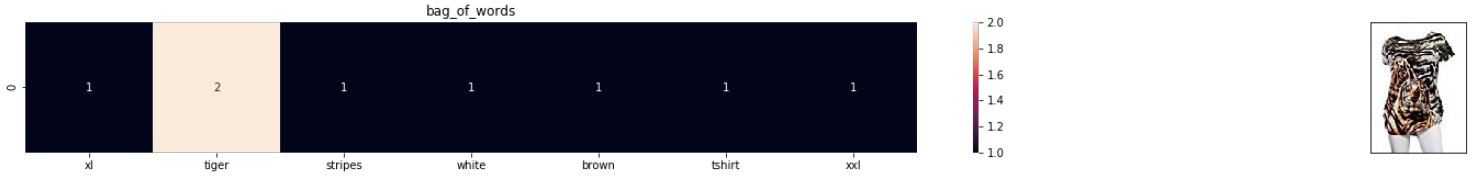
In [13]:

```
def bag_of_words_model(doc_id, num_results):

    # doc_id: apparel's id in given corpus
    # num_results implies the comparision datapoints like 20, 25, any number.

    # 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(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. text1, 3. text2, url, model
        get_result(doc_id = indices[i], text1 = data['title'].loc[df_indices[i]], text2 = data['title'].loc[df_indices[i]],
                url= data['medium_image_url'].loc[df_indices[i]], model = 'bag_of_words')
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print ('Brand:', data['brand'].loc[df_indices[i]])
        print ('Title:', data['title'].loc[df_indices[i]])
        print ('Euclidean similarity with the query image :', pdists[i])
        print('='*60)

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

#try 12566
#try 931
```

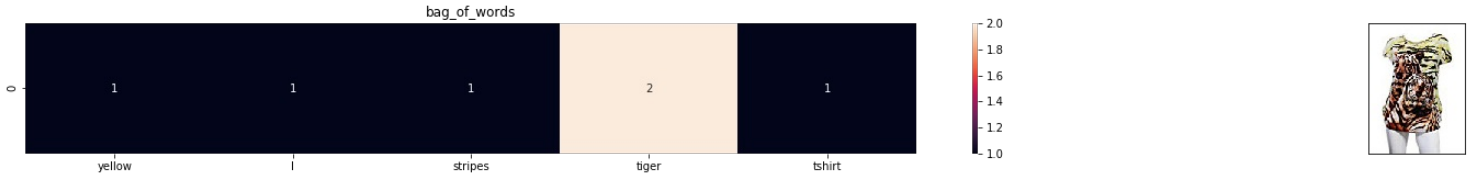


ASIN : B00JXQB5FQ
Brand: Si Row
Title: burnt umber tiger tshirt zebra stripes xl xxl
Euclidean similarity with the query image : 0.0
==

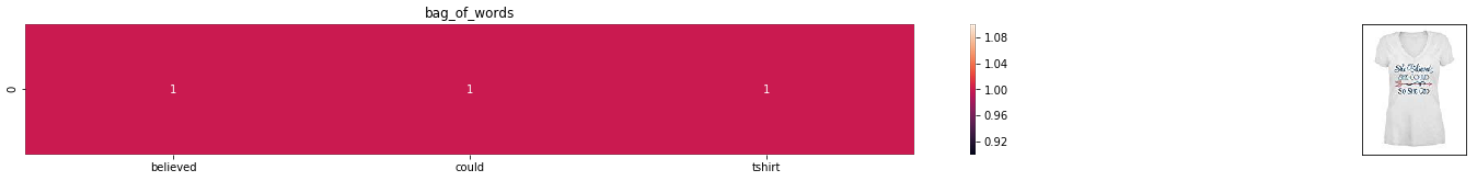
ASIN : B00JXQASS6
Brand: Si Row
Title: pink tiger tshirt zebra stripes xl xxl
Euclidean similarity with the query image : 1.7320508075688772
===



ASIN : B00JXQCWTO
Brand: Si Row
Title: brown white tiger tshirt tiger stripes xl xxl
Euclidean similarity with the query image : 2.449489742783178
===

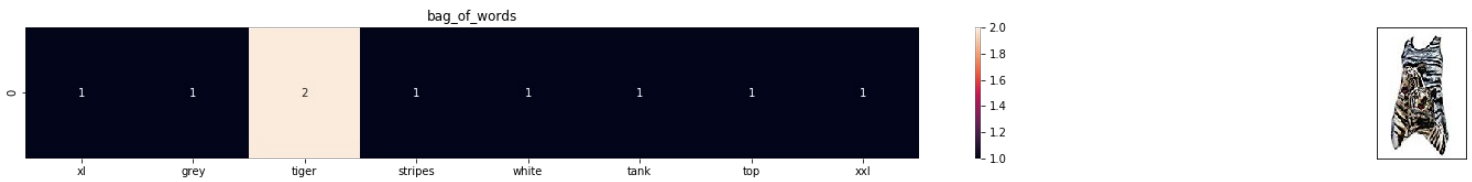


ASIN : B00JXQCUIC
Brand: Si Row
Title: yellow tiger tshirt tiger stripes l
Euclidean similarity with the query image : 2.6457513110645907
===



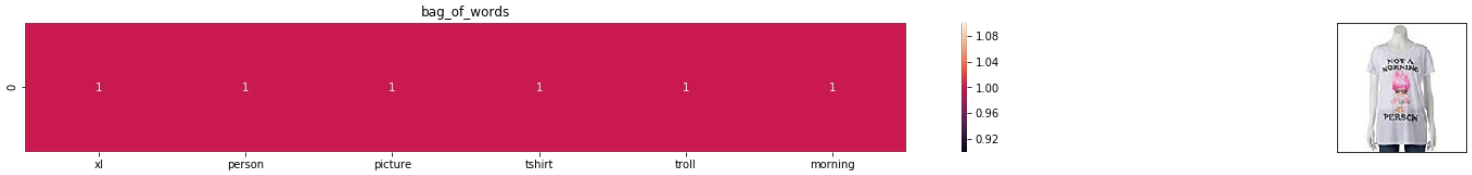
ASIN : B07568NZX4
Brand: Rustic Grace
Title: believed could tshirt
Euclidean similarity with the query image : 3.0
===

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



ASIN : B00JXQAFZ2
Brand: Si Row

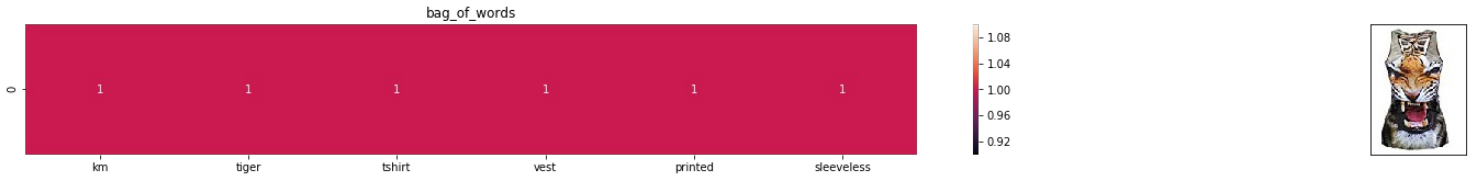
Title: grey white tiger tank top tiger stripes xl xxl
Euclidean similarity with the query image : 3.0
==========



ASIN : B01CLS8LMW
Brand: Awake
Title: morning person tshirt troll picture xl
Euclidean similarity with the query image : 3.1622776601683795
==========

ASIN : B01KVZUB6G
Brand: Merona
Title: merona green gold stripes
Euclidean similarity with the query image : 3.1622776601683795
==========

ASIN : B0733R2CJK
Brand: BLVD
Title: blvd womens graphic tshirt l
Euclidean similarity with the query image : 3.1622776601683795
==========

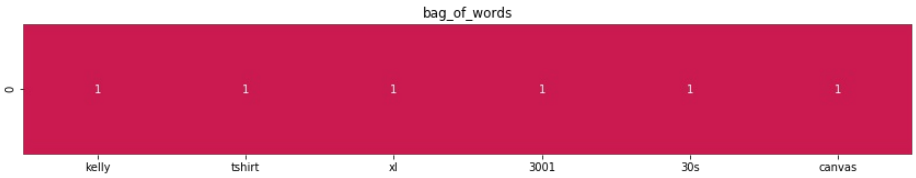


ASIN : B012VQLT6Y
Brand: KM T-shirt
Title: km tiger printed sleeveless vest tshirt
Euclidean similarity with the query image : 3.1622776601683795
==========

ASIN : B00JXQC8L6
Brand: Si Row
Title: blue peacock print tshirt l
Euclidean similarity with the query image : 3.1622776601683795
==========

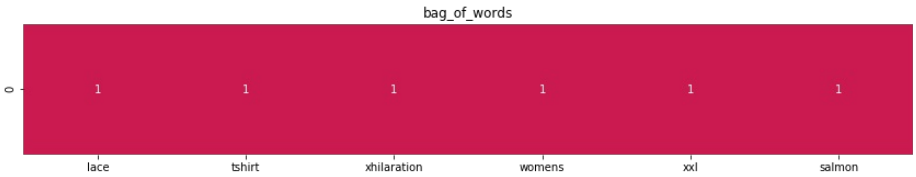
ASIN : B06XC3CZF6
Brand: Fjallraven
Title: fjallraven womens ovik tshirt plum xxl
Euclidean similarity with the query image : 3.1622776601683795
==========

ASIN : B005IT8OBA
Brand: Hetalia
Title: hetalia us girl tshirt
Euclidean similarity with the query image : 3.1622776601683795
==========



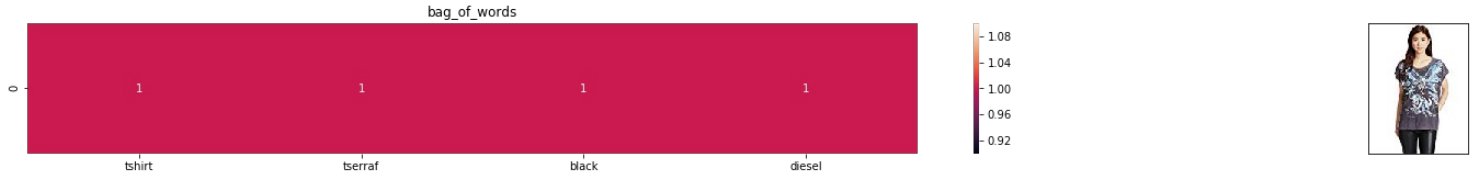
ASIN : B0088PN0LA
Brand: Red House
Title: canvas 3001 30s tshirt kelly xl
Euclidean similarity with the query image : 3.1622776601683795
==========

ASIN : B06X99V6WC
Brand: Brunello Cucinelli
Title: brunello cucinelli tshirt women white xl
Euclidean similarity with the query image : 3.1622776601683795
==========

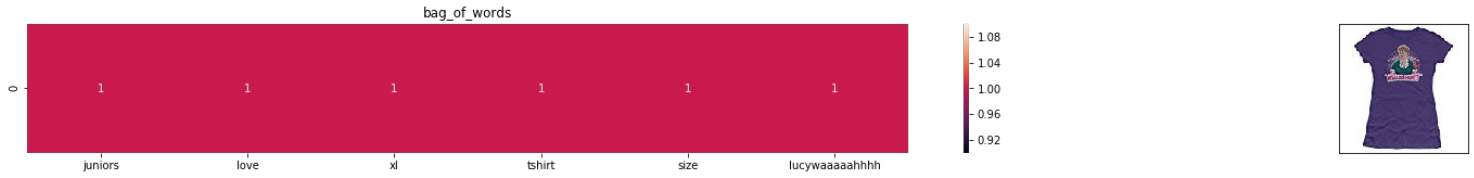


ASIN : B06Y1JPW1Q
Brand: Xhilaration
Title: xhilaration womens lace tshirt salmon xxl
Euclidean similarity with the query image : 3.1622776601683795
==========

ASIN : B06X6GX6WG
Brand: Animal
Title: animal oceania tshirt yellow
Euclidean similarity with the query image : 3.1622776601683795



ASIN : B017X8PW9U
Brand: Diesel
Title: diesel tserraf tshirt black
Euclidean similarity with the query image : 3.1622776601683795
==========



ASIN : B00IAA4JIQ
Brand: I Love Lucy
Title: juniors love lucywaaaaahhhh tshirt size xl
Euclidean similarity with the query image : 3.1622776601683795
==========

TF-IDF based product similarity

In [14]:

```
tfidf_title_vectorizer = TfidfVectorizer()
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
tfidf_title_features.get_shape()
```

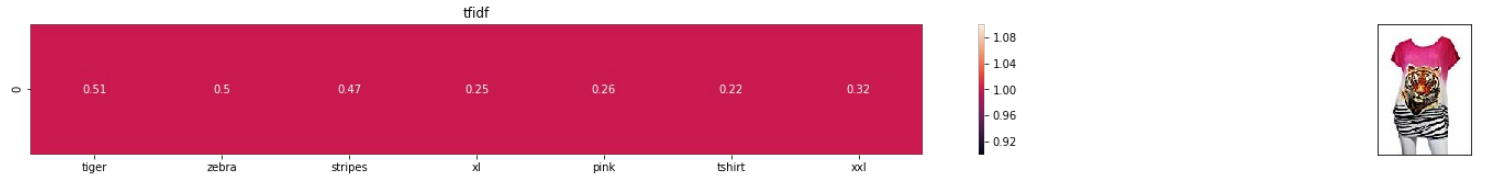
Out[14]:

(16042, 12609)

In [15]:

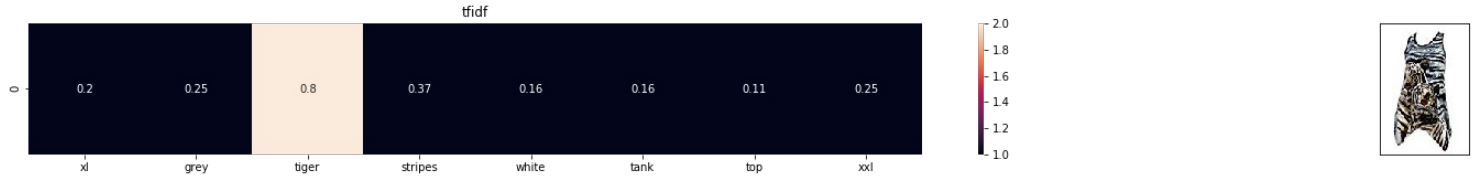
```
def tfidf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # 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. text1, 3. text2, url, model
        get_result(doc_id = indices[i], text1 = data['title'].loc[df_indices[i]], text2 = data['title'].loc[df_indices[i]],
                url= data['medium_image_url'].loc[df_indices[i]], model = 'tfidf')

        print('ASIN :',data['asin'].loc[df_indices[i]])
        print('BRAND :',data['brand'].loc[df_indices[i]])
        print ('Eucliden distance from the given image :', pdists[i])
        print('='*125)

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

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

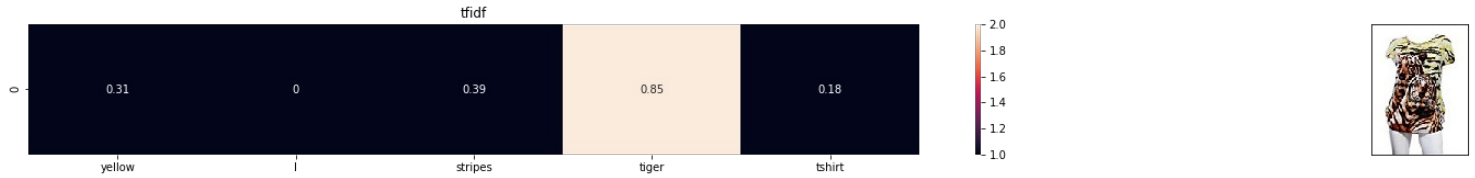


ASIN : B00JXQASS6
BRAND : Si Row
Eucliden distance from the given image : 0.7536331912451363

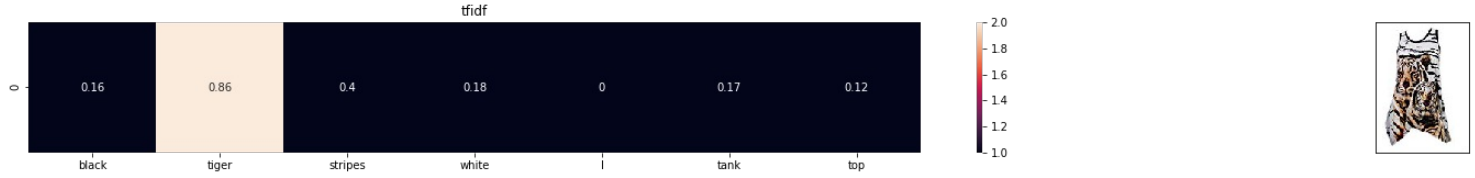
ASIN : B00JXQCWTO
BRAND : Si Row
Eucliden distance from the given image : 0.9357643943769647



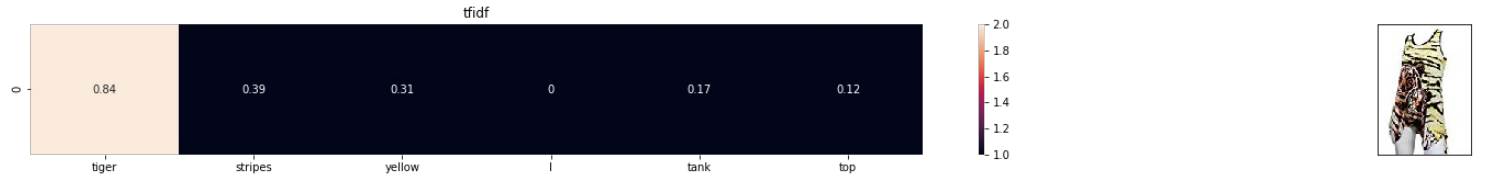
ASIN : B00JXQAFZ2
BRAND : Si Row
Eucliden distance from the given image : 0.9586153524200749



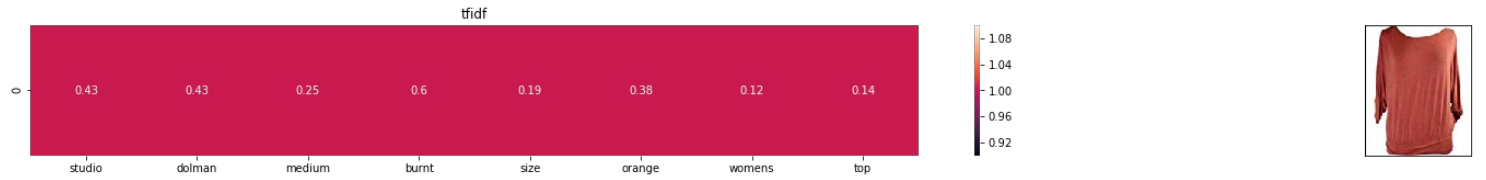
ASIN : B00JXQCUIC
BRAND : Si Row
Eucliden distance from the given image : 1.000074961446881



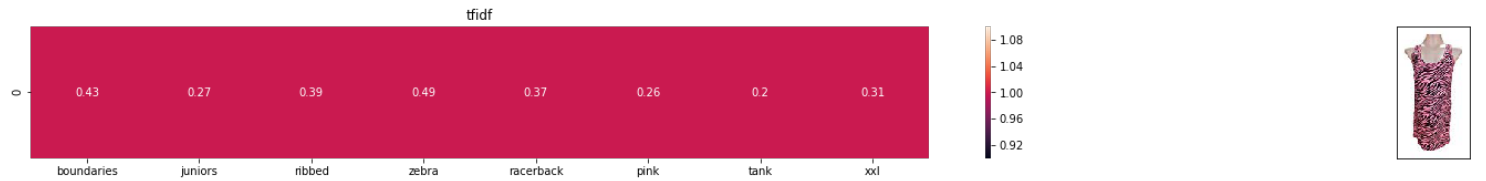
ASIN : B00JXQAO94
BRAND : Si Row
Eucliden distance from the given image : 1.023215552457452



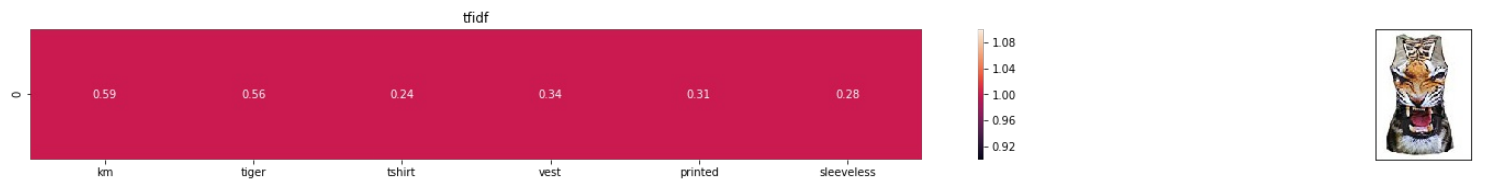
ASIN : B00JXQAUWA
BRAND : Si Row
Eucliden distance from the given image : 1.031991846303421



ASIN : B06XSCVFT5
BRAND : Studio M
Eucliden distance from the given image : 1.2106843670424716



ASIN : B06Y2GTYPM
BRAND : No Boundaries
Eucliden distance from the given image : 1.2121683810720831

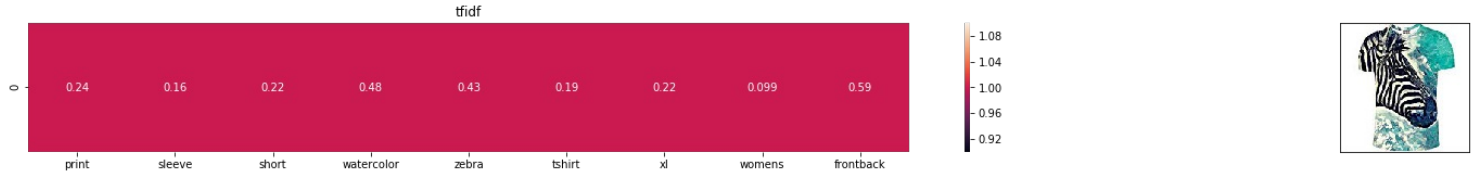


ASIN : B012VQLT6Y
BRAND : KM T-shirt
Eucliden distance from the given image : 1.219790640280982

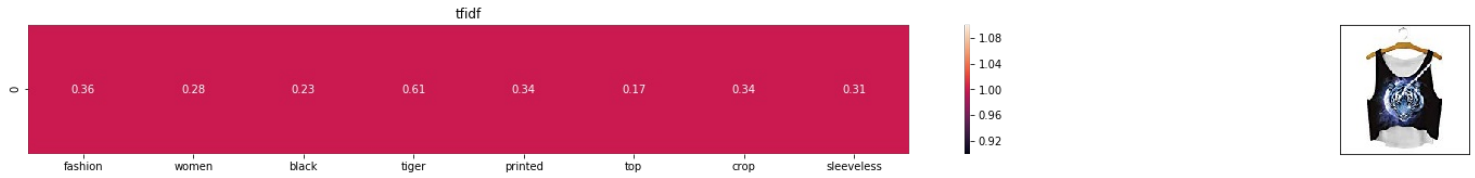
ASIN : B06Y1VN8WQ
BRAND : Black Swan
Eucliden distance from the given image : 1.2206849659998316

ASIN : B00Z6HEXWI
BRAND : Black Temptation
Eucliden distance from the given image : 1.221281392120943
==========

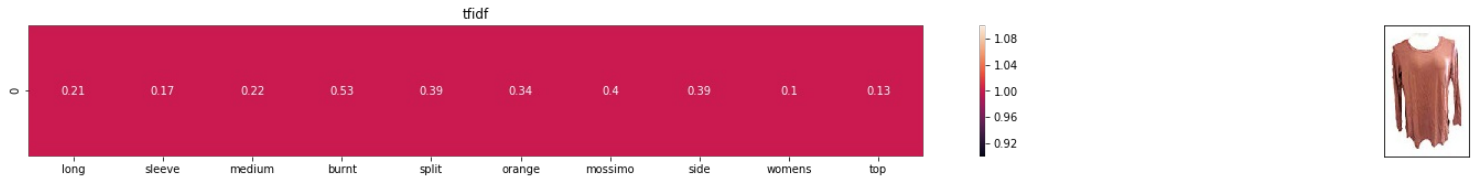
ASIN : B074TR12BH
BRAND : Ultra Flirt
Eucliden distance from the given image : 1.2313364094597743
==========



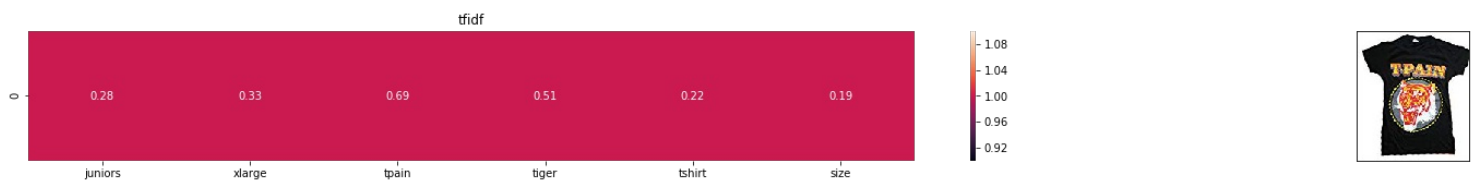
ASIN : B072R2JXKW
BRAND : WHAT ON EARTH
Eucliden distance from the given image : 1.2318451972624516
==========



ASIN : B074T8ZYGX
BRAND : MKP Crop Top
Eucliden distance from the given image : 1.2340607457359425
==========

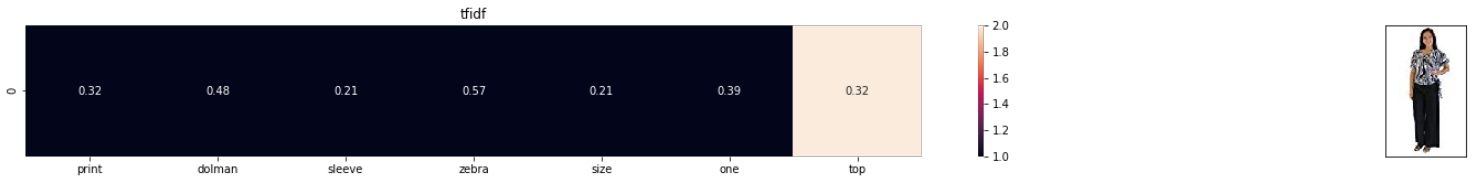


ASIN : B071ZDF6T2
BRAND : Mossimo
Eucliden distance from the given image : 1.2352785577664824
==========



ASIN : B01K0H02OG
BRAND : Tultex

Eucliden distance from the given image : 1.236457298812782
==========



ASIN : B00H8A6ZLI
BRAND : Vivian's Fashions
Eucliden distance from the given image : 1.24996155052848
==========

ASIN : B010NN9RXO
BRAND : YICHUN
Eucliden distance from the given image : 1.2535461420856102
==========

ASIN : B06XBY5QXL
BRAND : Liz Claiborne
Eucliden distance from the given image : 1.2538832938357722
==========

IDF based product similarity

In [16]:

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

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

In [17]:

```
# we need to convert the values into float
idf_title_vectorizer = CountVectorizer()
title_features  = idf_title_vectorizer.fit_transform(data['title'])
title_features  = title_features.astype(np.float)


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

    # to calculate idf_title_features we need to replace the count values with the idf values of the word
    # idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0] will return all documents in which the word
    # is present
    for j in title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0]:

        # we replace the count values of word i in document j with  idf_value of word i
```

```
    # idf_title_features[doc_id, index_of_word_in_courpus] = idf value of word
    title_features[j, idf_title_vectorizer.vocabulary_[i]] = idf_val
```

In [18]:

```
tfidf_title_vectorizer.idf_[:5]
```

Out[18]:

```
array([9.58441561, 9.98988072, 9.98988072, 9.98988072, 9.98988072])
```

In [19]:

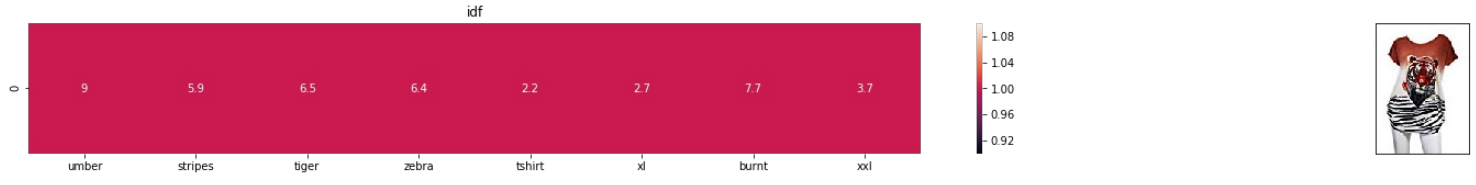
```
len(tfidf_title_vectorizer.idf_)
```

Out[19]:

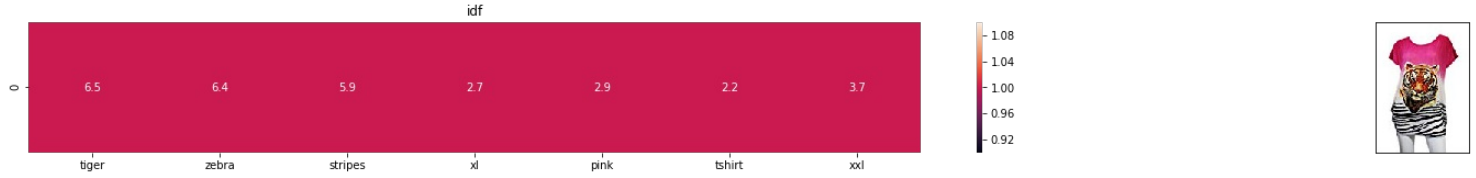
```
12609
```

In [20]:

```
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) = <X, Y> / (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = pairwise_distances(title_features, title_features[doc_id])

    # np.argsort will return indices of 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. text1, 3. text2, url, model
        get_result(doc_id = indices[i], text1 = data['title'].loc[df_indices[i]], text2 = data['title'].loc[df_indices[i]],
                url= data['medium_image_url'].loc[df_indices[i]], model = 'idf')

        print('ASIN :',data['asin'].loc[df_indices[i]])
        print('Brand :',data['brand'].loc[df_indices[i]])
        print ('euclidean distance from the given image :', pdists[i])
        print('='*125)


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

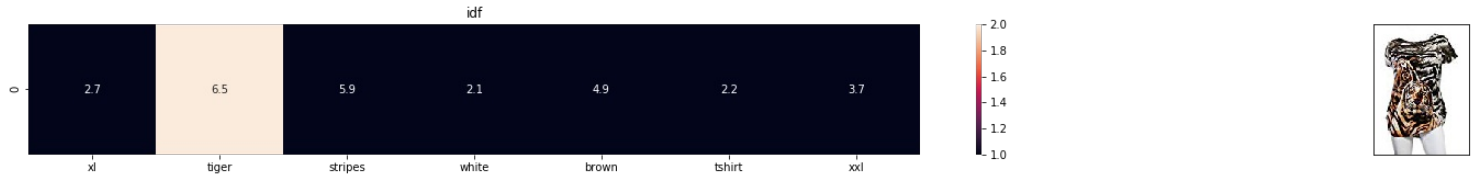


ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from the given image : 0.0
==========

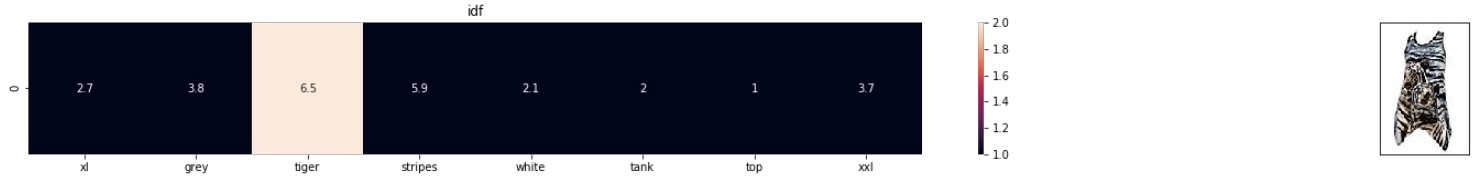


ASIN : B00JXQASS6
Brand : Si Row

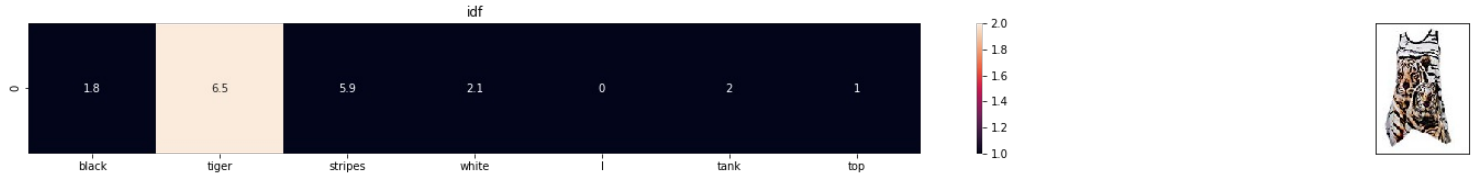
euclidean distance from the given image : 12.205071311221777



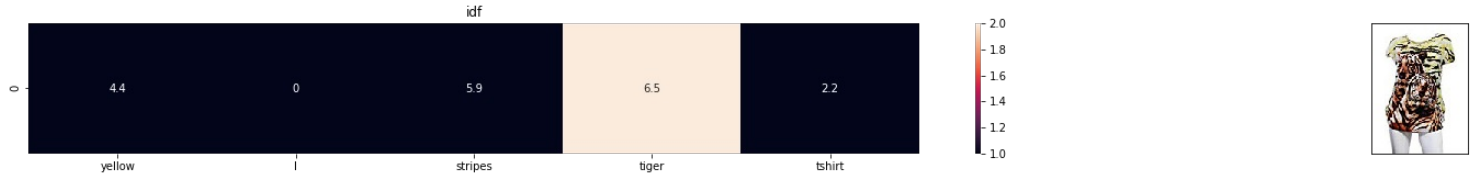
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from the given image : 14.468362685603465



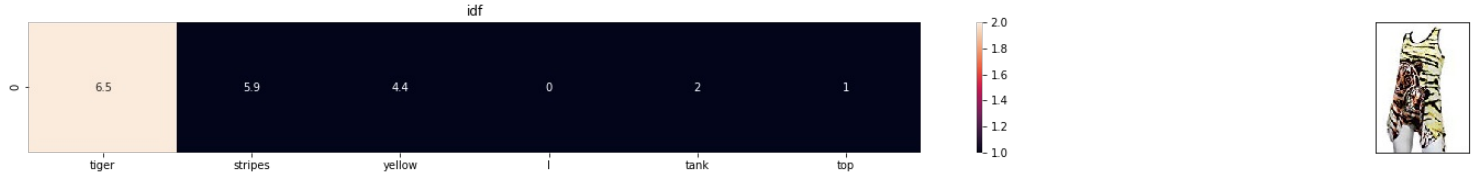
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from the given image : 14.486832924778964



ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from the given image : 14.833392966672909



ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from the given image : 14.898744516719225



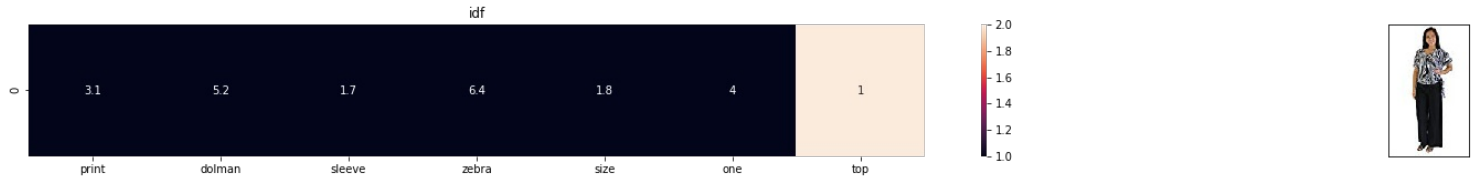
ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from the given image : 15.224458287343769

ASIN : B074T8ZYGX
Brand : MKP Crop Top
euclidean distance from the given image : 17.080812955631995

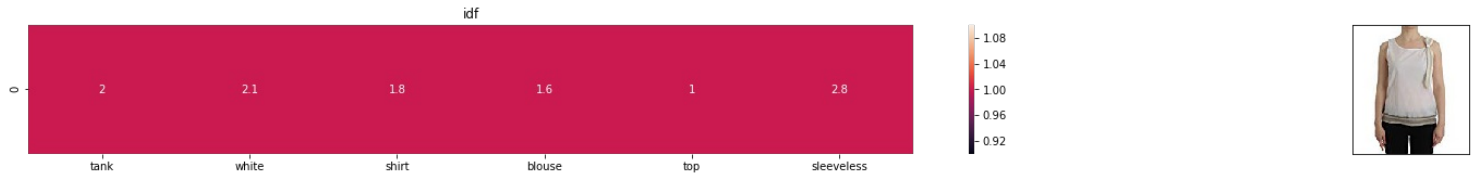
ASIN : B00KF2N5PU
Brand : Vietsbay
euclidean distance from the given image : 17.090168125645416

ASIN : B00JPOZ9GM
Brand : Sofra
euclidean distance from the given image : 17.153215337562703

ASIN : B074T9KG9Q
Brand : Rain
euclidean distance from the given image : 17.33671523874989

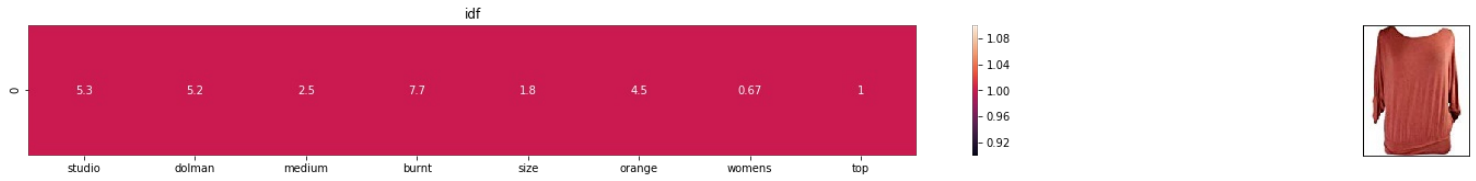


ASIN : B00H8A6ZLI
Brand : Vivian's Fashions
euclidean distance from the given image : 17.410075941001253

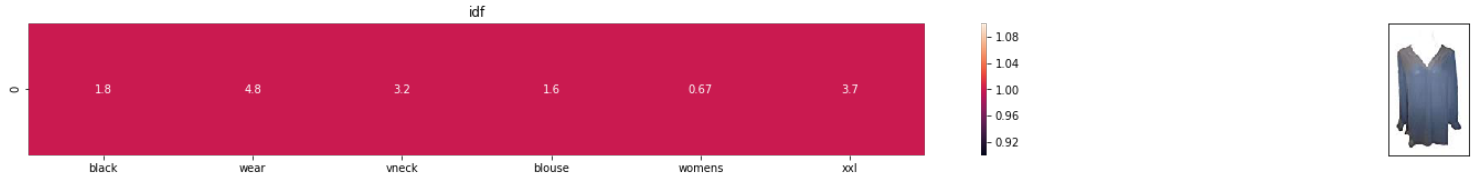


ASIN : B074G5G5RK
Brand : ERMANNO SCERVINO
euclidean distance from the given image : 17.539921335459557

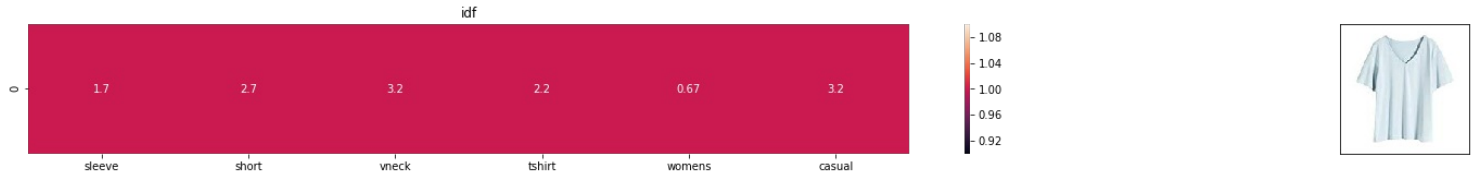
==========



ASIN : B06XSCVFT5
Brand : Studio M
euclidean distance from the given image : 17.61275854366134
==========



ASIN : B06Y6FH453
Brand : Who What Wear
euclidean distance from the given image : 17.623745282500135
==========



ASIN : B074V45DCX
Brand : Rain
euclidean distance from the given image : 17.634342496835046
==========

ASIN : B07583CQFT
Brand : Very J
euclidean distance from the given image : 17.63753712743611
==========

ASIN : B073GJGVBN
Brand : Ivan Levi
euclidean distance from the given image : 17.7230738913371
==========

ASIN : B012VQLT6Y
Brand : KM T-shirt
euclidean distance from the given image : 17.762588561202364
==========

ASIN : B00ZZMYBRG
Brand : HP-LEISURE
euclidean distance from the given image : 17.779536864674238
==========

Text Semantics based product similarity

In [21]:

```
# 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)

'''
```

Out[21]:

'\n# Set values for various parameters\nnum_features = 300   # Word vector dimensionality   \nmin_word_count = 1   # Minimum word count   \nnum_workers = 4   # Number of threads to run in parallel\ncontext = 10   # Context window size   \ndownsampling = 1e-3   # Downsample setting for frequent words\n\n# Initialize and train the model (this will take some time)\nfrom gensim.models import word2vec\nprint ("Training model...")\nmodel = word2vec.Word2Vec(sen_corpus, workers=num_workers,   size=num_features, min_count = min_word_count,   window = context)\n   \n'

In [22]:

```
documents = []
for i in title:
    i = i.split(' ')
    documents.append(i)
```

In [23]:

```
print(documents[:2])
```

[['featherlite', 'ladies', 'long', 'sleeve', 'stain', 'resistant', 'tapered', 'twill', 'shirt', 'xl', 'onyx', 'black', 'stone'], ['women', 'unique', 'cotton', 'special', 'olympics', 'world', 'games', 'white', 'size']]

In [24]:

```
from datetime import datetime
start = datetime.now()
```

```
# created my own w2v
model = Word2Vec(sentences = documents, size= 300, workers= -1, min_count= 1, window= 10)
model.save('my_w2v')
print('Time taken to complete:', datetime.now() - start)
```

Time taken to complete: 0:00:04.345044

In [25]:

```
model.wv.most_similar('cotton')
```

Out[25]:

```
[('rouge', 0.23219256103038788),
 ('foral', 0.21439394354820251),
 ('sanity', 0.20081272721290588),
 ('boobie', 0.19411848485469818),
 ('mara', 0.1904725581407547),
 ('bacon', 0.1889525055885315),
 ('gala', 0.1821260154247284),
 ('buddha', 0.18042384088039398),
 ('dame', 0.17767640948295593),
 ('mint', 0.17669783532619476)]
```

In [26]:

```
model['cotton']
```

Out[26]:

```
array([ 4.54596622e-04,  1.97915098e-04,  7.48534978e-04,  1.26417761e-03,
       -1.62527757e-03, -1.56093272e-03,  4.52998211e-05, -1.44097826e-03,
        9.04019922e-04,  1.29617983e-03,  9.71079688e-04,  2.01115705e-04,
       -1.56678422e-03, -2.13762192e-04, -8.10041674e-05, -1.29254026e-04,
        4.31444496e-04, -1.33764907e-03, -1.36185286e-03, -9.57553930e-05,
        7.76352419e-04,  1.89638027e-04,  9.97377138e-05, -3.90639267e-04,
        5.22339426e-04, -9.59995086e-04,  5.01468203e-05,  1.60948350e-03,
        6.39187230e-04, -7.39795971e-04, -2.06339013e-04, -9.12068412e-04,
       -1.79091308e-04, -2.82351655e-04, -3.03797624e-05, -1.53598306e-03,
       -5.70714823e-04, -1.48038415e-03, -1.05477183e-03,  1.64787075e-03,
       -5.72974968e-04,  9.31353308e-04, -6.79803255e-04,  1.40704634e-03,
       -7.02261925e-04, -1.43655972e-03, -9.38095734e-04, -9.02647909e-04,
       -1.12393161e-03,  5.03891970e-05,  1.62820425e-03, -1.02393830e-03,
        2.18641580e-04,  1.49677775e-03, -2.25703130e-04,  1.02725893e-03,
        1.63939397e-03,  1.18329655e-03,  3.91956215e-04, -5.58984948e-06,
       -1.61537272e-03, -3.85858235e-04,  1.57149756e-04,  1.24673091e-03,
       -1.84255390e-04, -7.08186068e-04, -1.04929483e-03, -1.02585671e-03,
       -4.27461811e-04,  3.67094617e-04,  1.47873897e-03,  1.19786221e-03,
       -7.71872510e-05, -7.48541905e-04,  1.33516779e-03,  7.33548950e-05,
       -3.57485318e-04,  6.32617623e-04, -8.25751165e-04,  6.68087960e-05,
        7.40458898e-04, -1.30972650e-03,  3.16960766e-04, -6.09271228e-04,
        4.93109284e-04,  1.56217092e-03, -8.49470263e-04, -3.65792454e-04,
       -4.81069175e-04,  1.27941906e-03, -6.62671344e-04, -1.08031568e-03,
        1.58927939e-03,  9.45930486e-04, -8.03859657e-05,  4.47372760e-04,
       -1.07764930e-03,  1.30290724e-03,  7.98216846e-04,  3.21191474e-04,
        3.43125575e-04, -1.25367590e-03, -1.48939376e-03, -1.50489563e-03,
       -5.90284530e-04, -1.24968810e-03, -8.18726316e-04, -1.71527223e-04,
       -1.51148951e-03,  1.07462762e-03,  6.38197293e-04, -1.46380789e-03,
       -9.31870425e-04, -1.58711779e-03, -1.47306433e-04,  1.20539544e-03,
        1.00848218e-03,  9.10171482e-04, -1.32770394e-03, -1.66523980e-03,
        5.96295053e-04,  1.10810087e-03,  1.14170590e-03,  6.17776299e-04,
        9.15128330e-04, -1.38296443e-03,  9.30085313e-04, -1.39551074e-03,
        8.58349144e-04, -1.07961020e-03,  1.51503703e-03, -1.57351641e-03,
        2.62300717e-04,  9.77807329e-04,  4.38628515e-04,  6.44000946e-04,
        1.39345194e-03, -2.24965643e-05, -1.35805341e-04,  8.08713317e-04,
        1.11739978e-03,  8.80743610e-04,  6.07855945e-05, -2.85934311e-05,
       -2.51595426e-04, -1.62804115e-03, -1.34504260e-03, -3.36177527e-06,
        1.48239068e-03, -3.90417226e-05,  1.05520732e-04,  2.98891915e-04,
        7.48353428e-04, -3.62608203e-04, -4.79785143e-04, -6.60721154e-04,
       -5.53995837e-04, -7.67913647e-04,  1.44911220e-03,  1.07068825e-03,
       -2.69803859e-04, -1.54099776e-03,  9.63035622e-04,  3.61724844e-04,
        7.72466010e-04,  1.47456094e-03, -9.54167626e-04,  3.21679327e-05,
        1.39153923e-03, -6.97649608e-04, -2.11082574e-04,  3.58448393e-04,
       -2.12360319e-04,  1.22502109e-03, -1.58210727e-03, -5.73638477e-04,
       -4.78544171e-05,  1.28476915e-03, -1.65790250e-03,  8.95519479e-05,
        5.45789662e-04, -1.31654378e-03,  8.54467682e-04,  1.14493196e-04,
        1.07228605e-03,  1.58168608e-03, -8.29920988e-04, -1.78005910e-04,
        1.40416599e-03,  1.12190575e-03,  8.99427047e-04,  6.33836433e-04,
```

-7.05580693e-04, -8.57177307e-04, 5.02231123e-04, 1.07463810e-03,
 1.19267043e-03, 4.54127090e-04, -3.85204097e-04, -3.19034472e-04,
 4.54424182e-04, -2.65673152e-04, 1.17384922e-03, 4.05765575e-04,
-4.01972240e-04, 1.05379231e-03, 1.32280099e-03, -1.36328372e-03,
-2.81660294e-04, 6.61321916e-04, -1.02913400e-04, -1.01676141e-03,
 1.43916204e-05, 1.03030005e-03, 1.65324530e-03, 1.34008646e-03,
-9.42388200e-04, 4.76072979e-04, -7.03197729e-04, 1.45341712e-03,
-2.40816182e-04, 1.22493005e-03, 1.93726053e-04, 1.24684500e-03,
-4.34472953e-04, 7.87897385e-04, -5.41839574e-04, -6.61653408e-04,
-7.31507200e-04, 8.31976708e-04, 1.12501346e-03, -5.39158704e-04,
-1.48330117e-03, -2.03634117e-05, 9.81673016e-04, 1.13853044e-03,
 1.15706481e-03, -1.55208376e-03, -9.27543442e-04, -1.10173004e-03,
 1.46140286e-03, 1.32857880e-03, -1.01429038e-03, -1.53240224e-04,
 1.05519209e-03, -2.09831764e-04, -3.93654394e-04, 1.54162713e-04,
-4.54118941e-04, -9.17856640e-04, -5.18986664e-04, 3.48391768e-04,
-3.78109748e-04, -3.43501277e-04, 2.26776086e-04, -9.75943170e-04,
-8.56540340e-04, 8.07592820e-04, 5.88993018e-04, -4.13151924e-04,
-1.26346922e-03, -1.50599750e-03, -8.03055475e-04, 1.60765008e-03,
-8.29665514e-04, -1.22868852e-03, 1.54768454e-03, -3.93713854e-04,
 4.80429553e-05, 1.20855880e-03, -1.63777161e-03, 5.53190650e-04,
 4.79761133e-04, -3.55946337e-04, 2.39790708e-04, 5.19535562e-04,
-1.36813417e-03, -1.59712625e-03, -9.47534980e-04, -7.28860090e-04,
-9.18073056e-04, 9.78710479e-04, -9.59375815e-04, -1.57636148e-03,
-1.37440476e-03, 1.12096046e-03, 7.53432570e-04, 1.64894562e-03,
-1.24748901e-03, 1.37607497e-03, -6.51566370e-05, 1.55430549e-04,
-4.32635745e-04, 4.90277162e-05, 1.47247931e-03, -1.57421586e-04,
 4.70484840e-04, 1.08909246e-03, -7.91928614e-04, 1.52108783e-03],
dtype=float32)

In [27]:

```
with open('word2vec_model', 'rb') as handle:
    model1 = pickle.load(handle)
```

In [28]:

```
model1['cotton']
```

Out[28]:

array([5.54199219e-02, 8.54492188e-03, -3.24218750e-01, 2.00195312e-01,
        6.88476562e-02, 7.17773438e-02, -7.99560547e-03, -2.44140625e-01,
       -1.10351562e-01, 7.61718750e-02, 2.58789062e-02, -2.50000000e-01,
       -1.24023438e-01, 1.83593750e-01, -2.96875000e-01, 2.44140625e-01,
       -1.82617188e-01, -2.51770020e-04, -1.51367188e-01, 5.90820312e-02,
       -2.89062500e-01, 5.88378906e-02, 1.42578125e-01, 1.46484375e-01,
        8.05664062e-02, 1.38549805e-02, -1.12792969e-01, -8.98437500e-02,
        4.95605469e-02, -1.37695312e-01, -1.83593750e-01, -7.76367188e-02,
       -1.56250000e-01, 7.76367188e-02, -3.94531250e-01, 5.31250000e-01,
        8.64257812e-02, -3.89099121e-03, 1.24023438e-01, 1.83593750e-01,
       -4.71191406e-02, 4.04296875e-01, 5.00488281e-02, -1.60156250e-01,
       -7.22656250e-02, -3.94531250e-01, -6.74438477e-03, -1.14257812e-01,
        6.59179688e-02, -2.57812500e-01, 1.24511719e-01, -2.57812500e-01,
       -1.40625000e-01, -7.38525391e-03, 1.35742188e-01, -2.22656250e-01,
       -9.61914062e-02, 1.72851562e-01, 1.40625000e-01, -2.08984375e-01,
        8.54492188e-03, -1.67968750e-01, -3.35937500e-01, -1.95312500e-01,
        1.41601562e-01, 2.09960938e-01, -4.29687500e-01, 7.95898438e-02,
        4.43359375e-01, 1.43554688e-01, 3.00781250e-01, -2.94921875e-01,
       -1.36718750e-01, -8.88671875e-02, -4.27734375e-01, 9.08203125e-02,
        2.33398438e-01, -2.23632812e-01, 4.19921875e-02, -9.71679688e-02,
       -2.53906250e-01, -4.83398438e-02, -6.17675781e-02, 1.12915039e-02,
        6.34765625e-02, 1.26953125e-01, 1.90429688e-01, 2.38281250e-01,
       -2.73437500e-01, 2.39257812e-01, -3.26171875e-01, 2.24609375e-01,
        1.80664062e-02, -2.83203125e-01, -1.49414062e-01, -1.35498047e-02,
        1.01074219e-01, 6.88476562e-02, 1.75781250e-01, -2.00195312e-01,
       -3.32031250e-02, -4.53125000e-01, -2.55859375e-01, -2.89306641e-02,
       -2.98828125e-01, 3.04687500e-01, 8.20312500e-02, 5.46875000e-02,
       -5.07812500e-01, -2.09960938e-01, -5.39550781e-02, 1.63085938e-01,
       -4.12597656e-02, 2.17773438e-01, -3.58886719e-02, 2.61718750e-01,
        3.54003906e-02, 7.74383545e-04, -3.17382812e-02, 1.18652344e-01,
        7.17773438e-02, 3.28125000e-01, -1.65039062e-01, -2.71484375e-01,
        1.73339844e-02, -3.75366211e-03, -1.98242188e-01, 1.39648438e-01,
        1.12304688e-01, 1.01562500e-01, -1.31835938e-01, -1.70898438e-01,
       -1.31835938e-02, 3.96728516e-04, 6.05468750e-02, -3.90625000e-02,
       -5.27343750e-01, 3.84765625e-01, 3.41796875e-02, -2.71484375e-01,
        2.09960938e-01, -2.33398438e-01, -5.12695312e-02, -3.84765625e-01,
       -1.96289062e-01, 2.20703125e-01, 1.11328125e-01, 2.77343750e-01,
       -3.78417969e-03, -1.09375000e-01, 1.41601562e-01, -2.63671875e-01,
       -1.92871094e-02, -1.66992188e-01, -8.05664062e-03, -2.69531250e-01,
       -3.51562500e-01, -2.16796875e-01, 4.73022461e-03, -3.33984375e-01,
       -6.78710938e-02, -4.00390625e-02, 2.69531250e-01, 1.14257812e-01,

```
       8.10546875e-02,  4.54101562e-02,  8.74023438e-02,  5.12695312e-03,
      -6.54296875e-02, -1.00097656e-01,  4.19921875e-02,  4.39453125e-01,
      -1.25000000e-01, -1.20605469e-01,  1.14135742e-02, -4.41406250e-01,
       7.95898438e-02, -2.01171875e-01, -5.49316406e-02,  1.26953125e-01,
      -1.47460938e-01, -1.54296875e-01, -2.56347656e-02,  3.33984375e-01,
       1.84570312e-01, -2.31445312e-01,  1.28906250e-01,  3.26171875e-01,
       2.37304688e-01,  2.96875000e-01,  1.09863281e-02,  1.37939453e-02,
      -1.17187500e-01,  1.48437500e-01,  2.41210938e-01,  9.42382812e-02,
      -2.94189453e-02, -2.27539062e-01, -1.02050781e-01, -2.48046875e-01,
       7.32421875e-02, -1.71875000e-01,  1.87500000e-01, -2.30468750e-01,
      -1.97265625e-01, -3.97949219e-02,  2.83203125e-01, -9.27734375e-02,
       1.62109375e-01,  2.75390625e-01, -1.56250000e-01,  2.25585938e-01,
       1.55273438e-01,  1.32812500e-01,  2.46093750e-01,  9.22851562e-02,
      -2.77343750e-01, -2.16796875e-01,  2.51464844e-02,  3.12500000e-01,
      -2.71484375e-01,  2.71484375e-01, -4.57031250e-01,  3.55468750e-01,
       6.54296875e-02, -2.29492188e-01,  1.20117188e-01, -7.66601562e-02,
       2.09960938e-01,  2.34375000e-02,  4.00390625e-02, -3.14941406e-02,
       1.66015625e-02,  1.60156250e-01, -3.10546875e-01, -2.71484375e-01,
       1.08398438e-01, -3.16406250e-01,  1.25976562e-01, -2.23632812e-01,
       3.08593750e-01,  1.06445312e-01,  1.11328125e-01,  1.12915039e-03,
      -1.26953125e-01, -2.77343750e-01, -8.88671875e-02, -4.10156250e-01,
      -6.93359375e-02,  2.36328125e-01,  7.42187500e-02, -1.71875000e-01,
       1.78710938e-01,  1.21093750e-01,  1.05468750e-01, -2.25585938e-01,
       1.85546875e-01, -2.26562500e-01,  8.30078125e-02, -1.77734375e-01,
       2.36816406e-02, -7.95898438e-02, -9.52148438e-02,  4.05273438e-02,
       3.92578125e-01,  5.49316406e-02, -2.59765625e-01, -1.02539062e-01,
       7.20214844e-03, -4.66308594e-02, -1.74804688e-01,  2.22656250e-01,
      -6.29882812e-02, -1.95312500e-01,  1.44531250e-01, -4.56542969e-02,
      -2.48046875e-01, -1.86523438e-01,  2.38037109e-02,  1.25000000e-01,
       2.71484375e-01,  2.04101562e-01,  3.57421875e-01, -2.28515625e-01,
      -2.67578125e-01, -1.96289062e-01, -3.55468750e-01,  9.32617188e-02,
      -5.37109375e-02,  1.88476562e-01, -3.92578125e-01, -8.64257812e-02,
       1.96289062e-01, -6.44531250e-02,  1.44531250e-01,  5.46875000e-02,
       2.49023438e-01,  2.44140625e-01,  2.92968750e-01,  3.86718750e-01],
     dtype=float32)
```

In [29]:

```
# 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(title_features[doc_id, idf_title_vectorizer.vocabulary_[i]] * model[i])
            elif m_name == 'avg':
                vec.append(model[i])
        else:
            # if the word in our courpus is not there in the google word2vec corpus, we are just ignoring it
            vec.append(np.zeros(shape=(300,)))
    # we will return a numpy array of shape (#number of words in title * 300 ) 300 = len(w2v_model[word])
    # each row represents the word2vec representation of each word (weighted/avg) in given sentance
    return  np.array(vec)

def get_distance(vec1, vec2):

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

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

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



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

    ax = plt.subplot(gs[0])
    # ploting the heap map based on the pairwise distances
    ax = sns.heatmap(np.round(s1_s2_dist,4), annot=True)
    # set the x axis labels as recommended apparels title
    ax.set_xticklabels(sentence2.split())
    # set the y axis labels as input apparels title
    ax.set_yticklabels(sentence1.split())
    # set title as recommended apparels title
    ax.set_title(sentence2)

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

    plt.show()
```

In [30]:

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

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

    featureVec = np.zeros((num_features,), dtype="float32")
    # we will intialize a vector of size 300 with all zeros
    # we add each word2vec(wordi) to this fetureVec
    nwords = 0

    for word in sentence.split():
        nwords += 1
        if word in vocab:
            if m_name == 'weighted' and word in  idf_title_vectorizer.vocabulary_:
                featureVec = np.add(featureVec, 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 sentance, its of shape (1, 300)
    return featureVec
```

Average Word2Vec product similarity.

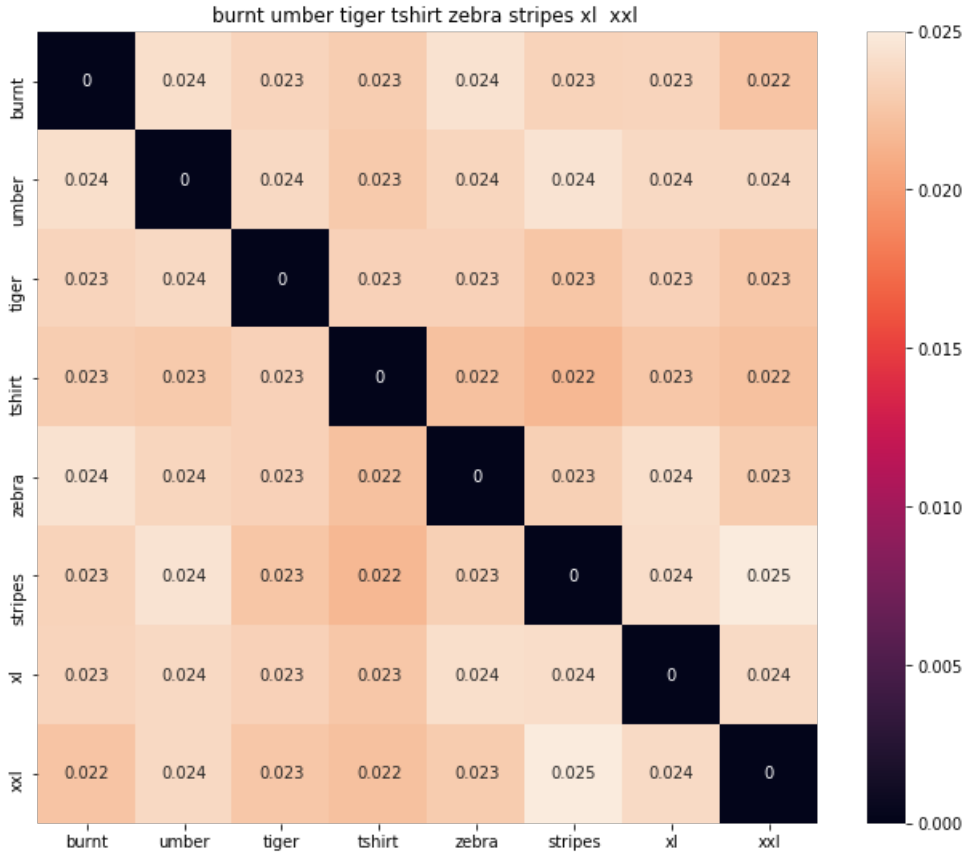
In [31]:

```
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'))
```

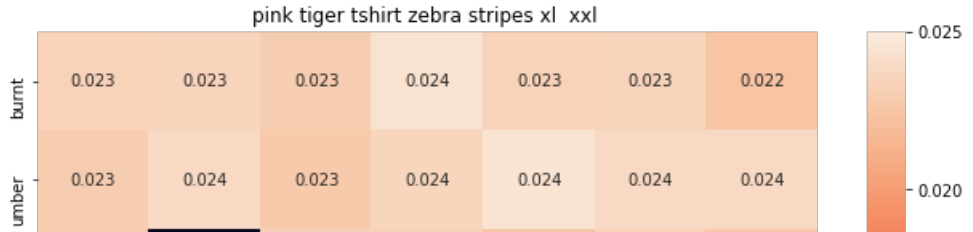
```
    doc_id += 1

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

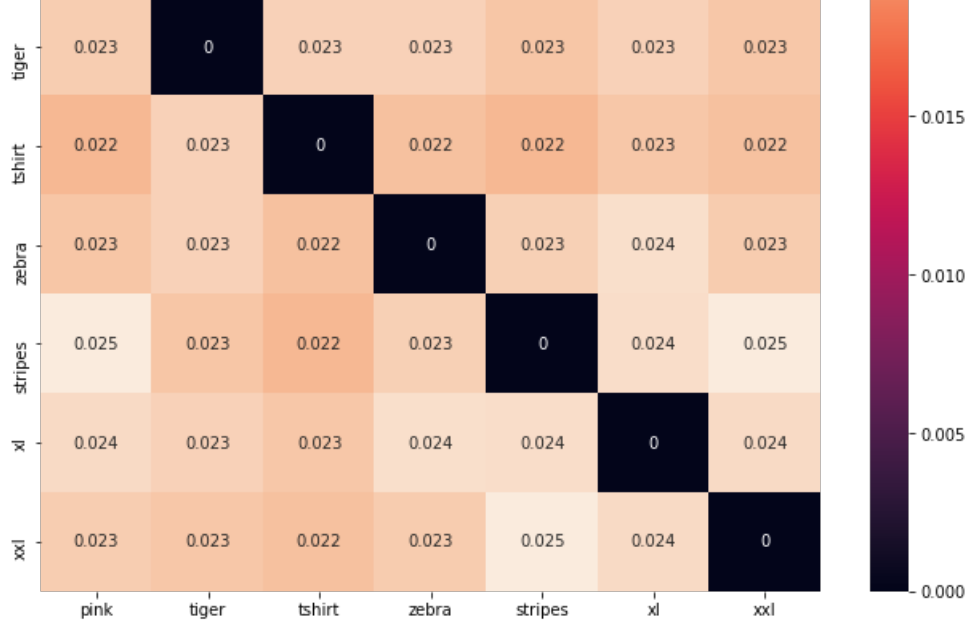
In [32]:

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

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

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

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

    for i in range(0, len(indices)):
        heat_map_w2v(data['title'].loc[df_indices[0]],data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i], 'avg')
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print('BRAND :',data['brand'].loc[df_indices[i]])
        print ('euclidean distance from given input image :', pdists[i])
        print('='*125)


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

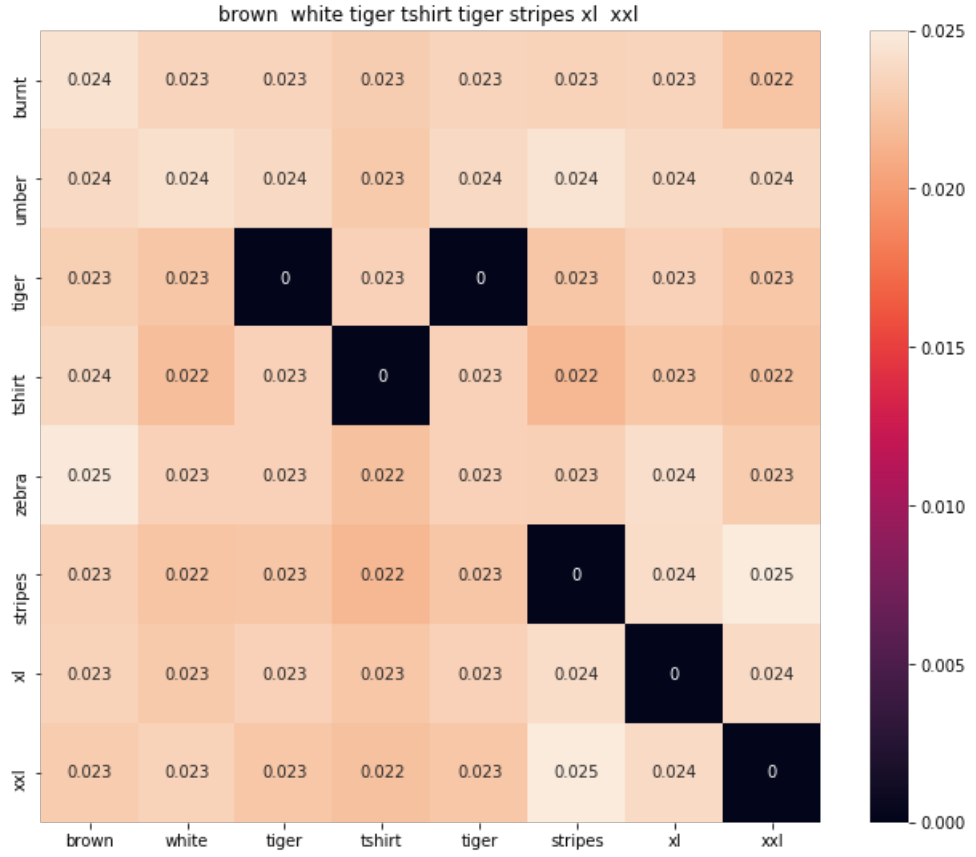


ASIN : B00JXQB5FQ
BRAND : Si Row
euclidean distance from given input image : 8.231806e-11
==========

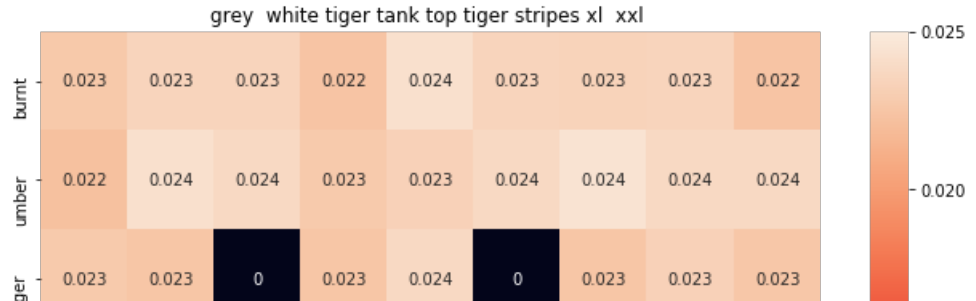


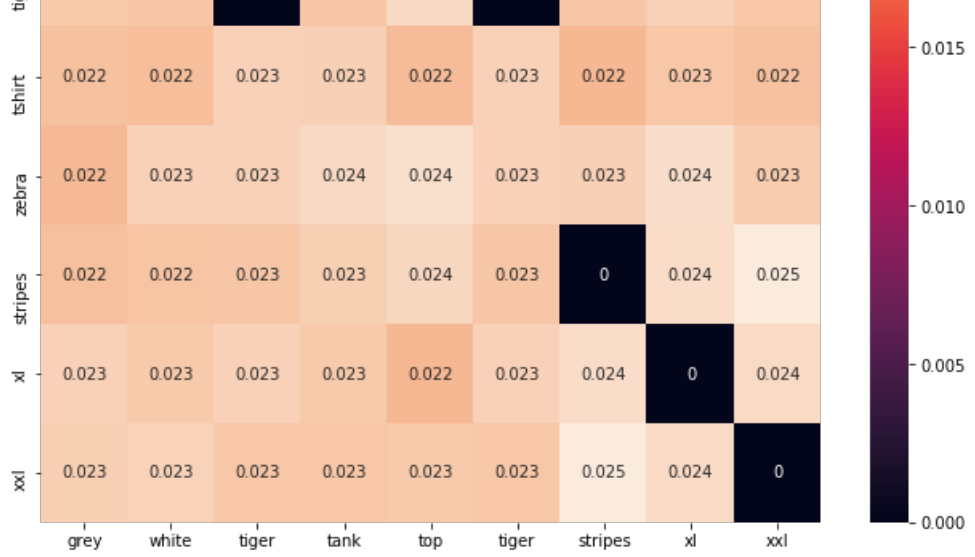


ASIN : B00JXQASS6
BRAND : Si Row
euclidean distance from given input image : 0.0037562624



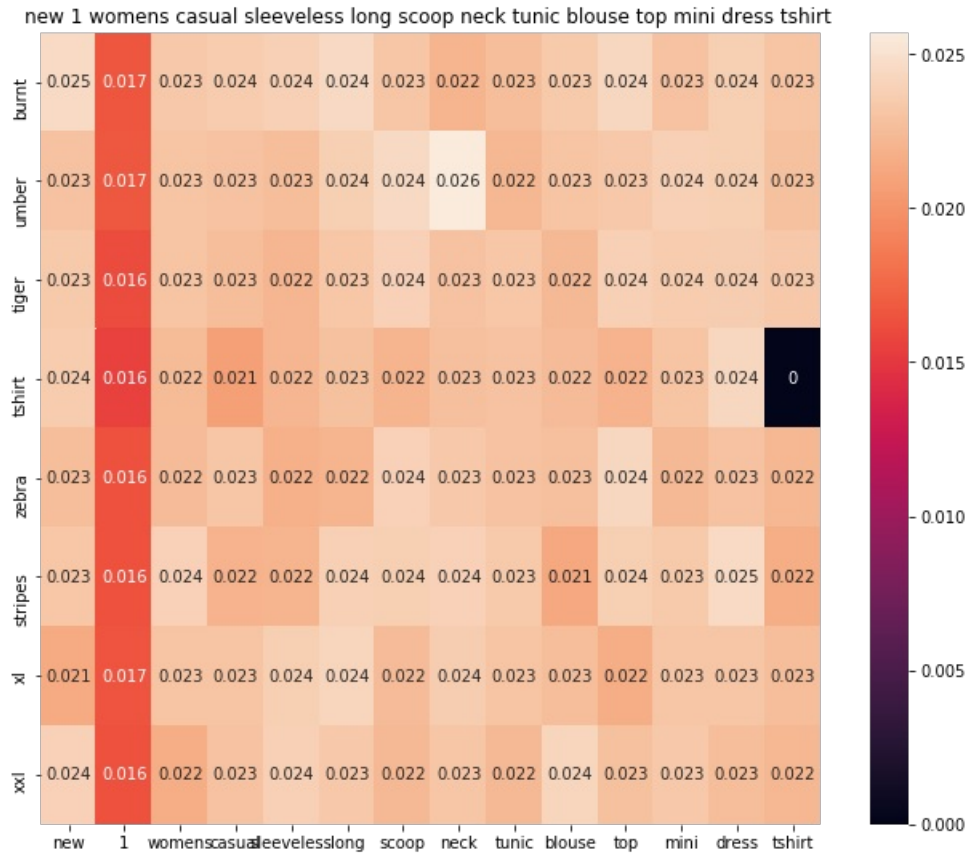
ASIN : B00JXQCWTO
BRAND : Si Row
euclidean distance from given input image : 0.0053255428





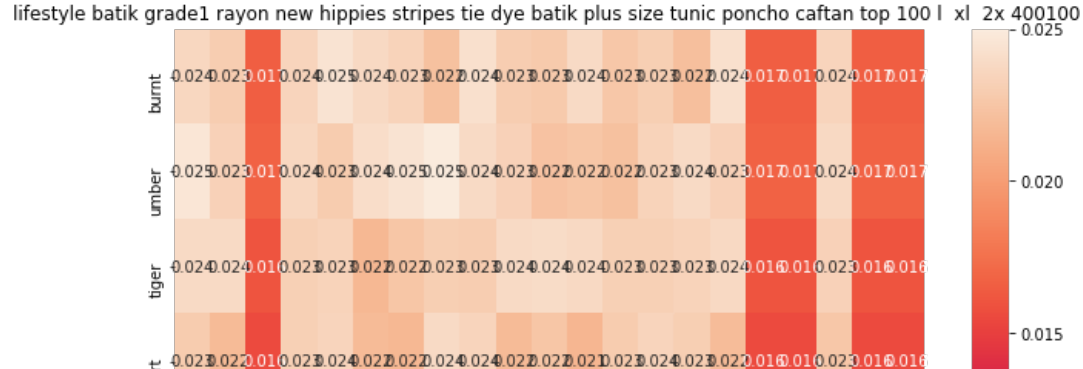
ASIN : B00JXQAFZ2
BRAND : Si Row
euclidean distance from given input image : 0.005475388

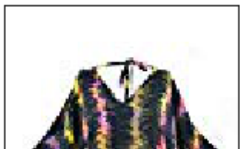
==========

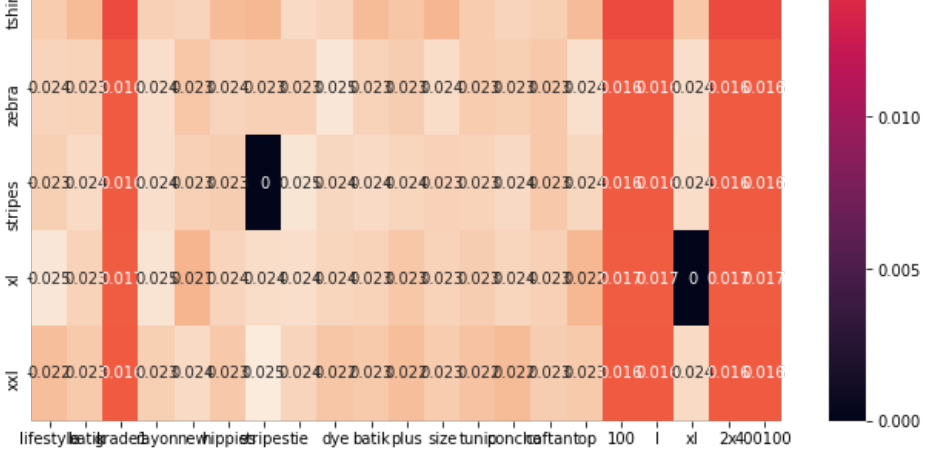


ASIN : B073CXSNM6
BRAND : General
euclidean distance from given input image : 0.005742025

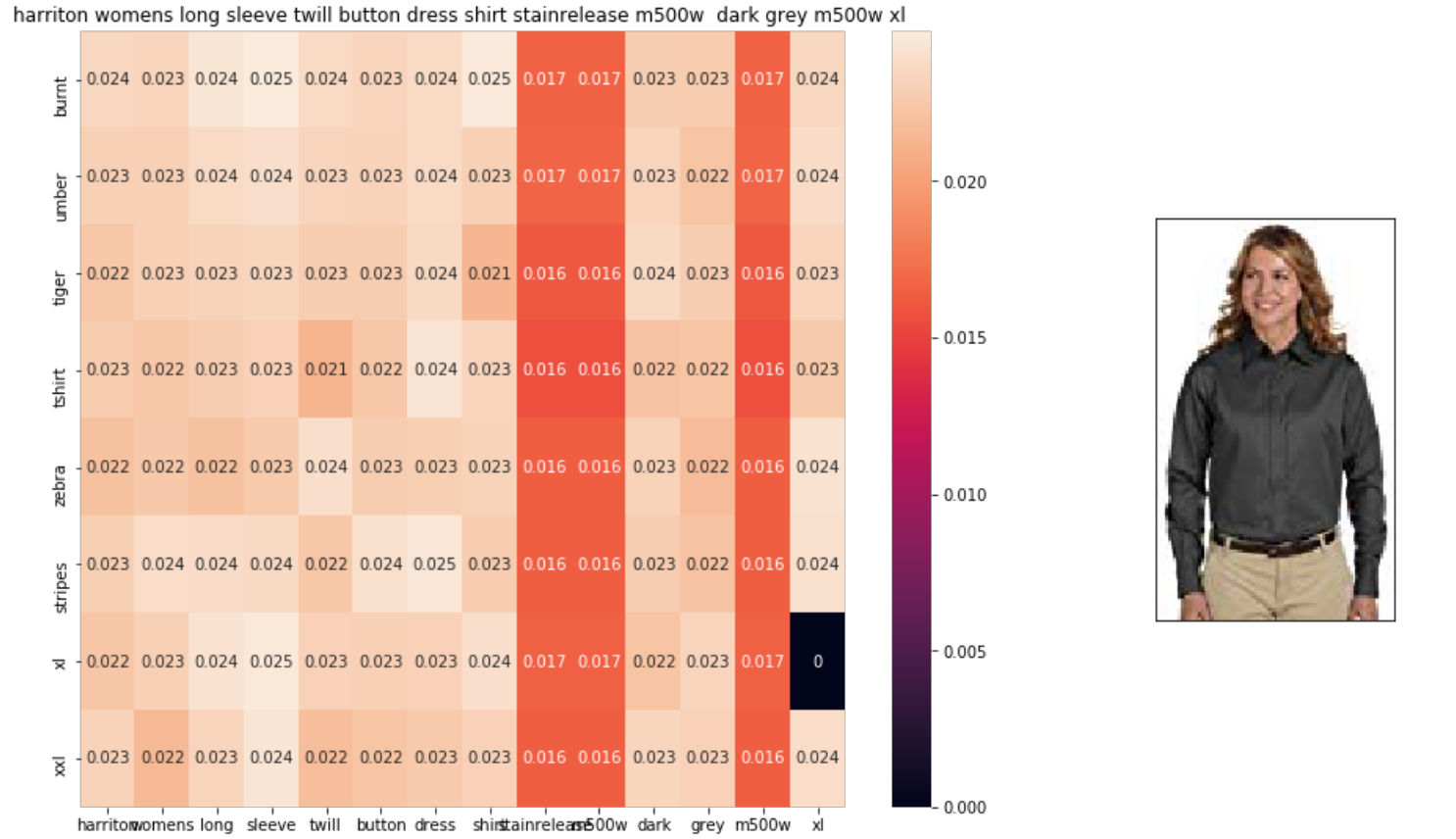
==========



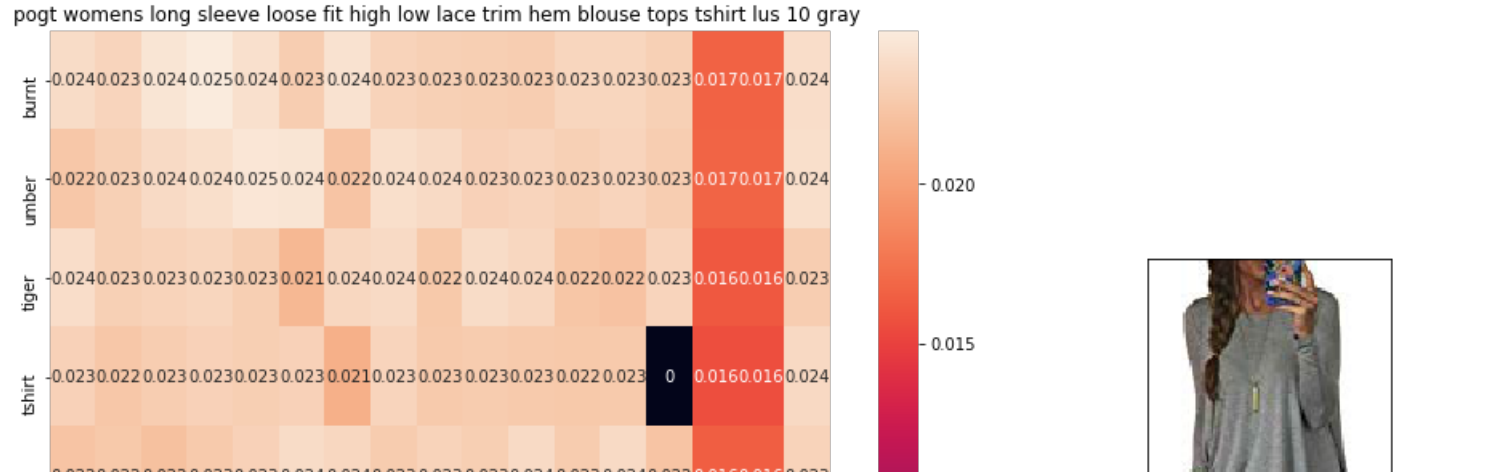


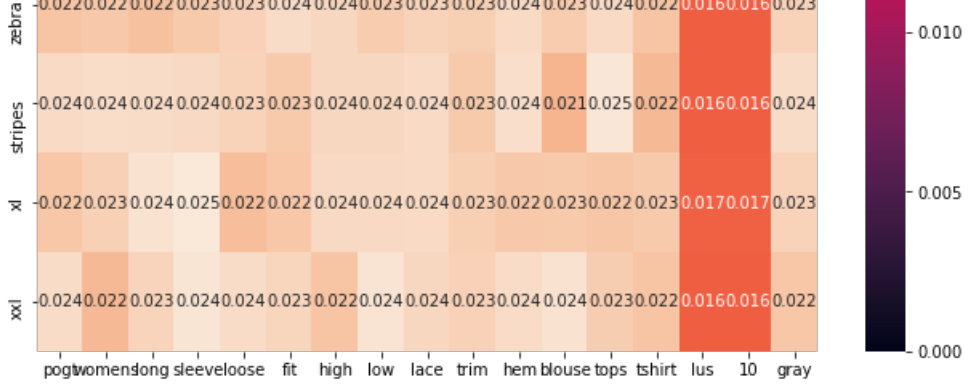


ASIN : B01N7V5ETV
BRAND : Lifestyle Batik
euclidean distance from given input image : 0.0057500987



ASIN : B008XNWR66
BRAND : Harriton
euclidean distance from given input image : 0.0057958313

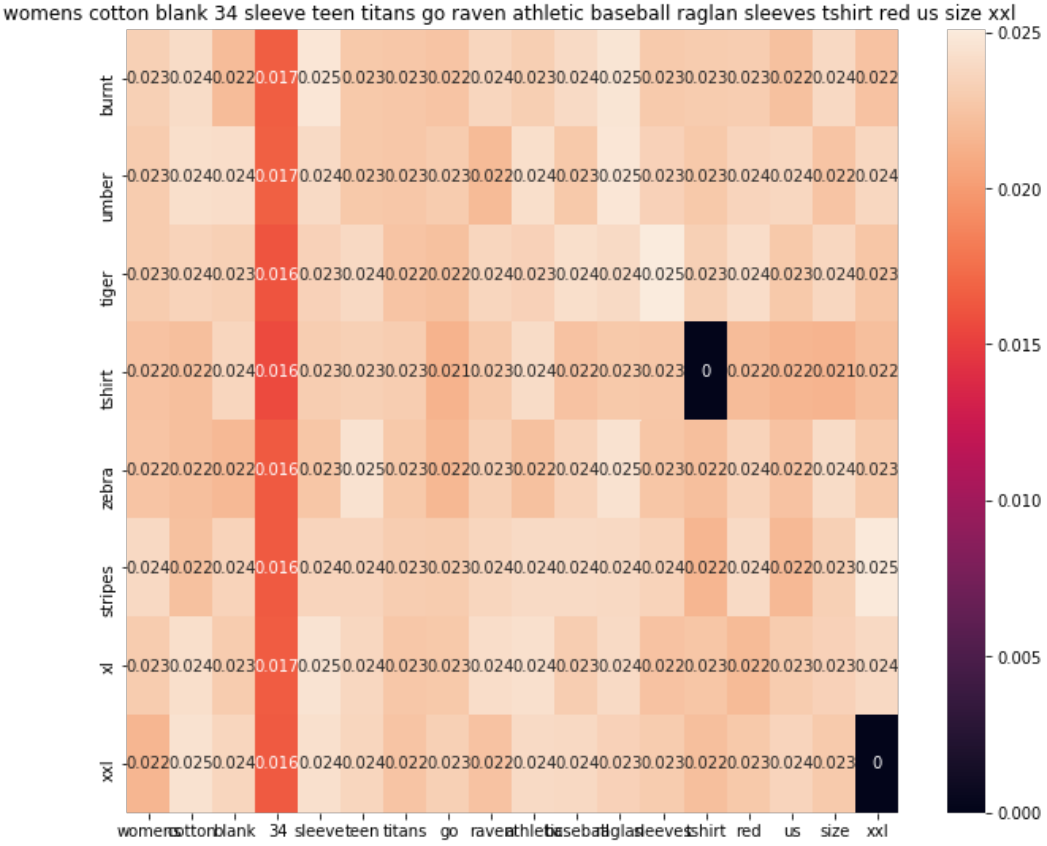




ASIN : B01M19NC0Y
BRAND : POGT
euclidean distance from given input image : 0.0058030924

==========

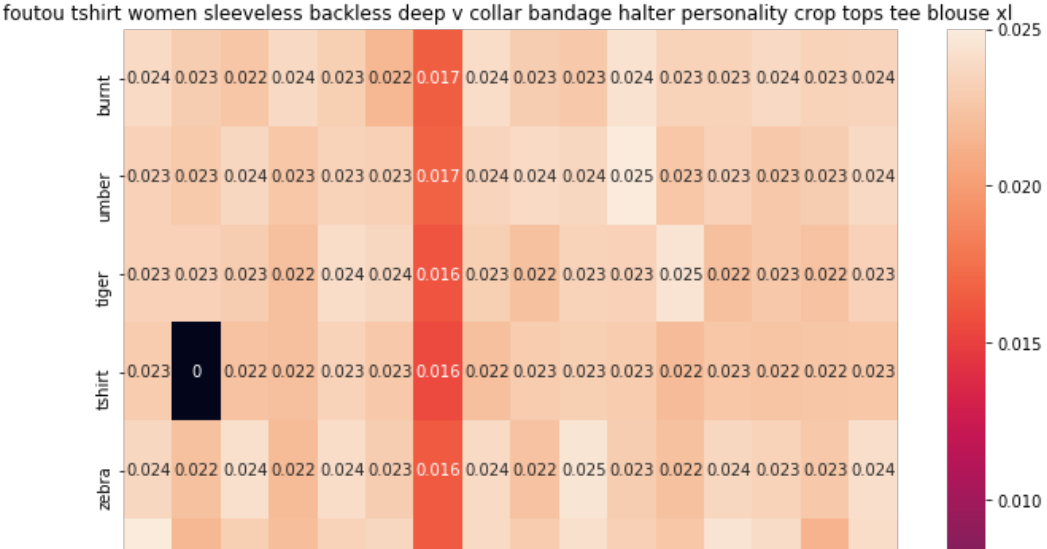
womens cotton blank 34 sleeve teen titans go raven athletic baseball raglan sleeves tshirt red us size xxl



ASIN : B01M4IJ7IR
BRAND : SDFJY
euclidean distance from given input image : 0.0058121355

==========

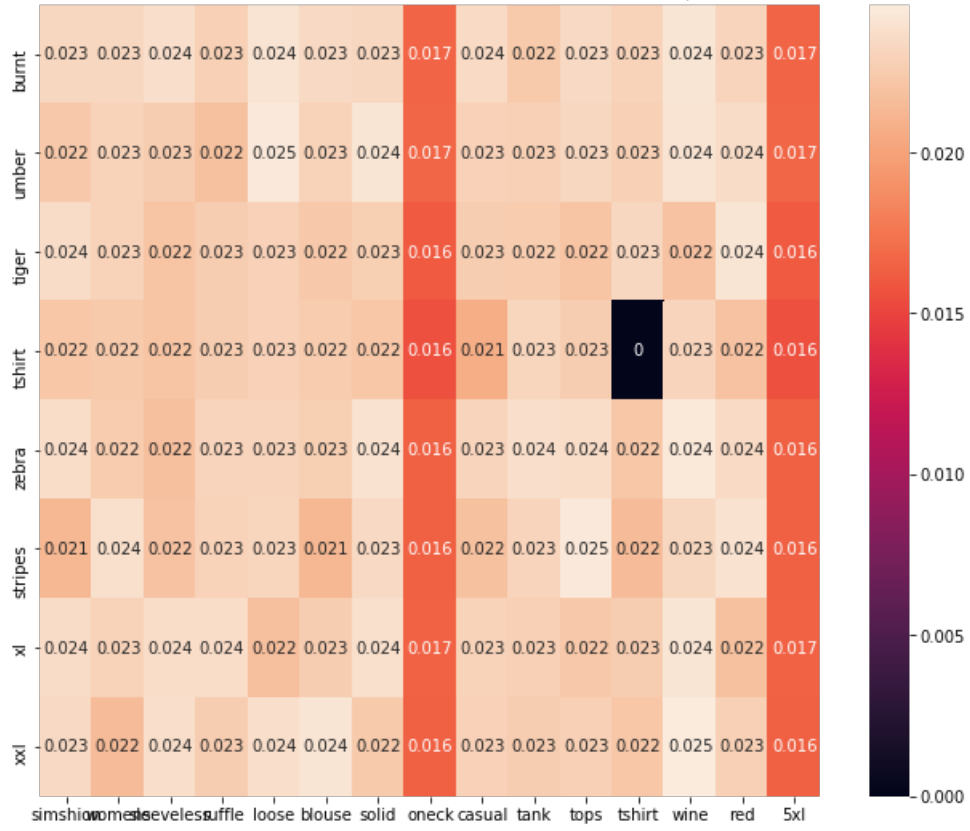
foutou tshirt women sleeveless backless deep v collar bandage halter personality crop tops tee blouse xl



ASIN : B06XXJKTGC
BRAND : Foutou
euclidean distance from given input image : 0.005823288

---

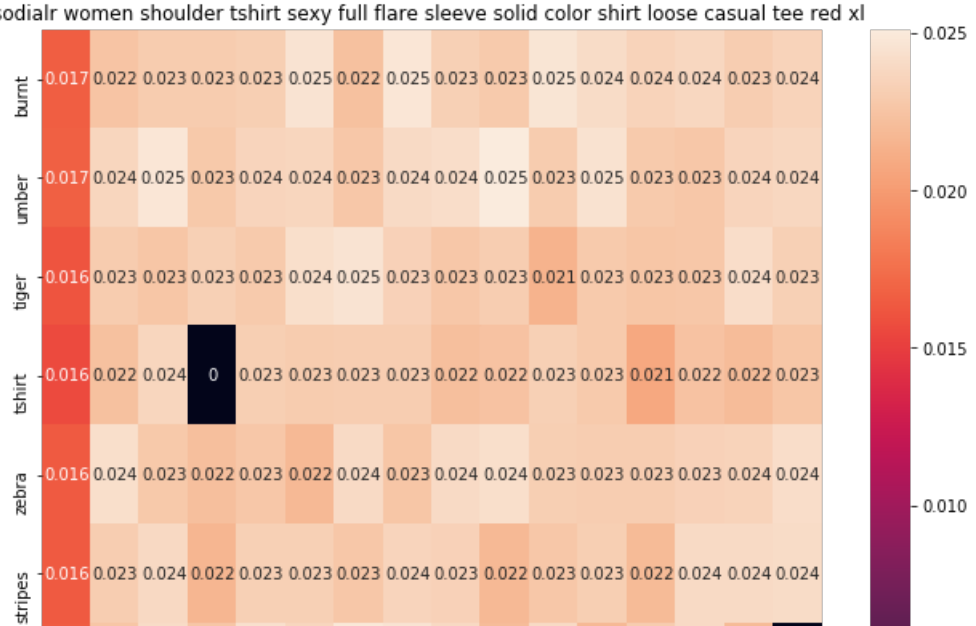
simshion womens sleeveless ruffle loose blouse solid oneck casual tank tops tshirt wine red 5xl

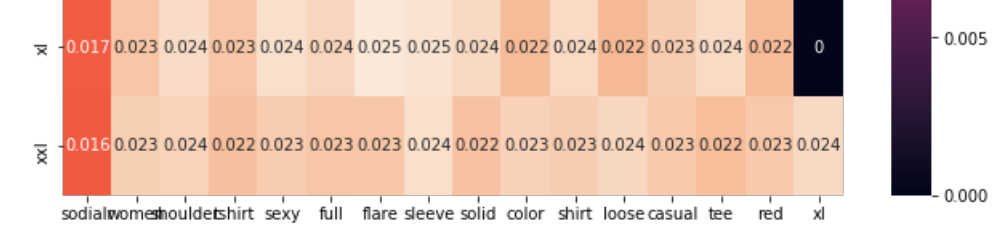


ASIN : B072QS773R
BRAND : SIMSHION
euclidean distance from given input image : 0.0058479505

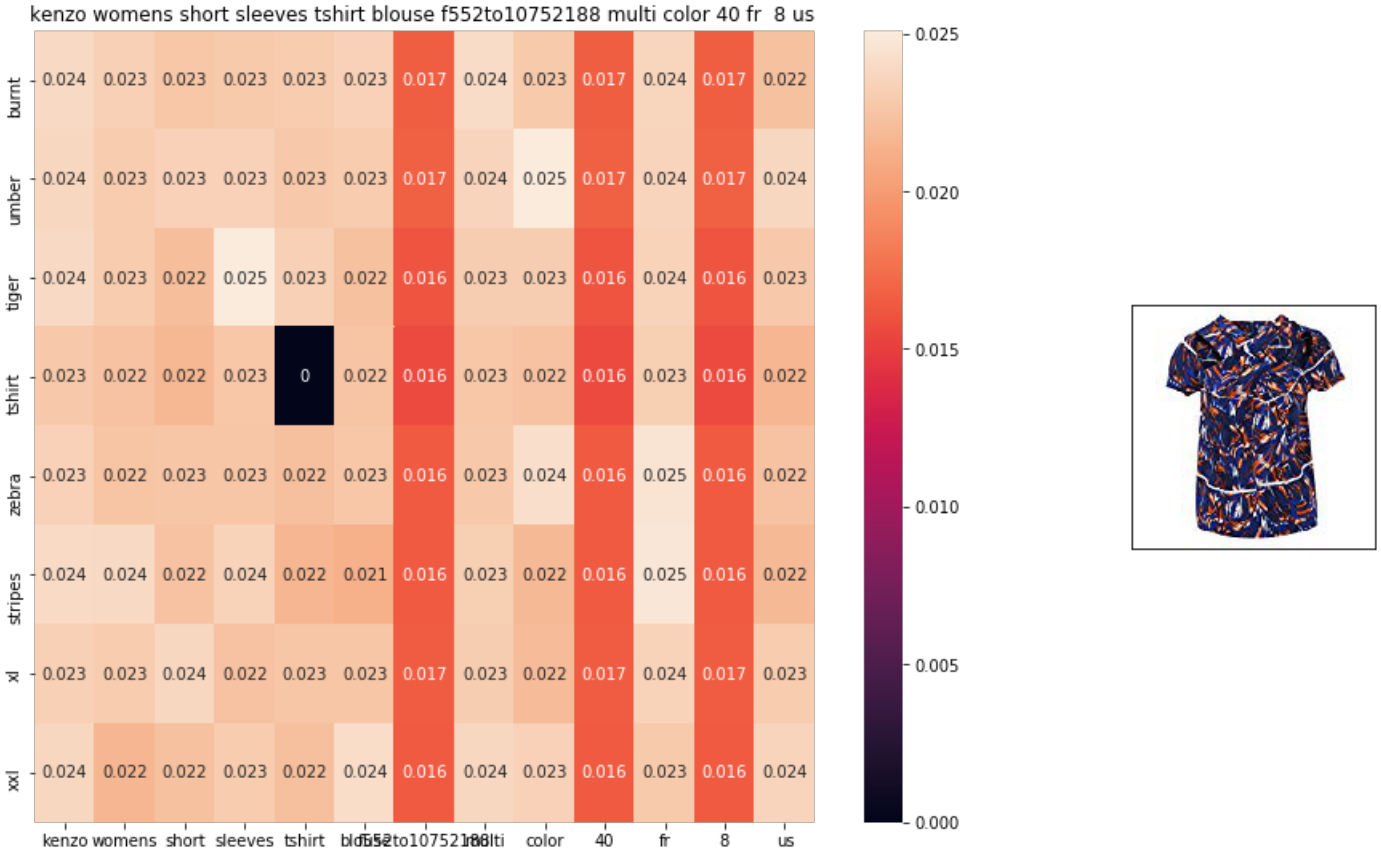
---

sodialr women shoulder tshirt sexy full flare sleeve solid color shirt loose casual tee red xl

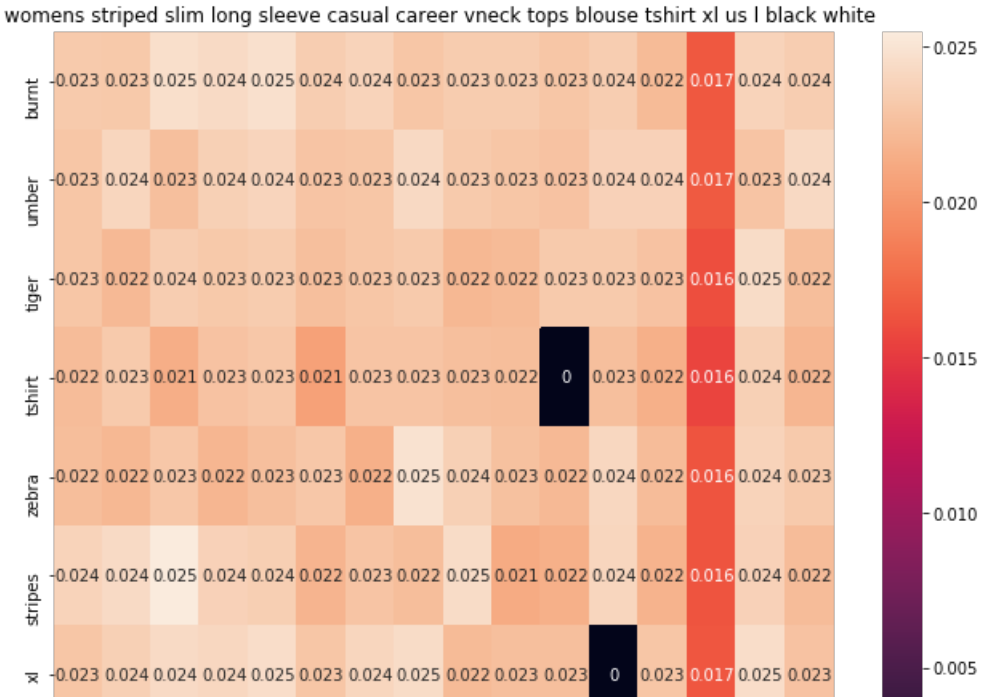


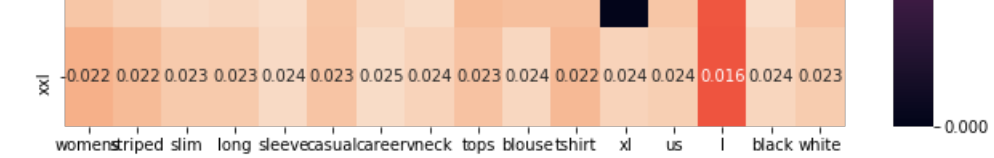


ASIN : B073WBF4D8
BRAND : SODIAL(R)
euclidean distance from given input image : 0.0058487765

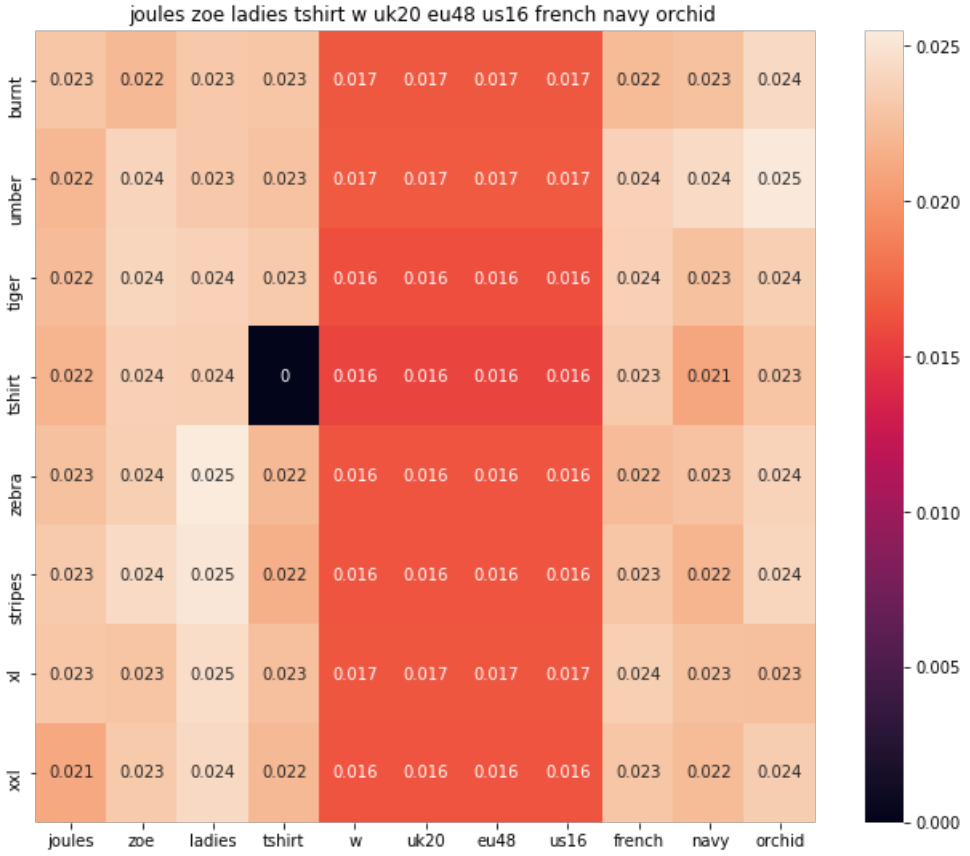


ASIN : B06W9JR55J
BRAND : Kenzo
euclidean distance from given input image : 0.005857495

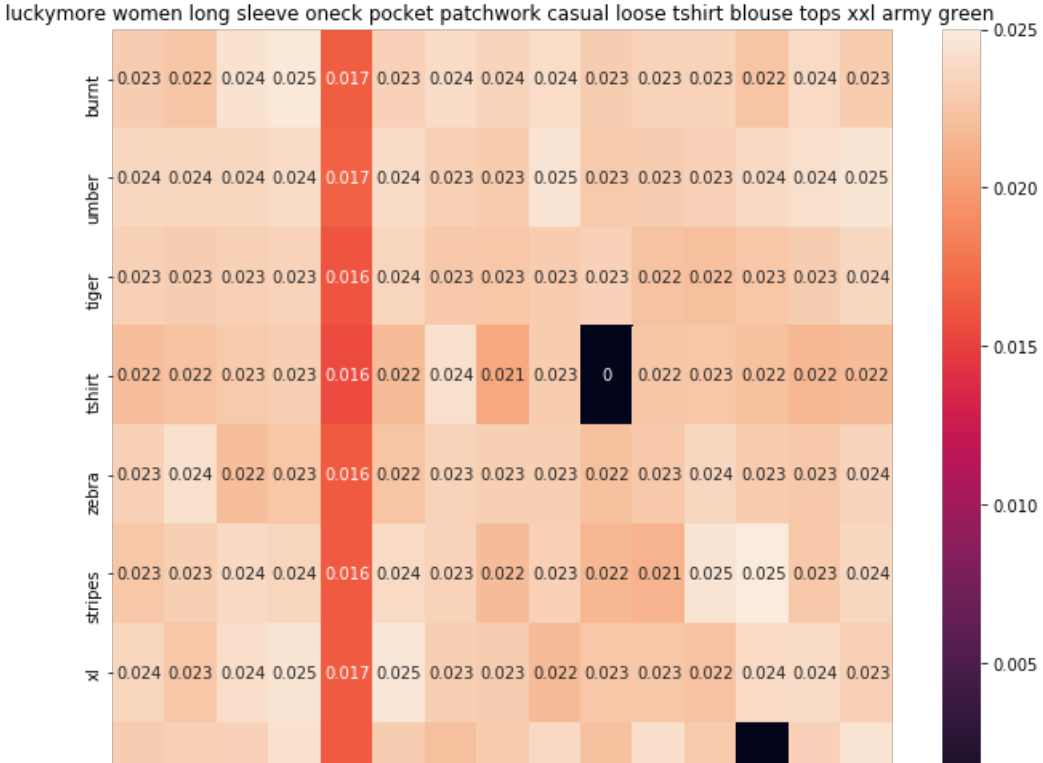


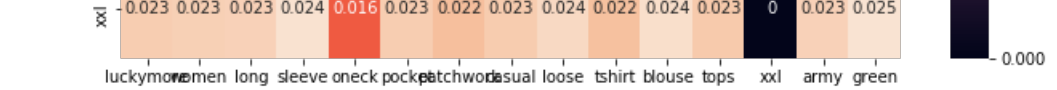


ASIN : B00MTHH2PS
BRAND : ACEFAST INC
euclidean distance from given input image : 0.005858263

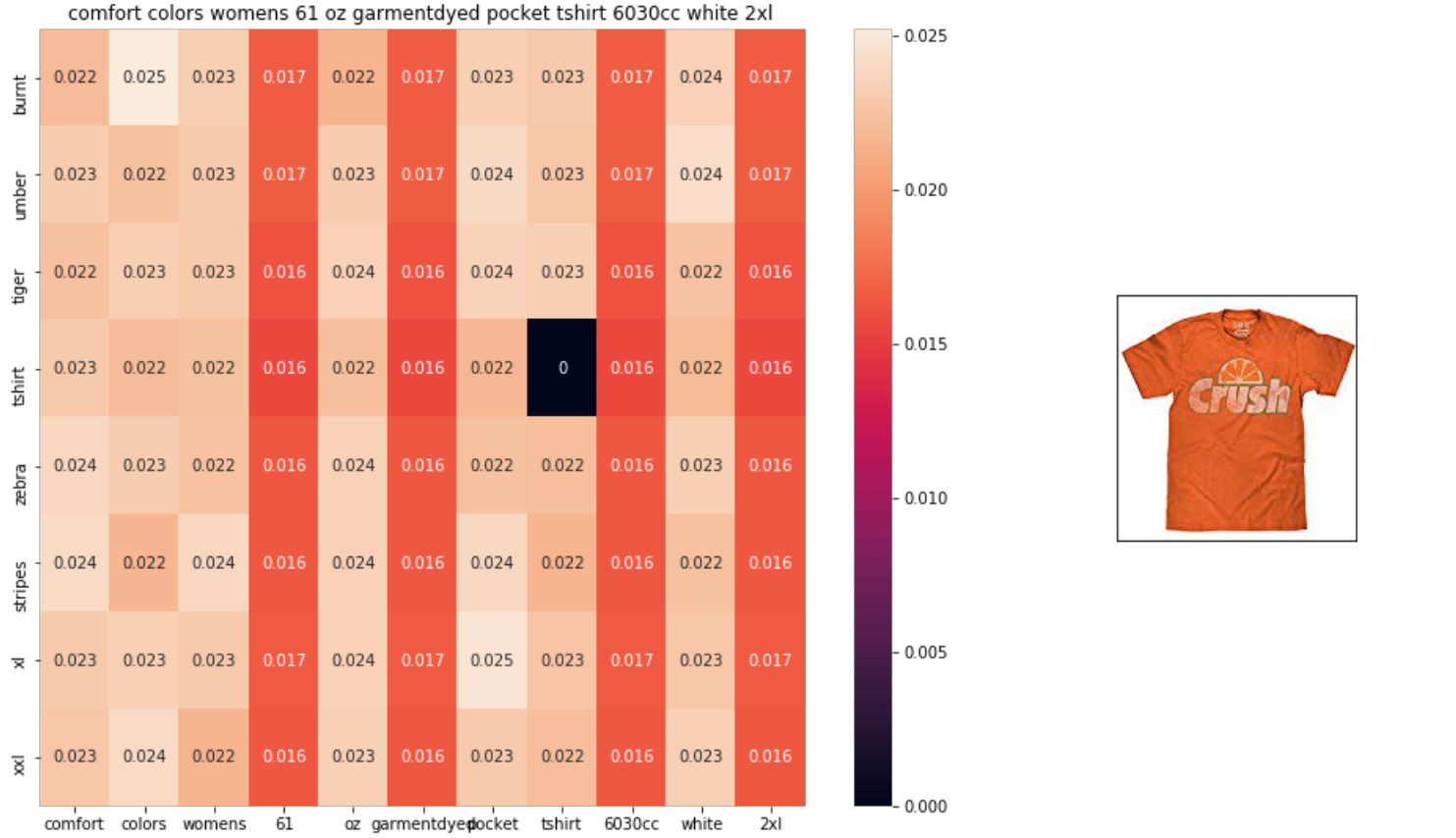


ASIN : B01N4X62S2
BRAND : Joules
euclidean distance from given input image : 0.005865807

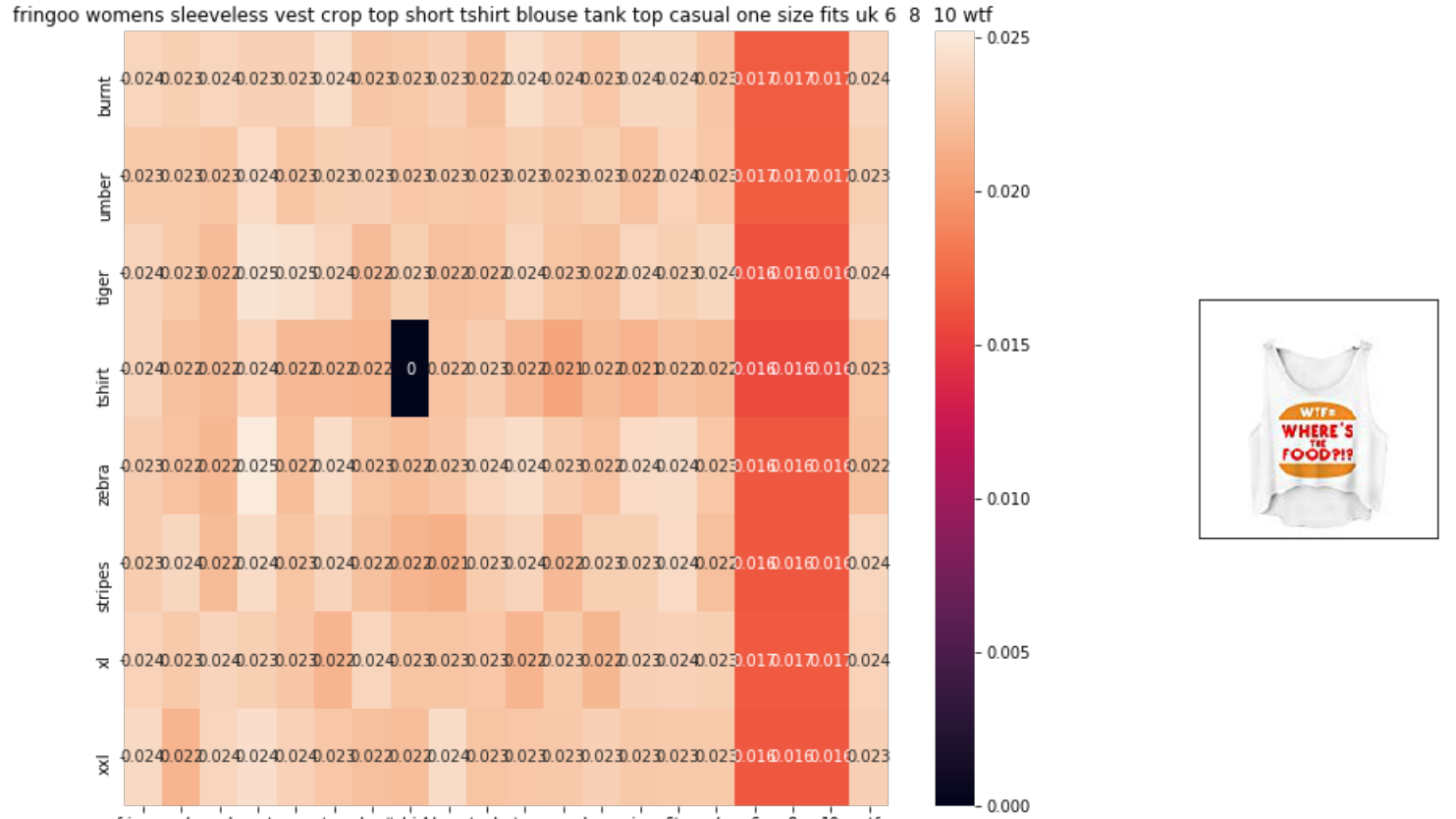




ASIN : B01L6EUUJE
BRAND : Luckymore
euclidean distance from given input image : 0.0058750007
==========

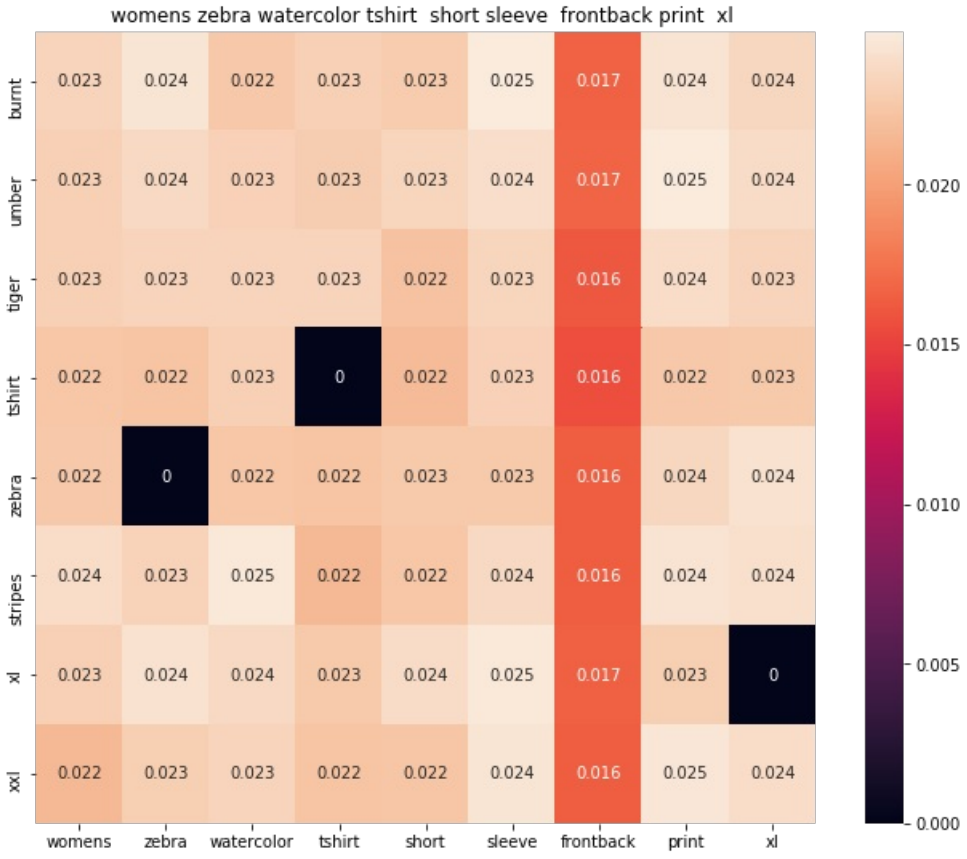


ASIN : B014WBVRNG
BRAND : Comfort Colors
euclidean distance from given input image : 0.0058898837
==========

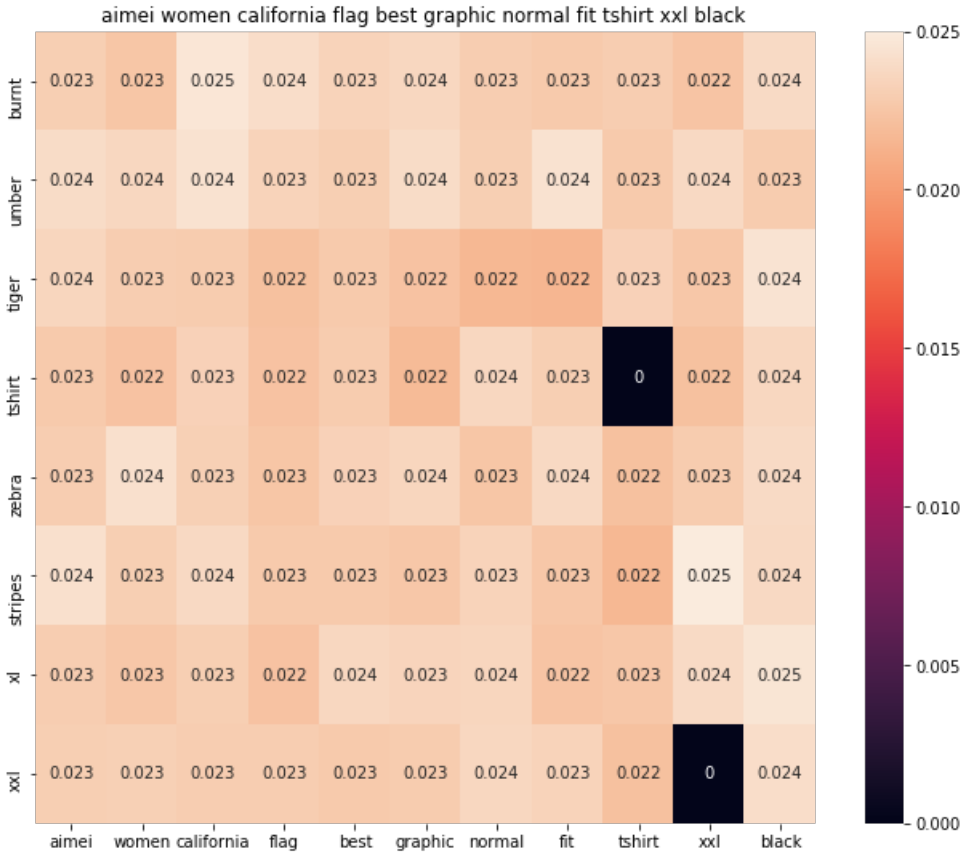


fringoonsleeveless vest crop top shorttshirtblousetank topcasualone size fits uk 6 8 10 wtf

ASIN : B010W7TQ6O
BRAND : Fringoo
euclidean distance from given input image : 0.0058909995
==========



ASIN : B072R2JXKW
BRAND : WHAT ON EARTH
euclidean distance from given input image : 0.00591293
==========



ASIN : B012KY4OQ4
BRAND : ZHENGAIMEI
euclidean distance from given input image : 0.0059157484
==========

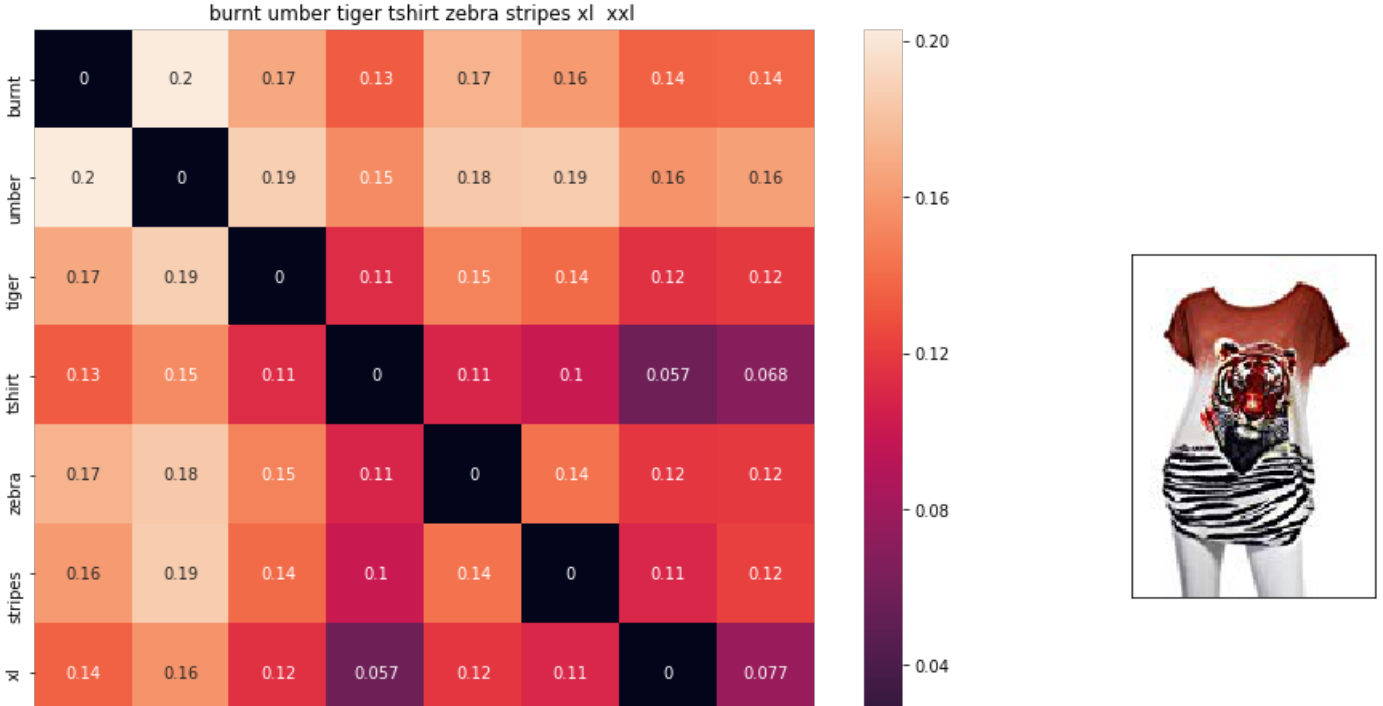
IDF weighted Word2Vec for product similarity

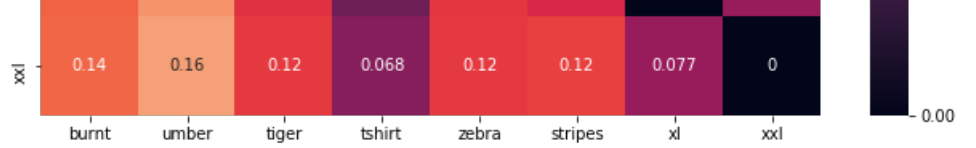
In [33]:

```
w2v_title_weight = []

# for every title we build a weighted vector representation
doc_id = 0
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)
```

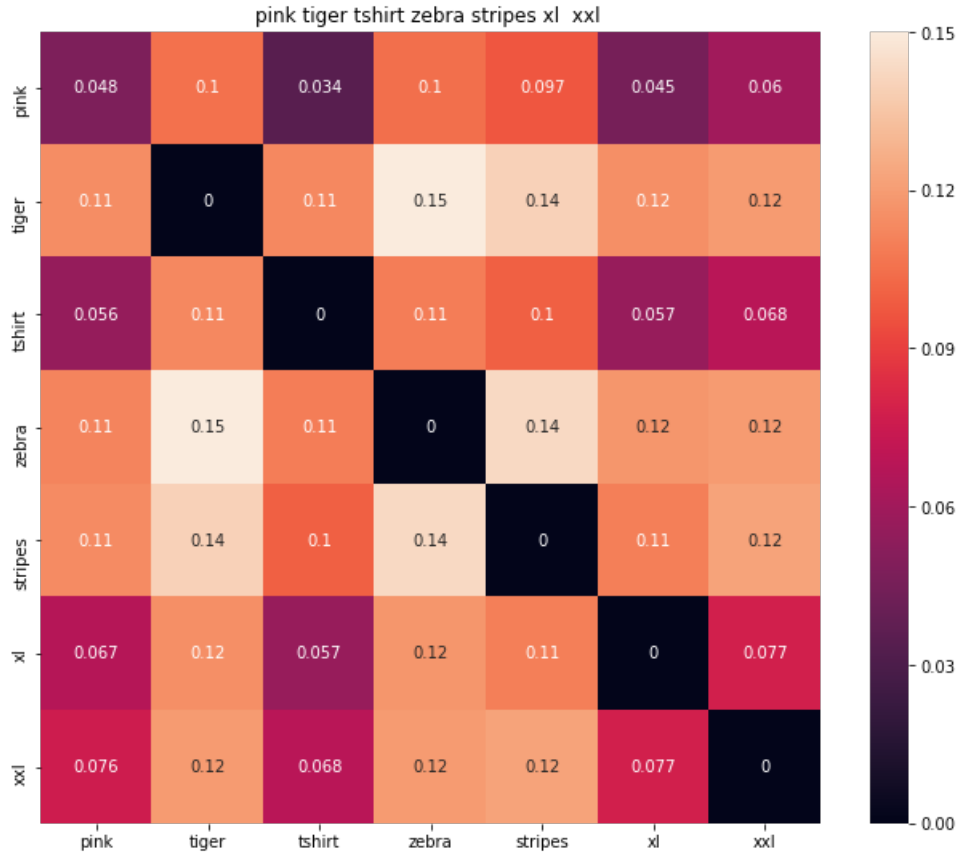
In [34]:

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

    # pairwise_dist will store the distance from given input apparel to all remaining 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(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[i]],data['title'].loc[df_indices[i]],
                 data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i], 'weighted')
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print('Brand :',data['brand'].loc[df_indices[i]])
        print('euclidean distance from input :', pdists[i])
        print('='*125)

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

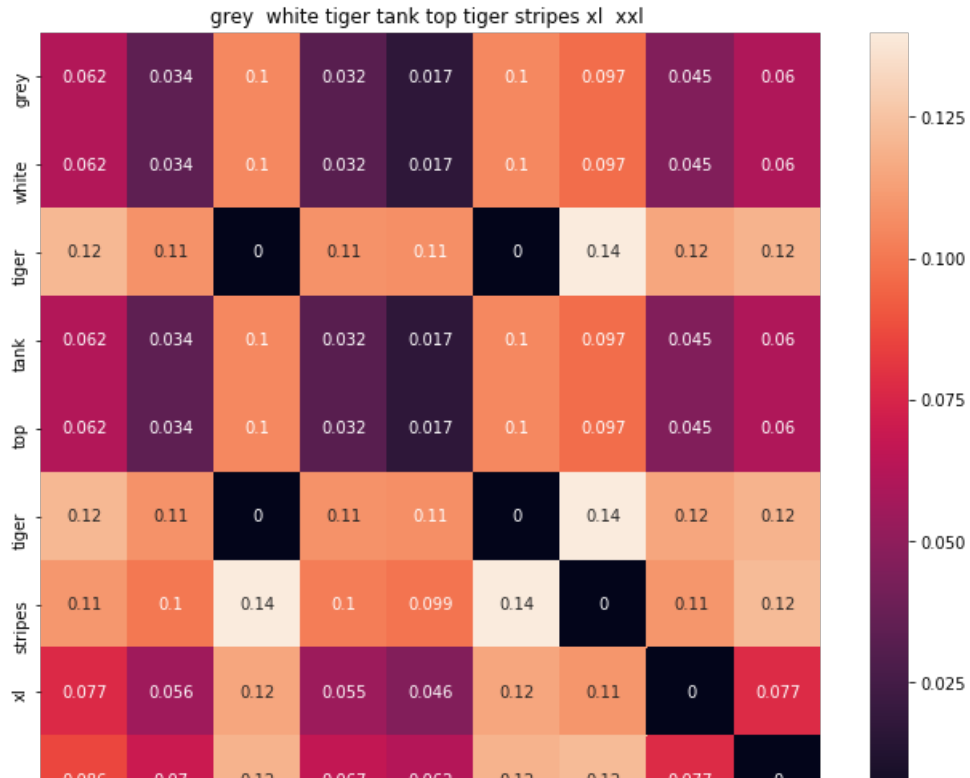




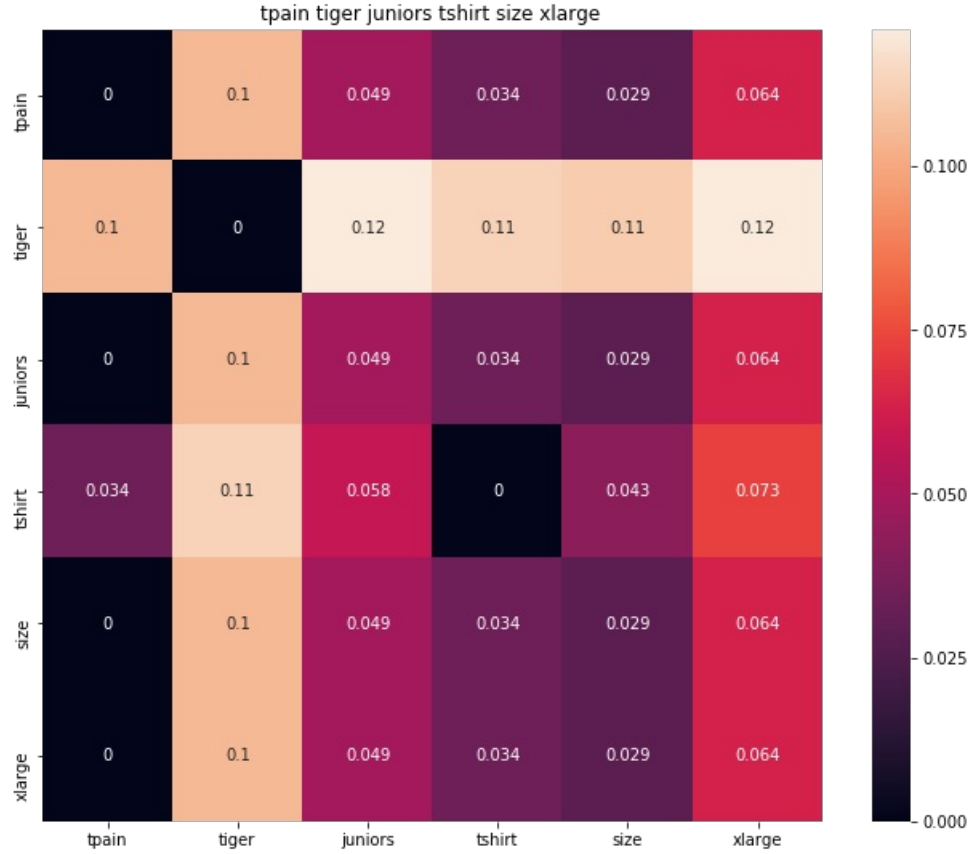
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.0



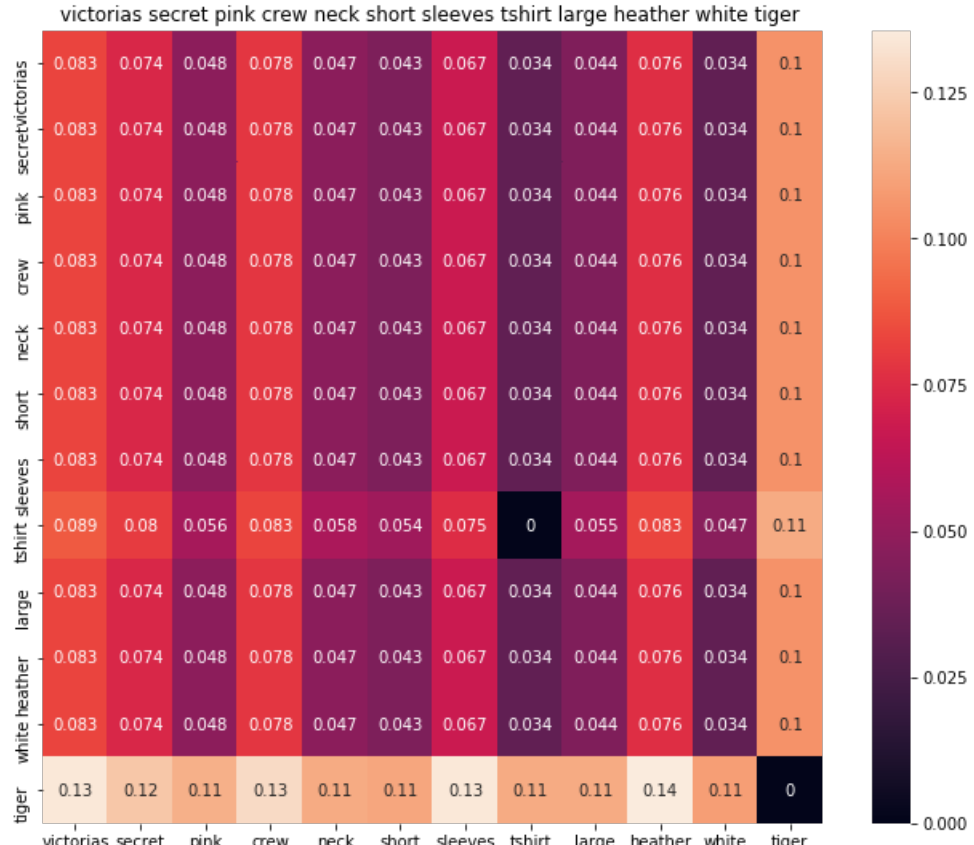
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 0.025277356



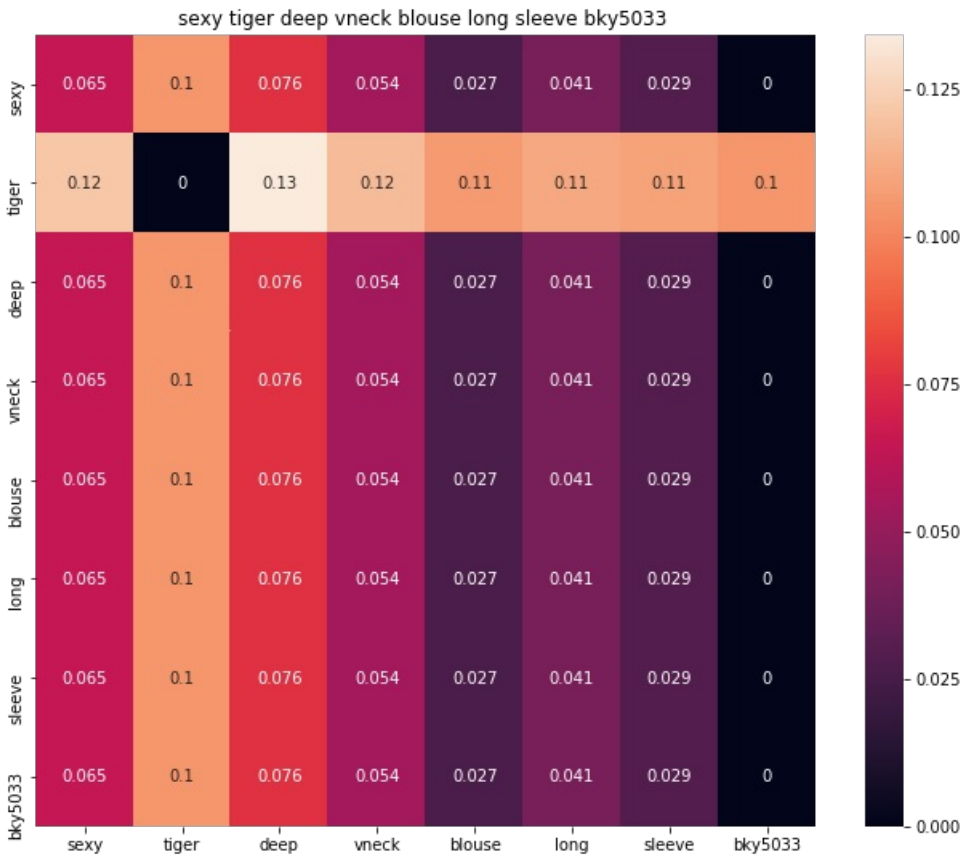
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 0.029932518



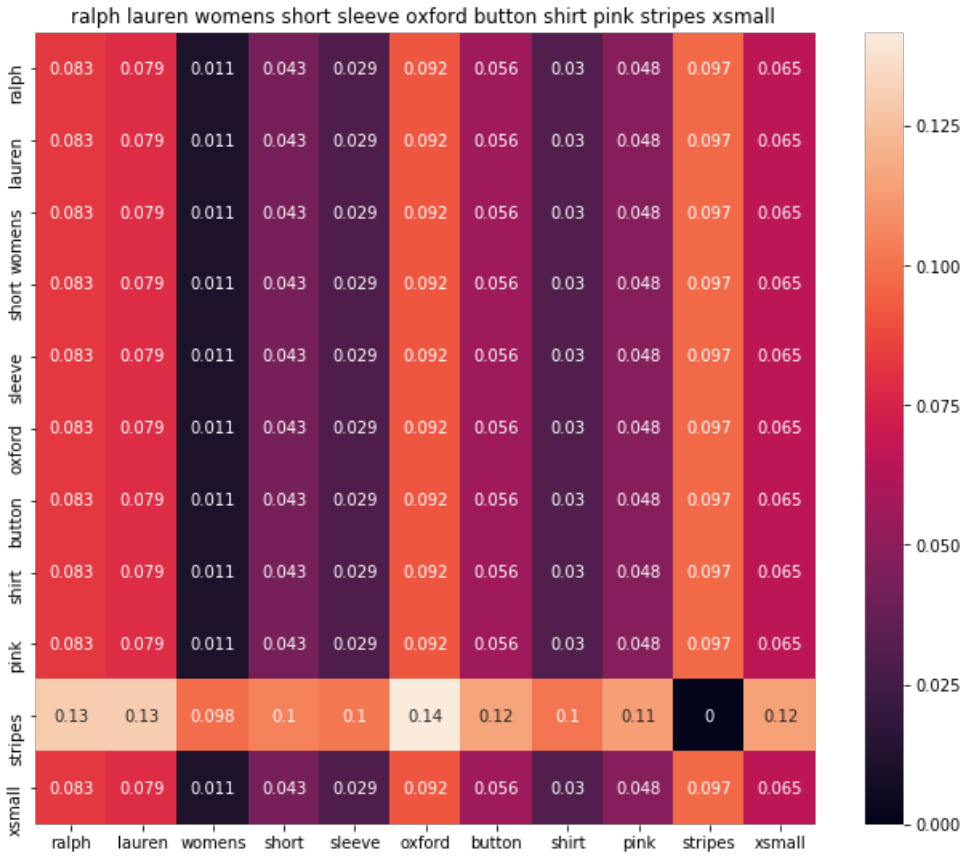
ASIN : B01K0H02OG
Brand : Tultex
euclidean distance from input : 0.030965429



ASIN : B0716MVPGV
Brand : V.Secret
euclidean distance from input : 0.032354787
==========

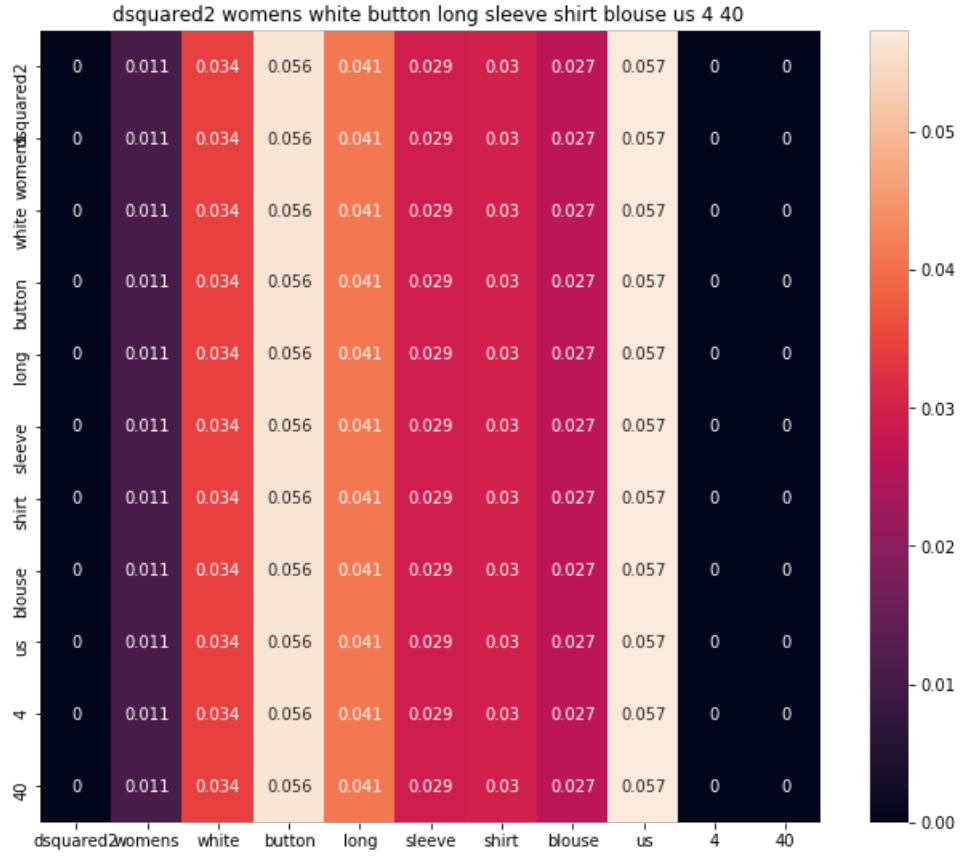


ASIN : B00R09MFAC
Brand : bankhunyabangyai store
euclidean distance from input : 0.03259634
==========



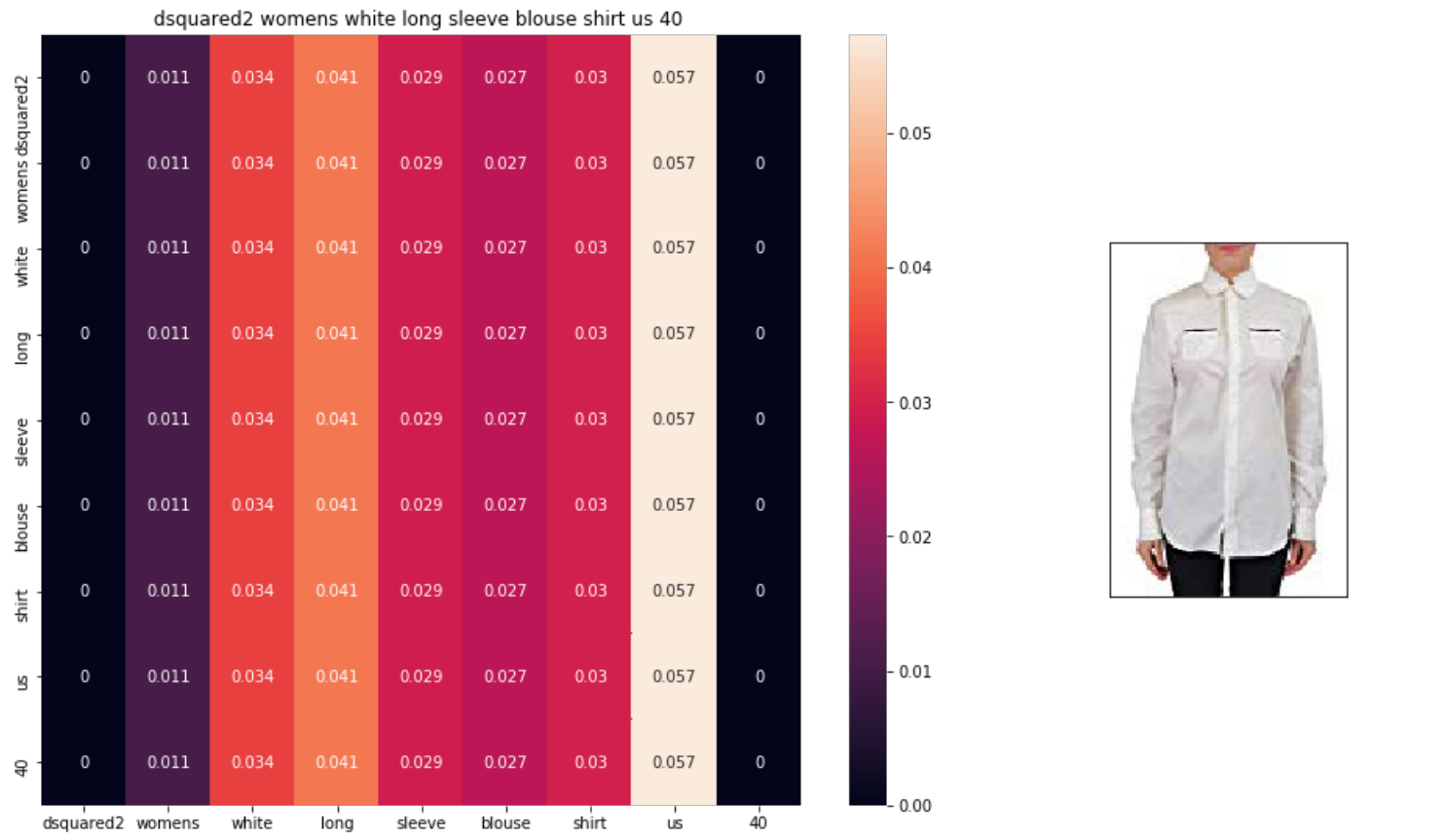
ASIN : B01ZXAGPU8
Brand : RALPH LAUREN
euclidean distance from input : 0.032625005

ASIN : B00Z6HEXWI
Brand : Black Temptation
euclidean distance from input : 0.032639246



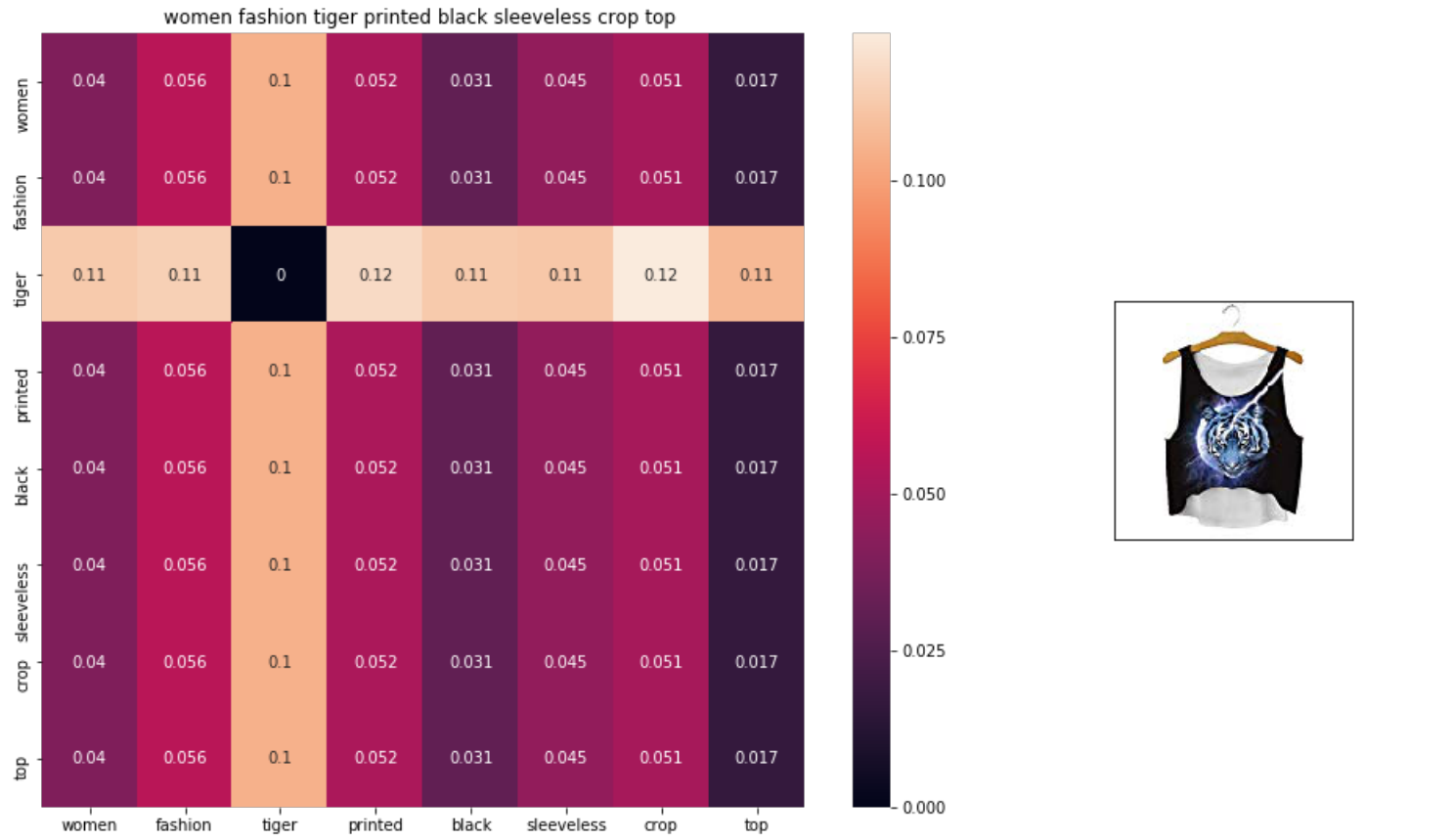
ASIN : B00WTEL180
Brand : DSQUARED2
euclidean distance from input : 0.032654457

==========



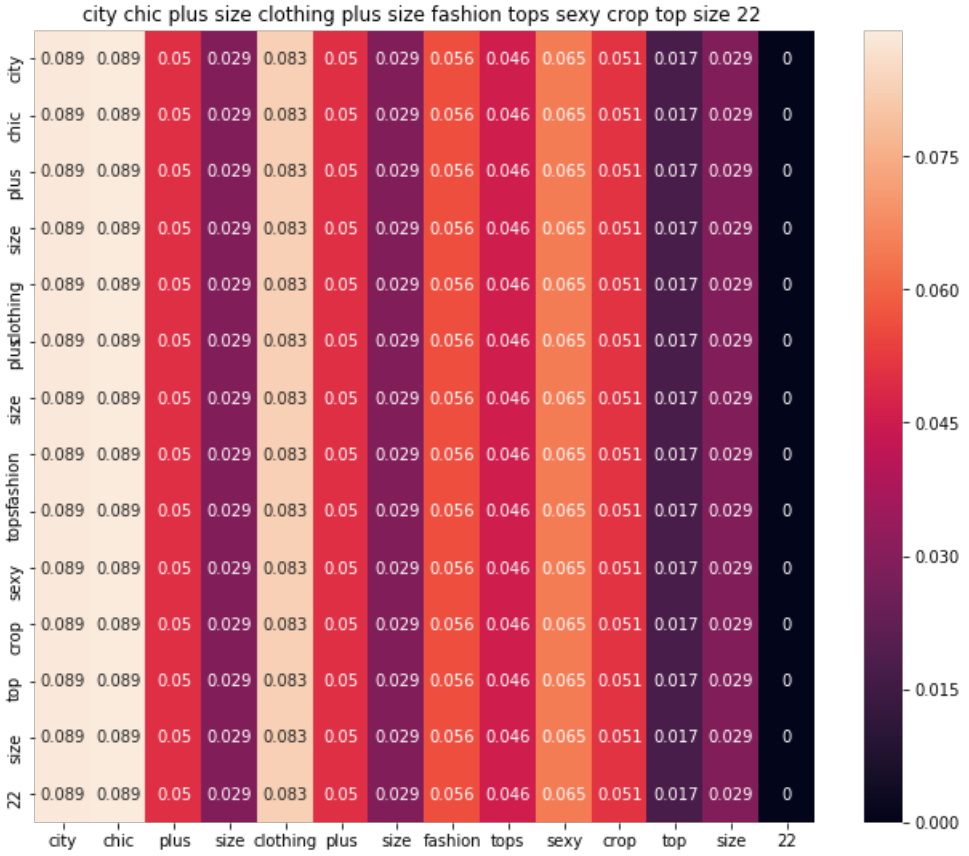
ASIN : B01758P216
Brand : DSQUARED2
euclidean distance from input : 0.032674517

==========



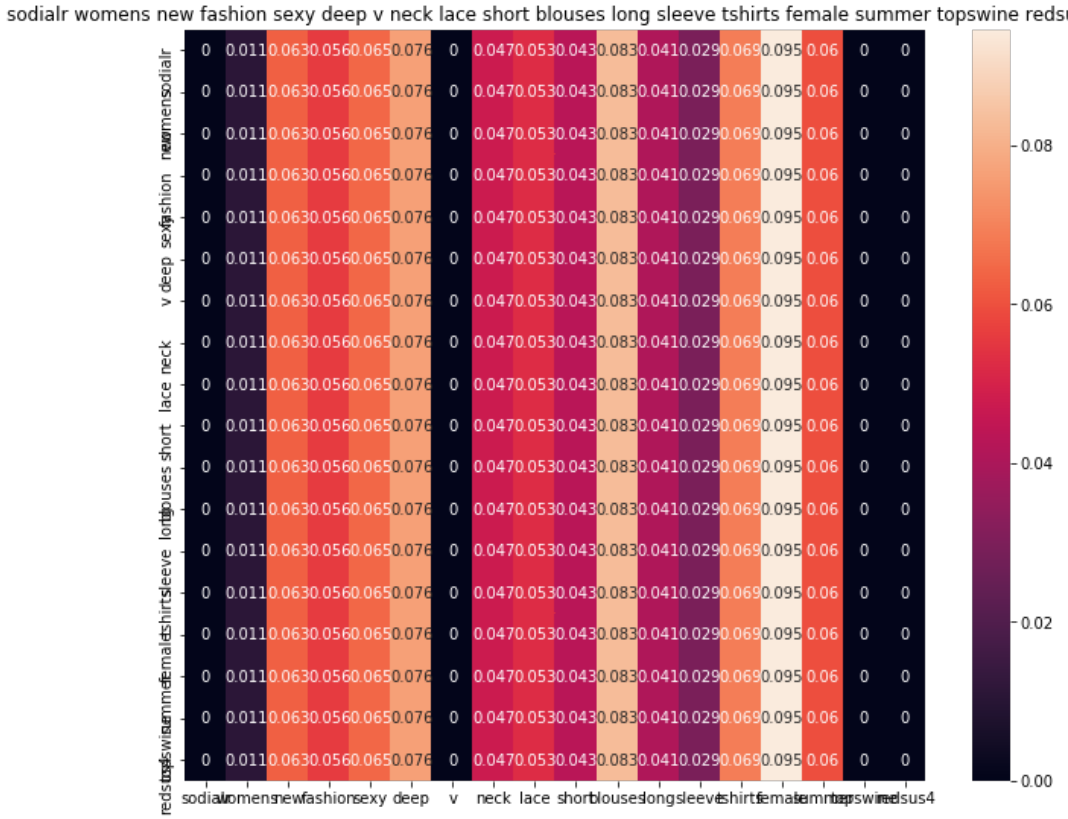
ASIN : B074T8ZYGX
Brand : MKP Crop Top
euclidean distance from input : 0.03269034

==========



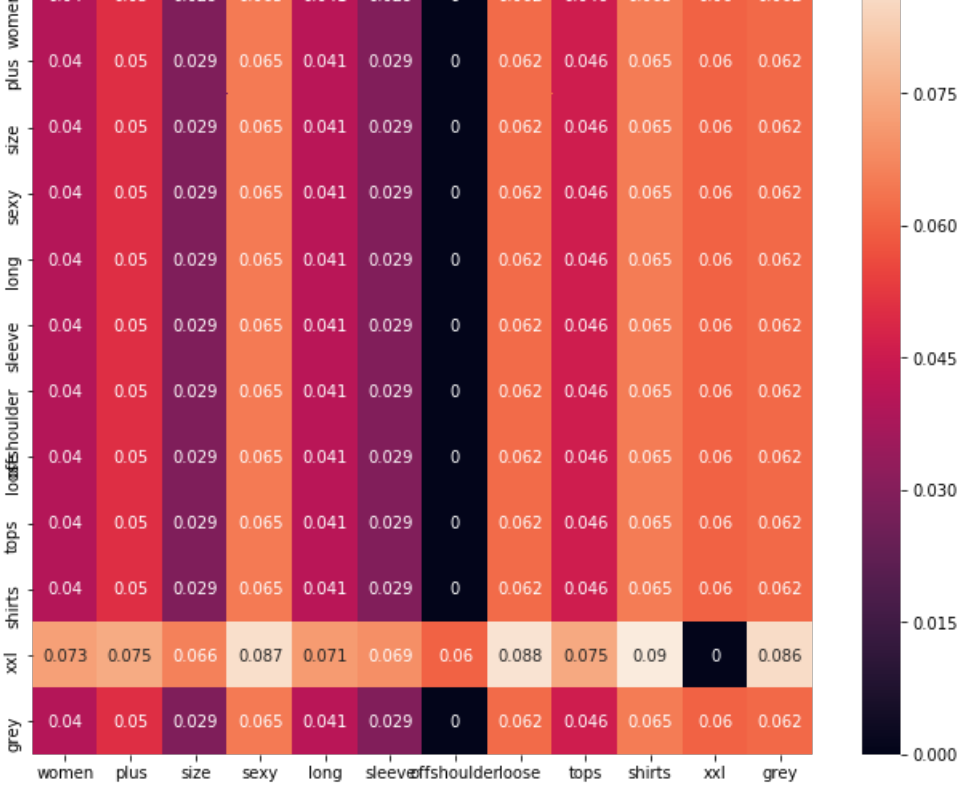
ASIN : B01HJNIBEC
Brand : City Chic
euclidean distance from input : 0.032721933

==========

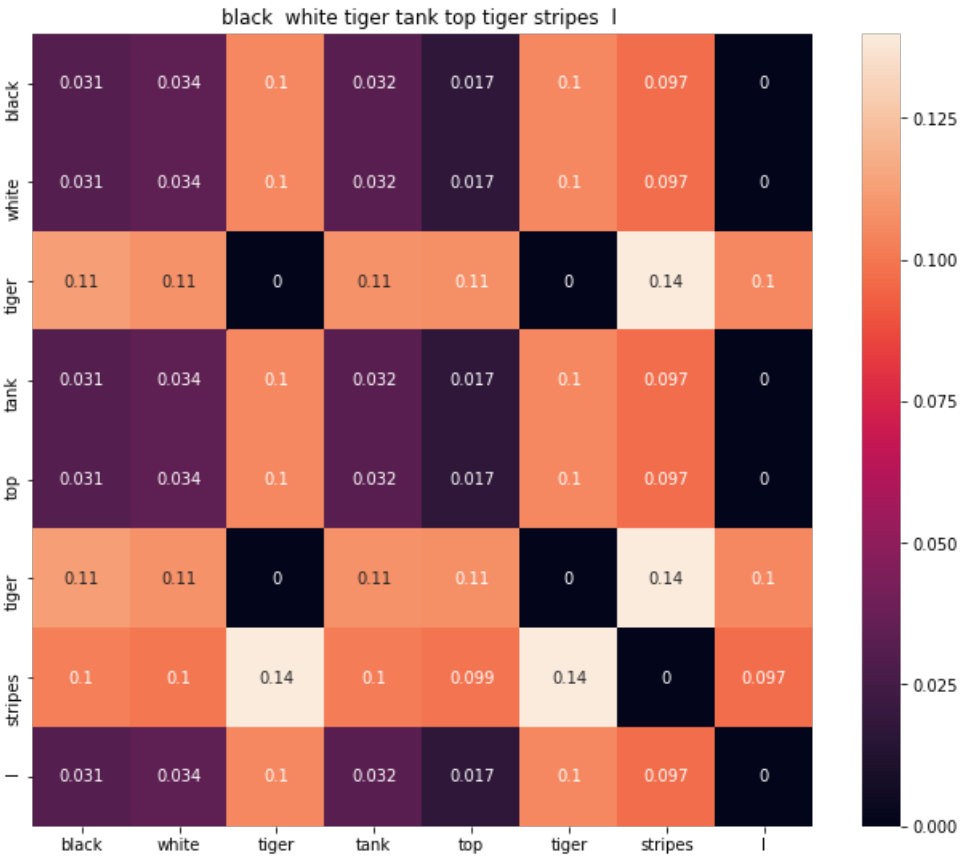


ASIN : B074FT1X1M
Brand : SODIAL(R)
euclidean distance from input : 0.03273076

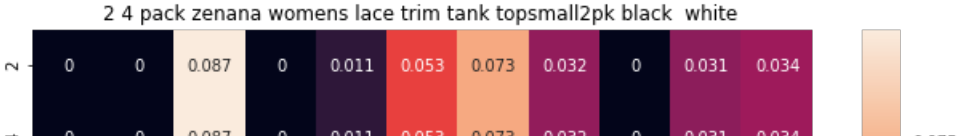
==========

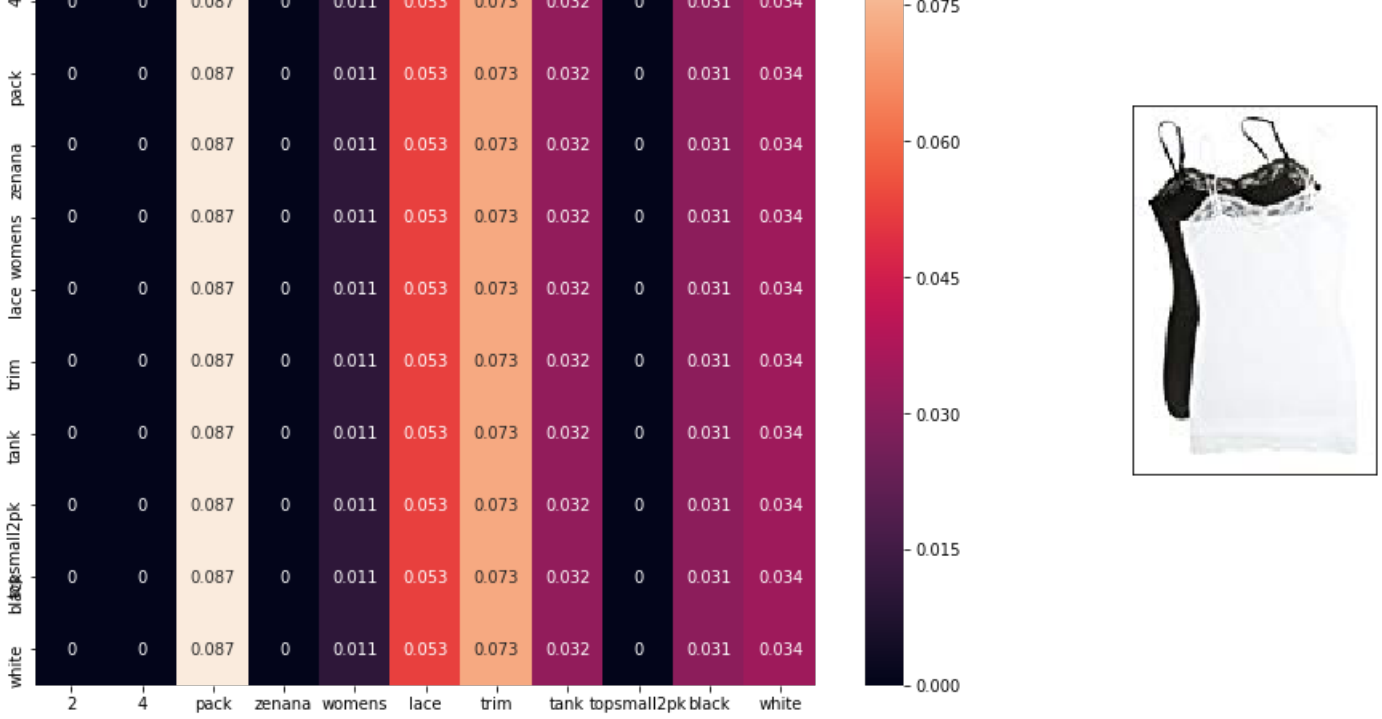


ASIN : B01EQUG5AS
Brand : Cutecc
euclidean distance from input : 0.03279542



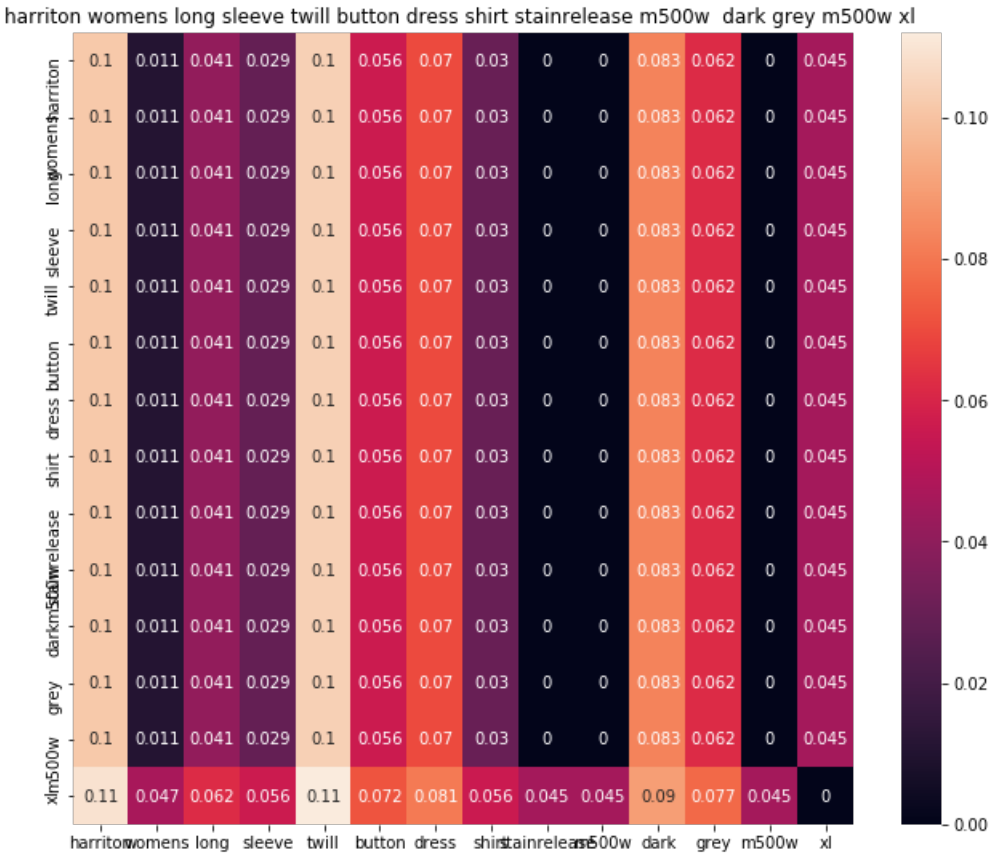
ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 0.03281579





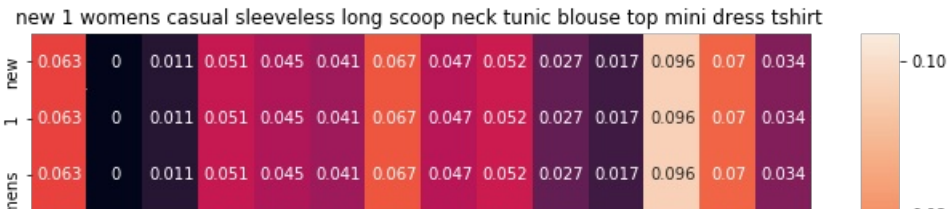
ASIN : B00DHKRSOG
Brand : Zenana Outfitters
euclidean distance from input : 0.032818317

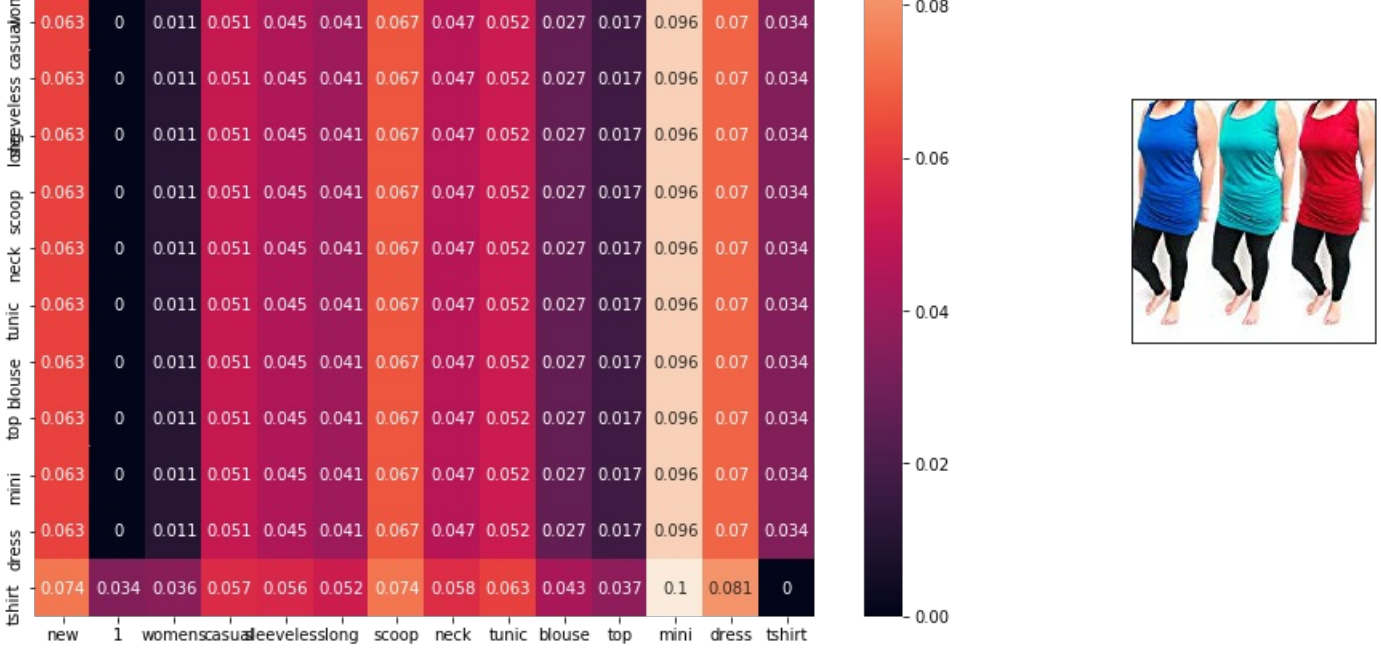
==========



ASIN : B008XNWR66
Brand : Harriton
euclidean distance from input : 0.03287094

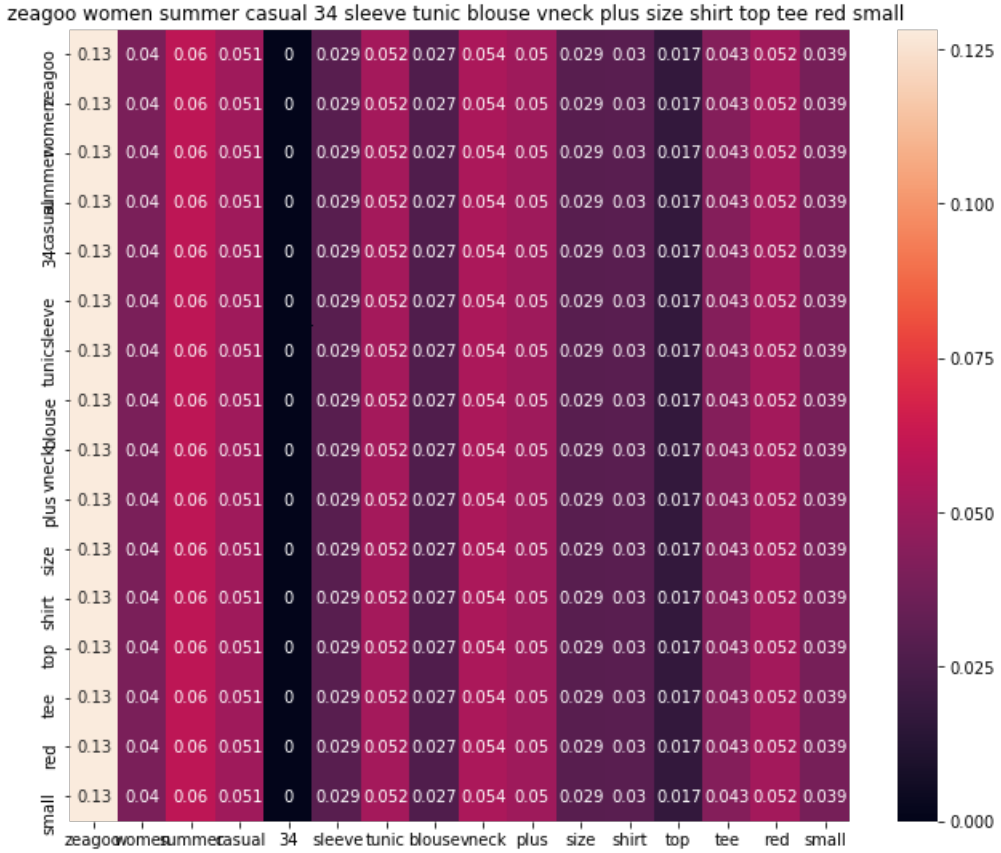
==========





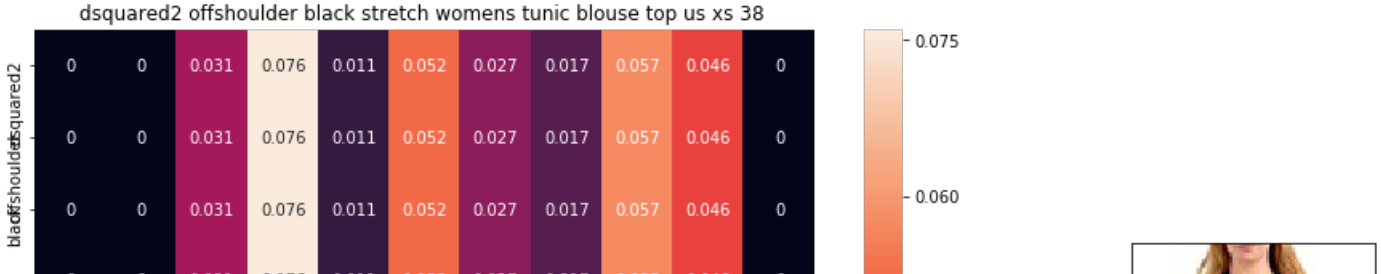
ASIN : B073CXSNM6
Brand : General
euclidean distance from input : 0.0328726
==========

zeagoo women summer casual 34 sleeve tunic blouse vneck plus size shirt top tee red small



ASIN : B0725RCTLT
Brand : Zeagoo
euclidean distance from input : 0.0328831
==========

dsquared2 offshoulder black stretch womens tunic blouse top us xs 38



ASIN : B01NA88EHJ
Brand : DSQUARED2
euclidean distance from input : 0.03290037
==========

Weighted similarity using brand and color

In [35]:

```
data['brand'][:5]
```

Out[35]:

```
4                 FeatherLite
6     HX-Kingdom Fashion T-shirts
15                FeatherLite
27                    FNC7C
46                Fifth Degree
Name: brand, dtype: object
```

In [36]:

```
data['color'][:5]
```

Out[36]:

```
4     Onyx Black/ Stone
6                 White
15                White
27               Purple
46                Black
Name: color, dtype: object
```

In [37]:

```
data.head(2)
```

Out[37]:

	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

In [38]:

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

```
# replace spaces with hypen
brands = [x.replace(" ", "-") for x in data['brand'].values]
types = [x.replace(" ", "-") for x in data['product_type_name'].values]
colors = [x.replace(" ", "-") for x in data['color'].values]

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

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

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

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

In [39]:

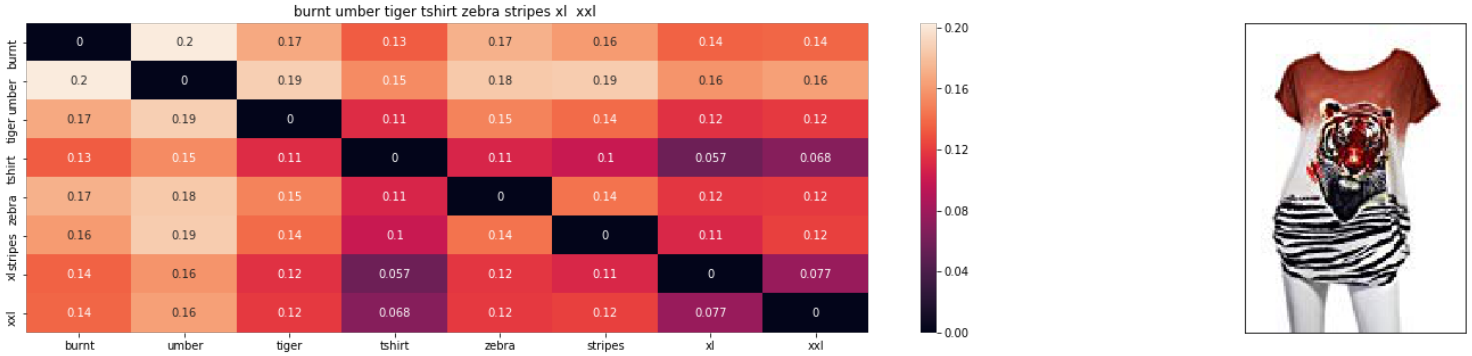
```
def heat_map_w2v_brand(sentence1, sentence2, url, doc_id1, doc_id2, index1, index2, model):

    # 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
    # index1: index of document1 in the data frame
    # index2: 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(sentence1, 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(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)

    data_matrix = [['Asin','Brand', 'Color', 'Product type'],
            [data['asin'].loc[index1], brands[doc_id1], colors[doc_id1], types[doc_id1]], # input apparel's features
            [data['asin'].loc[index2], brands[doc_id2], colors[doc_id2], types[doc_id2]]] # recommonded apparel's features

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

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

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

    # in first 25*10 grids we plot heatmap
    ax1 = plt.subplot(gs[:, :-5])
    # ploting the heap map based on the pairwise distances
    ax1 = sns.heatmap(np.round(s1_s2_dist,6), annot=True)
    # set the x axis labels as recommended apparels title
    ax1.set_xticklabels(sentence2.split())
    # set the y axis labels as input apparels title
    ax1.set_yticklabels(sentence1.split())
    # set title as recommended apparels title
    ax1.set_title(sentence2)

    # 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 [40]:

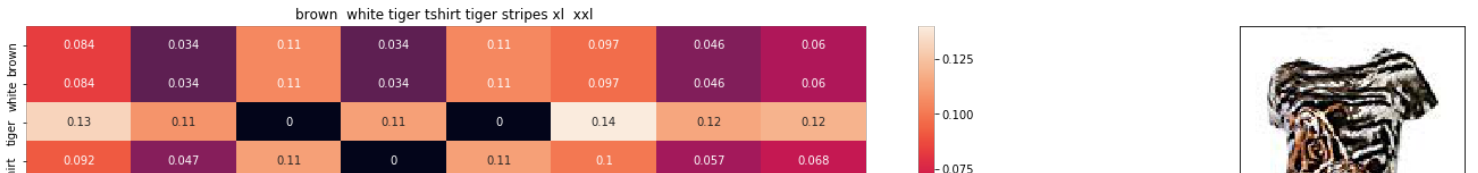
```
def idf_w2v_brand(doc_id, w1, w2, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand and color features

    # 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) = <X, Y> / (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))
    ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id]) # 3 items [brand + type(shirt/top) + color]
    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[i]],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)
```

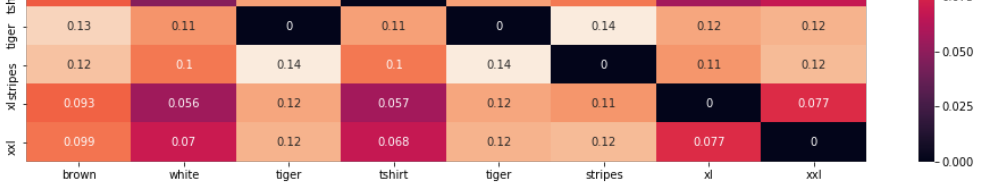
In [41]:

```
# same imp to 'idf' and '[brand + type(shirt/top) + color]'
idf_w2v_brand(12566, 5, 5, 20)
# in the give heat map, each cell contains the euclidean distance between words i, j
```



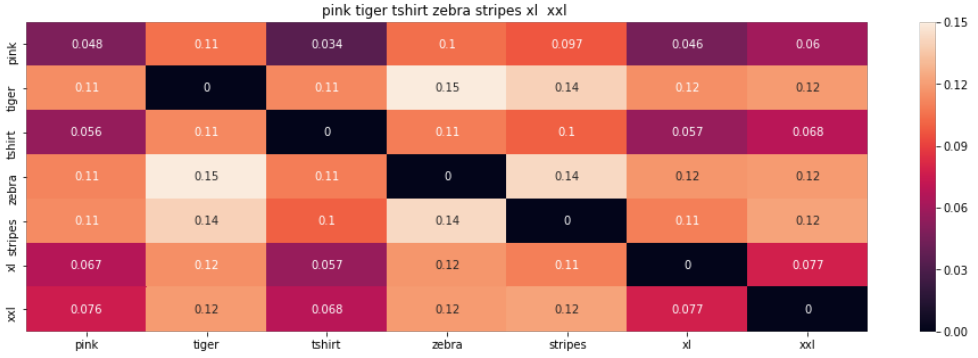
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.0
==========





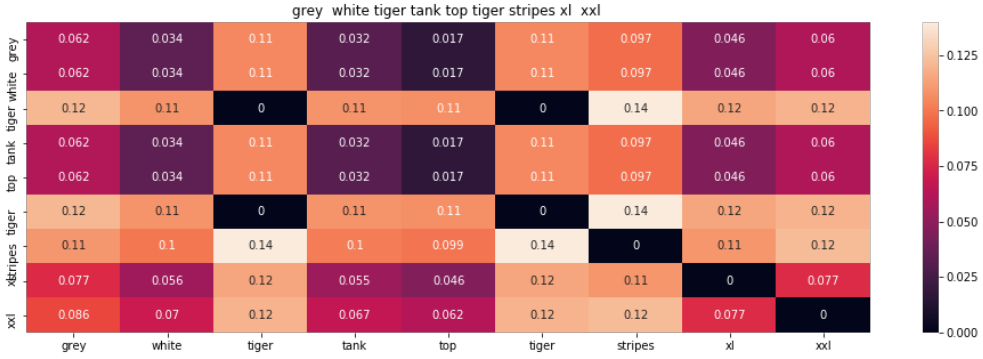
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 0.01671193391084671

==========



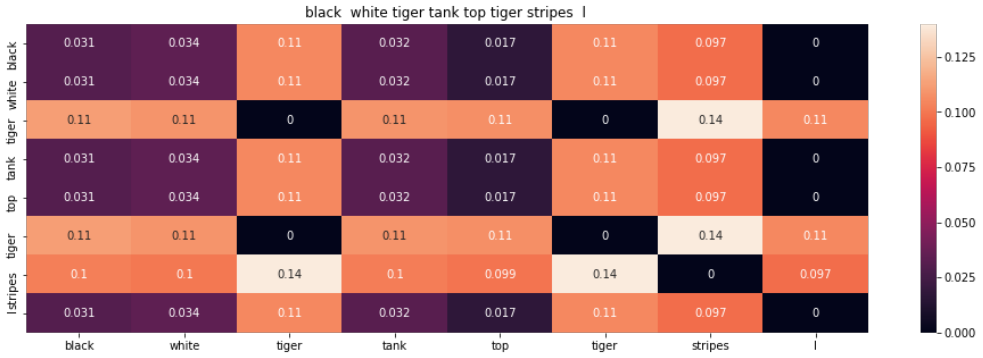
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 0.7197454588431981

==========



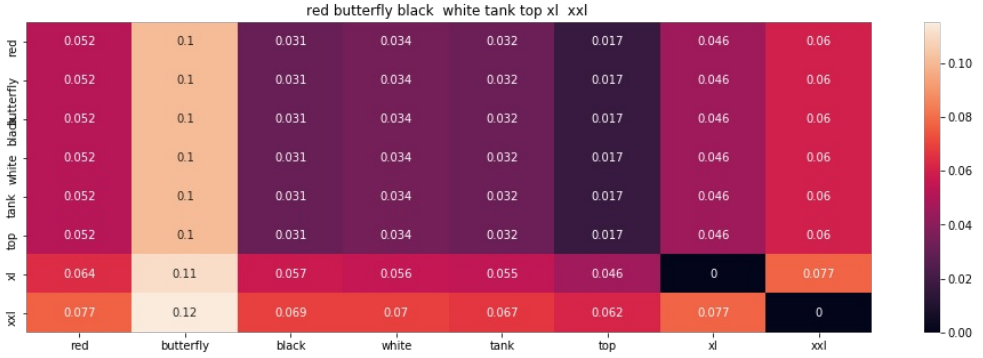
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 0.7220730395931867

==========



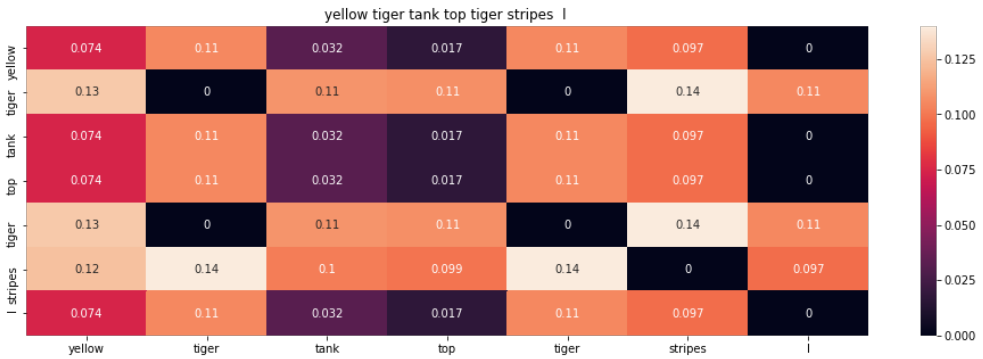
ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 0.7235146762747433

==========



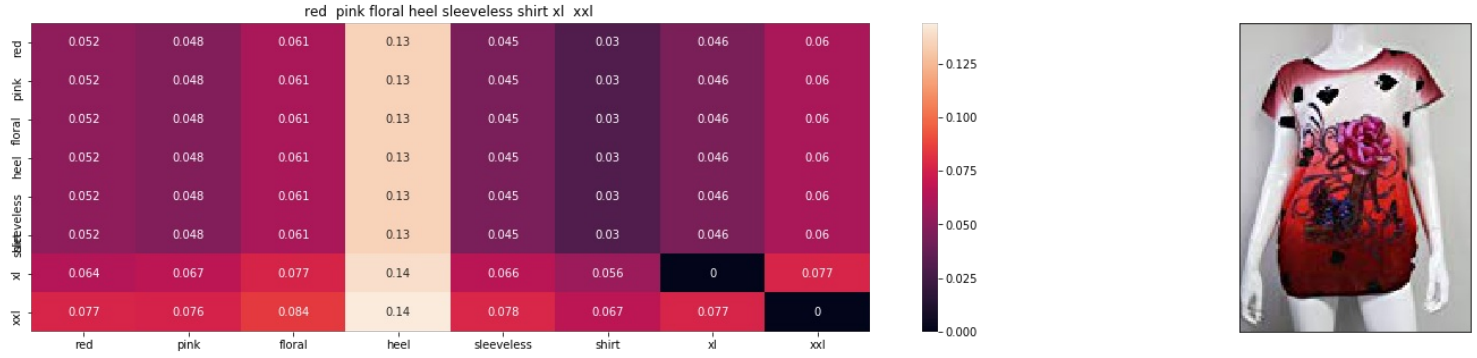
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 0.7249321388024953

==========

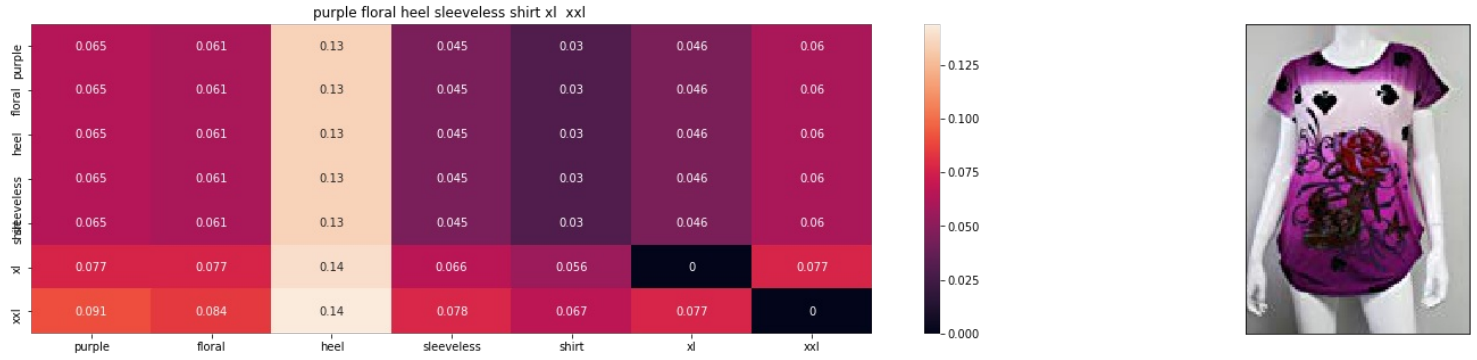


ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 0.725615827919927

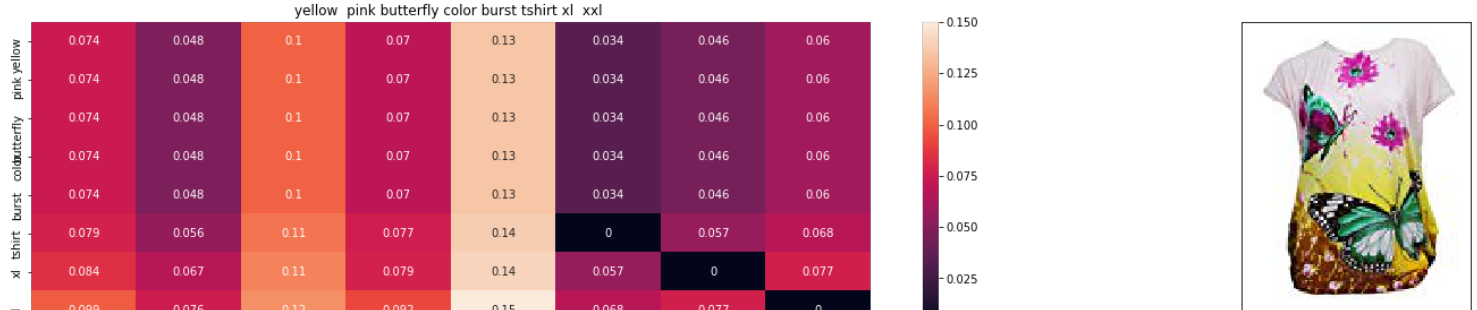
==========

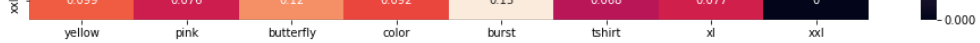


ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 0.725842121424165
==========

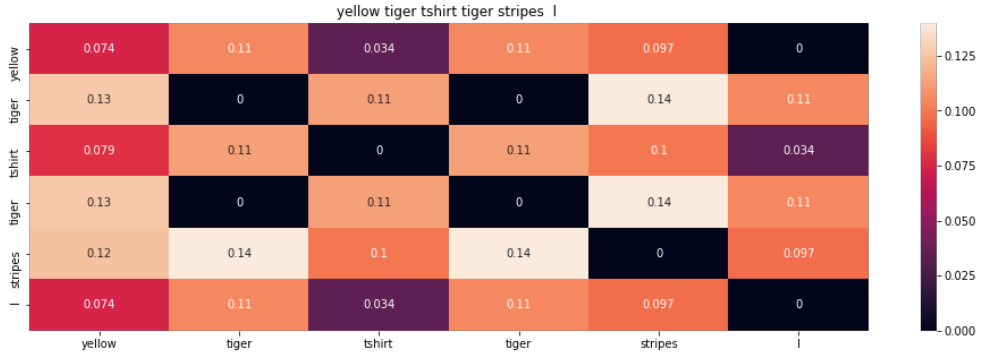


ASIN : B00JV63VC8
Brand : Si Row
euclidean distance from input : 0.7269492940921453
==========

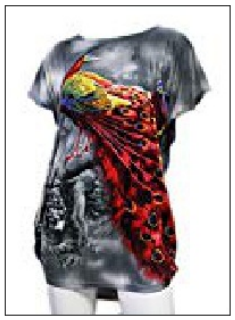




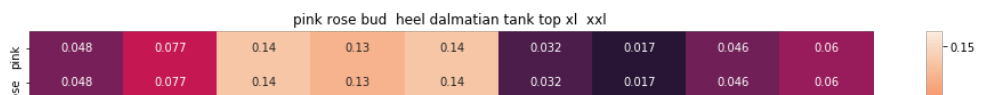
ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 0.7271824585218098

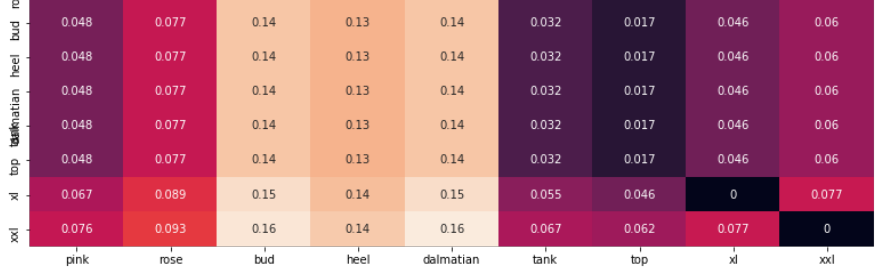


ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 0.727232799114671



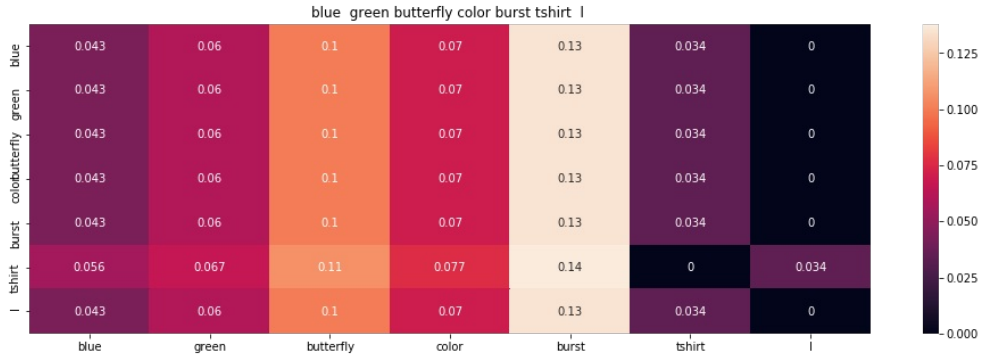
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 0.7274800078887609





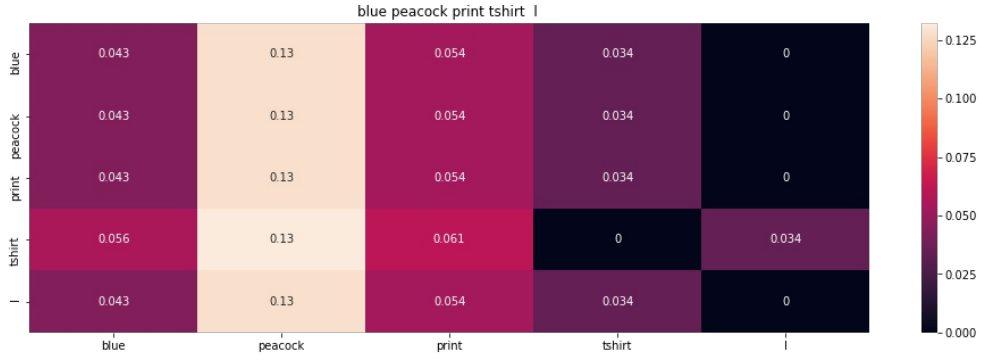
ASIN : B00JXQAX2C
Brand : Si Row
euclidean distance from input : 0.727993790986028

==========



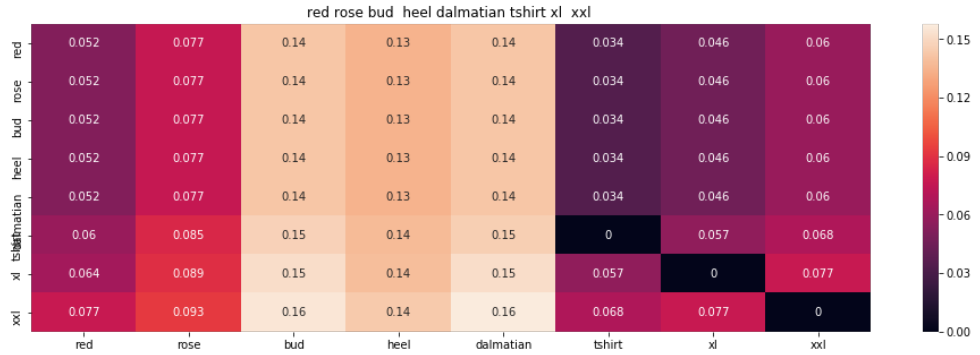
ASIN : B00JXQC0C8
Brand : Si Row
euclidean distance from input : 0.7285432921666768

==========



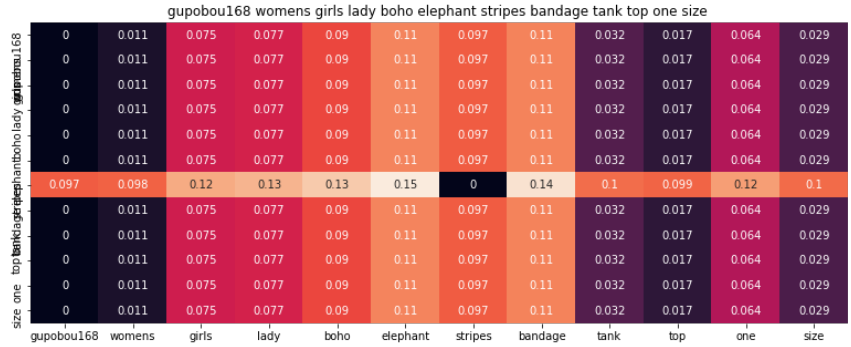
ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 0.7295051308293011

==========



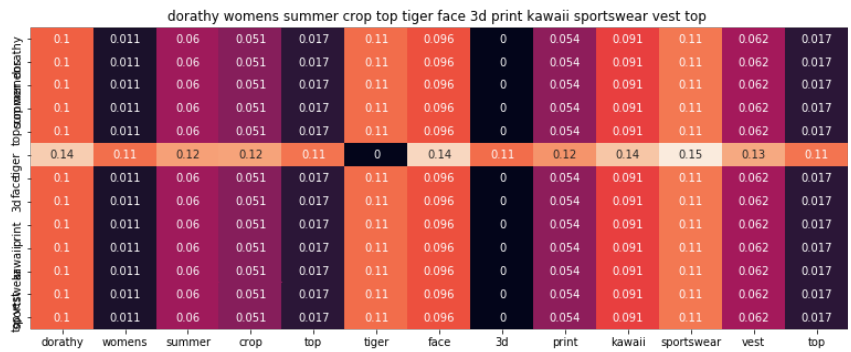
ASIN : B00JXQABB0
Brand : Si Row
euclidean distance from input : 0.7295532124061254

==========



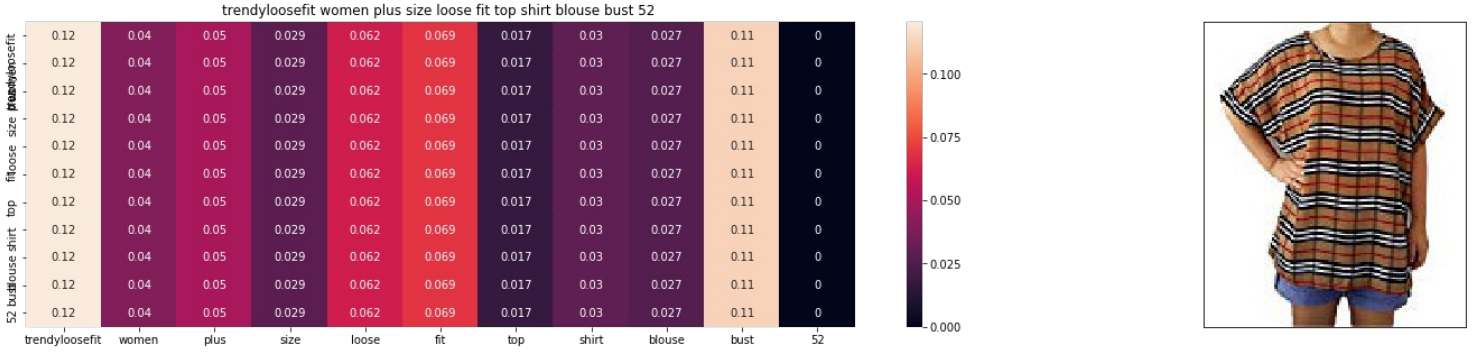
ASIN : B01ER184O6
Brand : GuPoBoU168
euclidean distance from input : 1.1354913249372576

==========



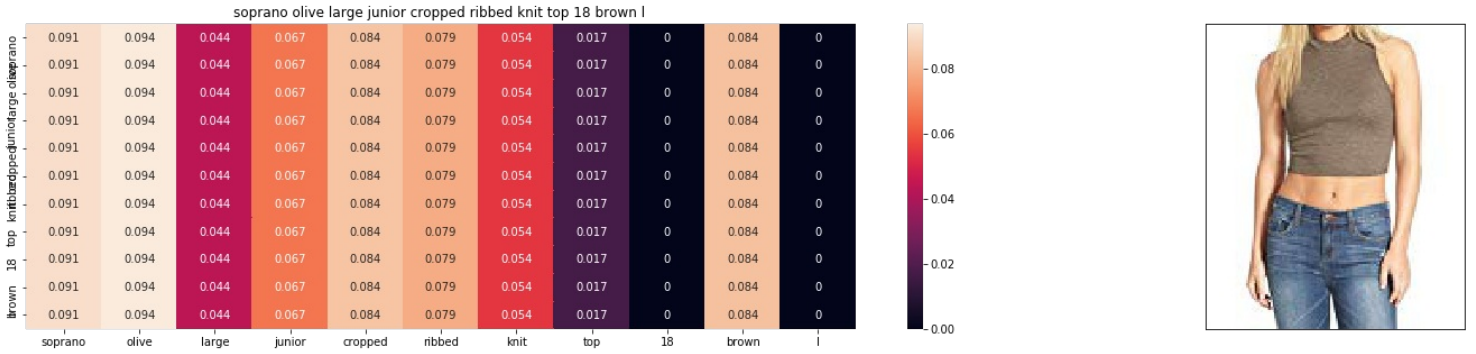
ASIN : B01HDKPMVQ
Brand : Dorathy
euclidean distance from input : 1.1355079754947757

==========



ASIN : B019EGEV9Q
Brand : Trendyloosefit
euclidean distance from input : 1.1357362925767038

==========

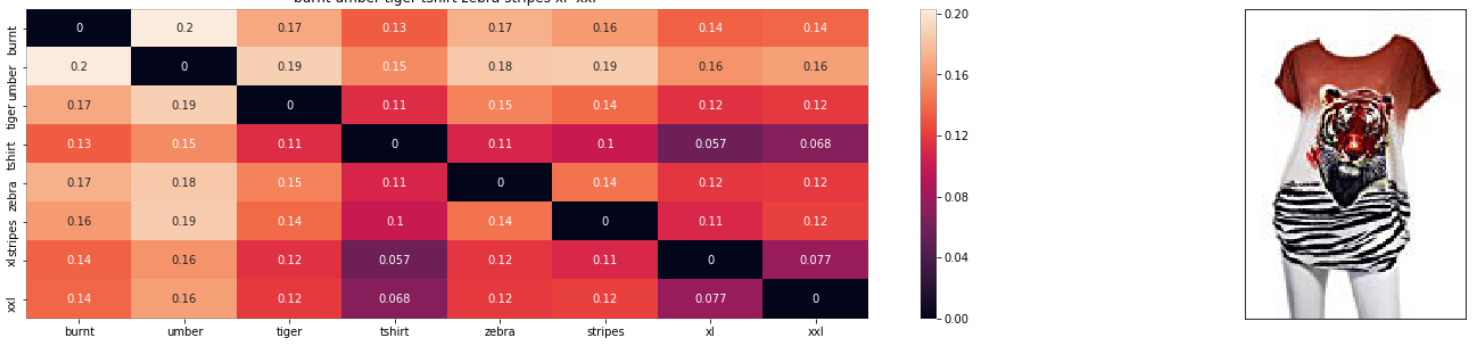


ASIN : B07288KFHF
Brand : Soprano
euclidean distance from input : 1.1357444807647799

==========

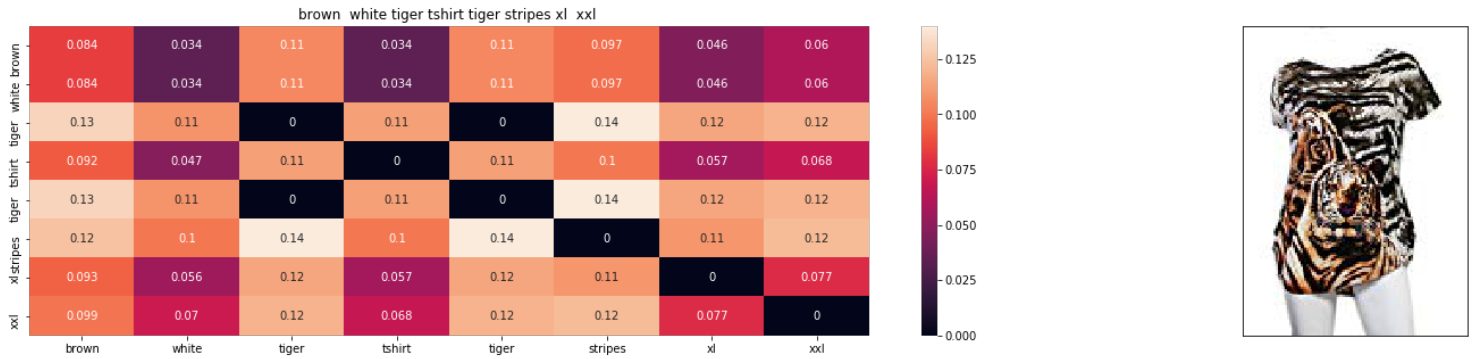
In [42]:

```
# Imp to 'idf'  = 5
# and
# Imp to '[brand + type(shirt/top) + color]' = 50

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



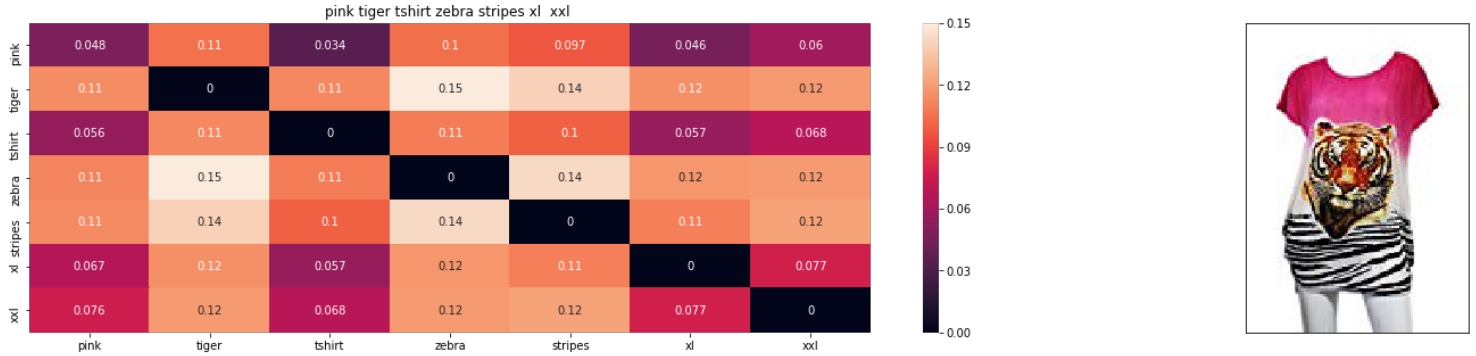
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.0

==========



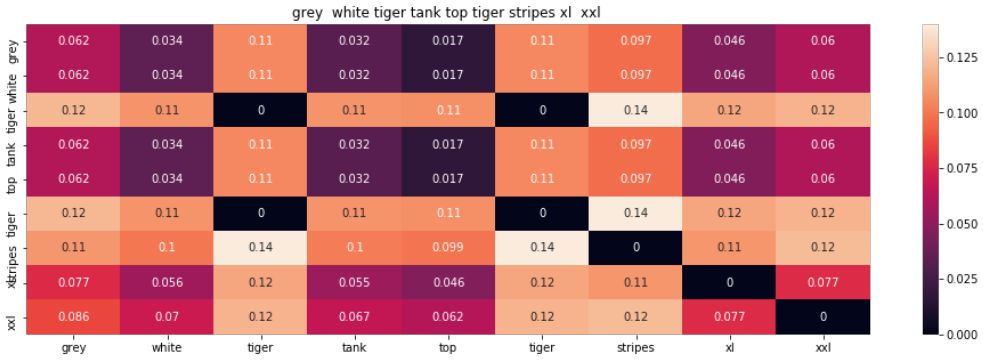
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 0.0030385334383357656

==========



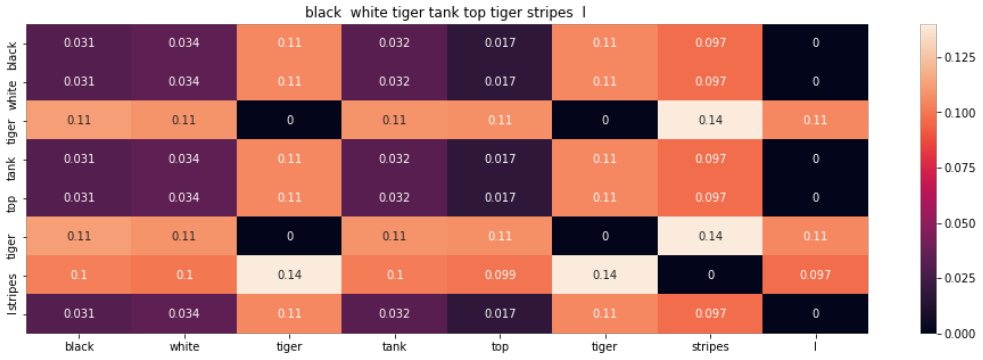
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 1.2879466344585684

==========



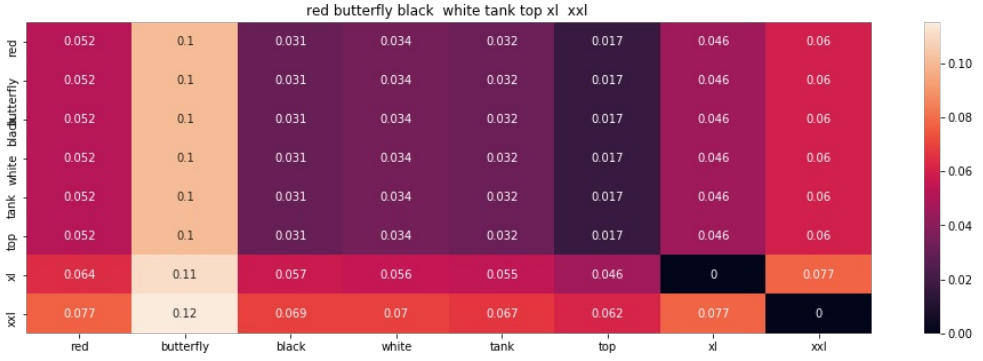
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 1.2883698309585663

==========

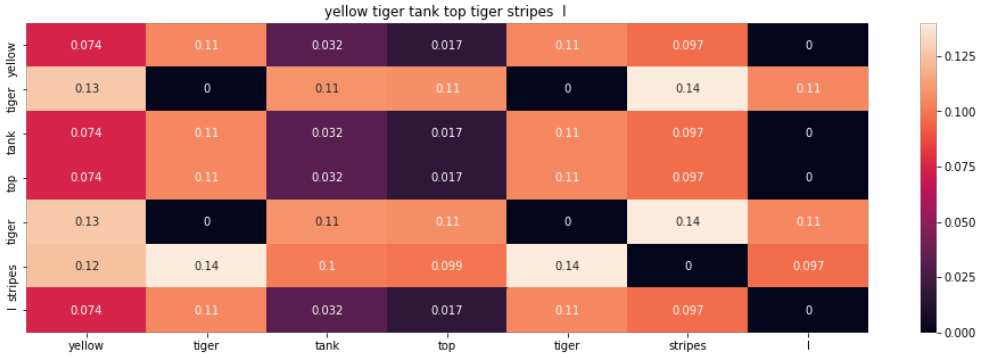


ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 1.2886319467188494

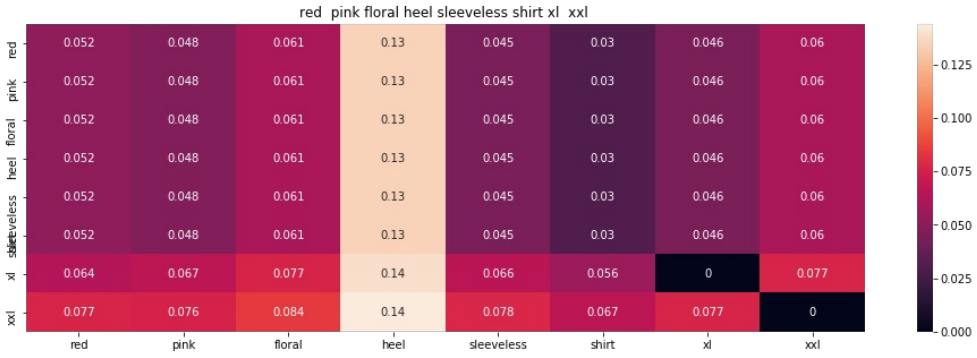
==========



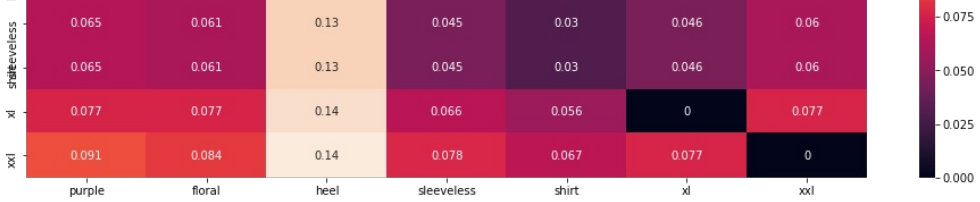
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 1.2888896671784407
==========



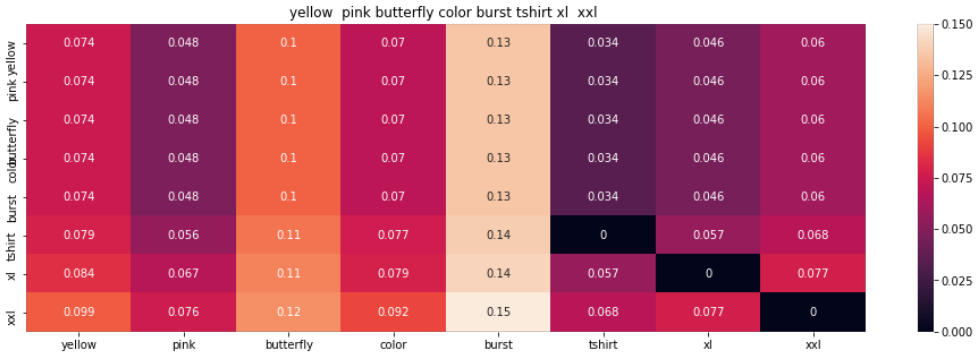
ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 1.289013974290701
==========



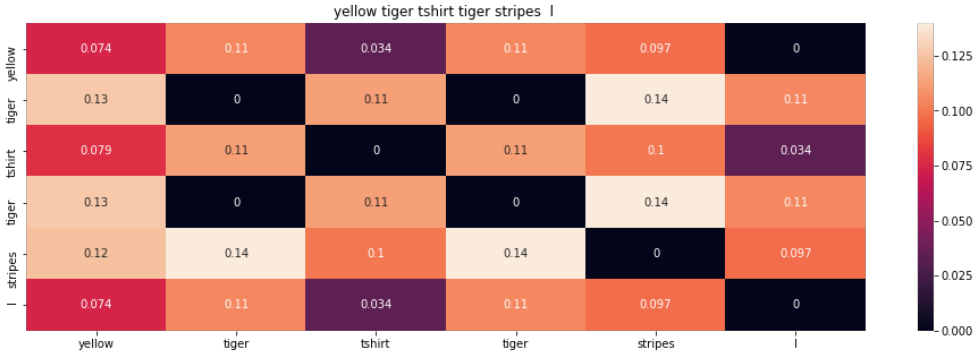
ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 1.2890551185641987
==========



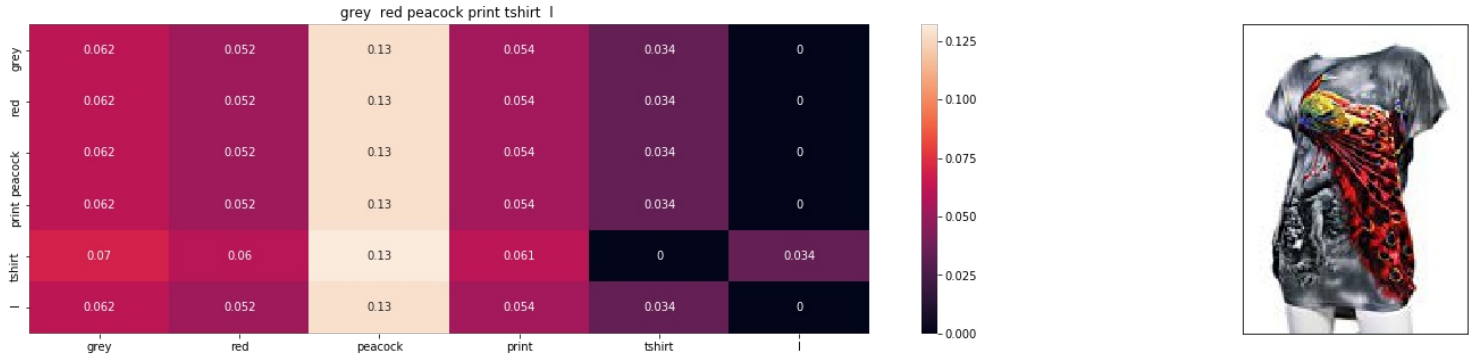
ASIN : B00JV63VC8
Brand : Si Row
euclidean distance from input : 1.2892564226856498



ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 1.2892988162183159

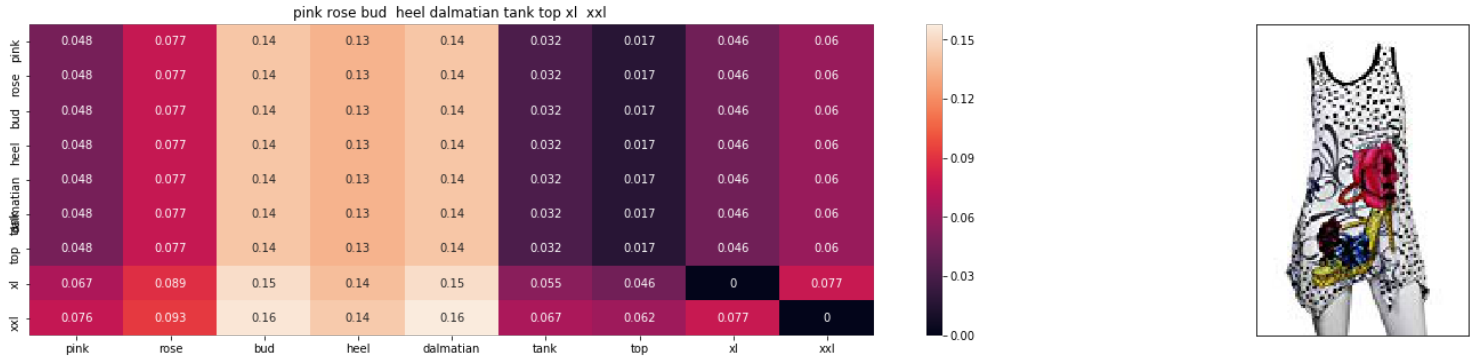


ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 1.2893079690533817



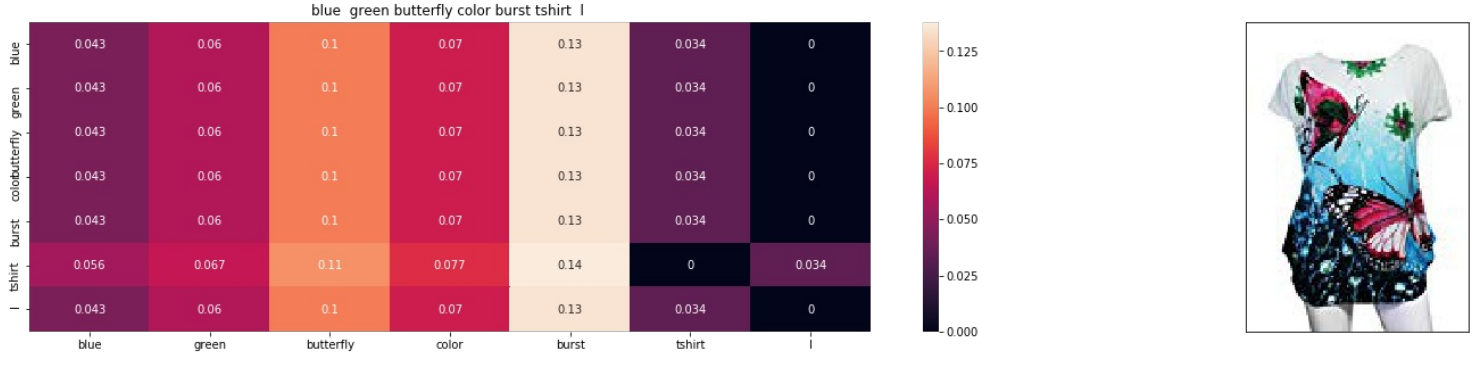
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 1.2893529161032162

==========



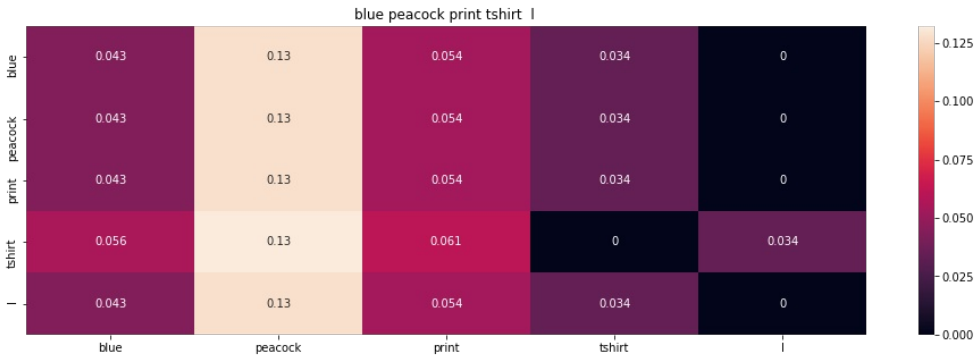
ASIN : B00JXQAX2C
Brand : Si Row
euclidean distance from input : 1.28944633121181

==========



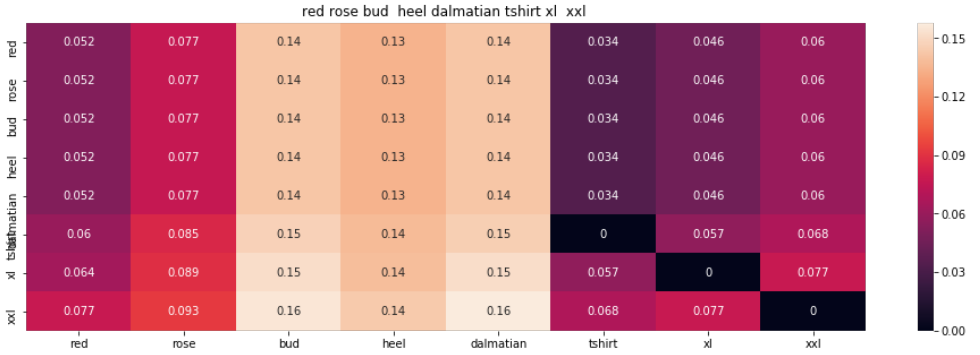
ASIN : B00JXQC0C8
Brand : Si Row
euclidean distance from input : 1.2895462405173828

==========



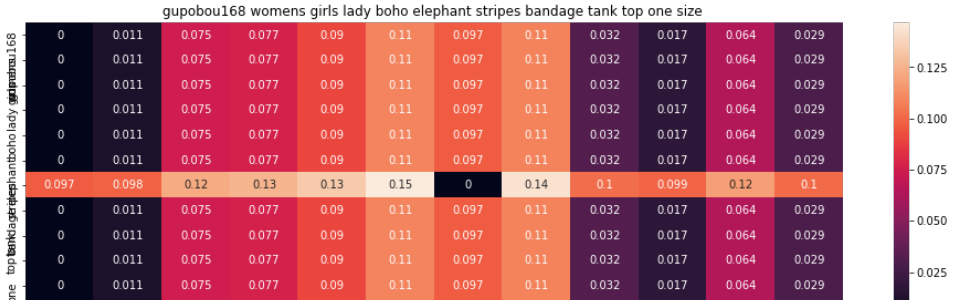
ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 1.2897211202742234

==========



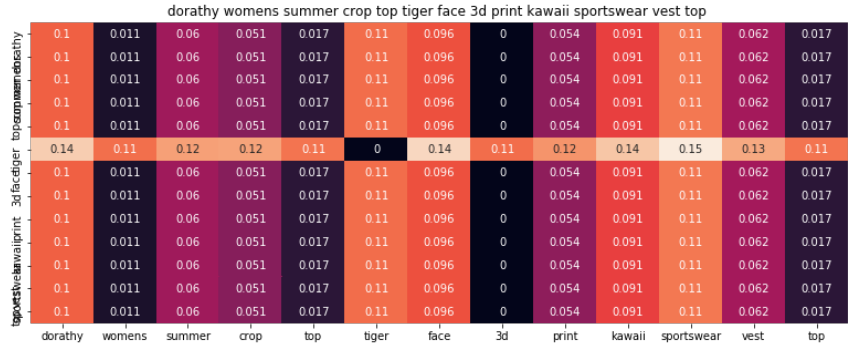
ASIN : B00JXQABB0
Brand : Si Row
euclidean distance from input : 1.2897298623791007

==========



ASIN : B01ER184O6
Brand : GuPoBoU168
euclidean distance from input : 2.0359631315793294

==========

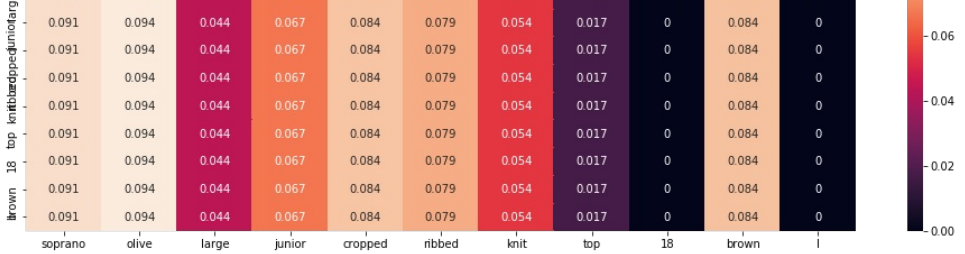


ASIN : B01HDKPMVQ
Brand : Dorathy
euclidean distance from input : 2.0359661589534235

==========

ASIN : B019EGEV9Q
Brand : Trendyloosefit
euclidean distance from input : 2.036007671150138

==========



ASIN : B07288KFHF
Brand : Soprano
euclidean distance from input : 2.036009159911606
==========

Keras and Tensorflow to extract features

In [43]:

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
from keras import applications
import requests
from PIL import Image
from io import BytesIO
```

```
Using TensorFlow backend.
```

In [44]:

```
data.shape
```

Out[44]:

(16042, 7)

In [45]:

```
data.head(1)
```

Out[45]:

	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

In [46]:

```
data['medium_image_url'].iloc[0]
```

Out[46]:

'https://images-na.ssl-images-amazon.com/images/I/31VspXbakvL._SL160_.jpg'

In [47]:

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



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

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

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

```
# each image is converted into 25088 length dense-vector


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

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


def save_bottlebeck_features():

    #Function to compute VGG-16 CNN for image 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()

'''
```

Out[47]:

```
"\n# dimensions of our images.\nimg_width, img_height = 224, 224\n\ntop_model_weights_path = 'bottleneck_fc_model.h5'\ntrain_data_dir = 'images
2/'\nnb_train_samples = 16042\nepochs = 50\nbatch_size = 1\n\n\ndef save_bottlebeck_features():\n    \n    #Function to compute VGG-16 CNN for i
mage feature extraction.\n    \n    asins = []\n    datagen = ImageDataGenerator(rescale=1. / 255)\n    \n    # build the VGG16 network\n    model = ap
plications.VGG16(include_top=False, weights='imagenet')\n    generator = datagen.flow_from_directory(\n        train_data_dir,\n        target_size=(img
_width, img_height),\n        batch_size=batch_size,\n        class_mode=None,\n        shuffle=False)\n\n    for i in generator.filenames:\n        asins.app
end(i[2:-5])\n\n    bottleneck_features_train = model.predict_generator(generator, nb_train_samples // batch_size)\n    bottleneck_features_train = bot
tleneck_features_train.reshape((16042,25088))\n    \n    np.save(open('16k_data_cnn_features.npy', 'wb'), bottleneck_features_train)\n    np.save(op
en('16k_data_cnn_feature_asins.npy', 'wb'), np.array(asins))\n    \n\nsave_bottlebeck_features()\n\n"
```

Visual features based product similarity.

In [48]:

```
#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)
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.amzon.com/dp/'+ asins[indices[i]])

get_similar_products_cnn(12566, 20)
```

Product Title: burnt umber tiger tshirt zebra stripes xl xxl
Euclidean Distance from input image: 6.32596e-06
Amazon Url: www.amzon.com/dp/B00JXQB5FQ

Product Title: pink tiger tshirt zebra stripes xl xxl
Euclidean Distance from input image: 30.05017
Amazon Url: www.amzon.com/dp/B00JXQASS6

Product Title: yellow tiger tshirt tiger stripes l
Euclidean Distance from input image: 41.261116
Amazon Url: www.amzon.com/dp/B00JXQCUIC

Product Title: brown white tiger tshirt tiger stripes xl xxl
Euclidean Distance from input image: 44.000156
Amazon Url: www.amzon.com/dp/B00JXQCWTO

Product Title: kawaii pastel tops tees pink flower design
Euclidean Distance from input image: 47.38248

Amazon Url: www.amzon.com/dp/B071FCWD97

Product Title: womens thin style tops tees pastel watermelon print
Euclidean Distance from input image: 47.71842
Amazon Url: www.amzon.com/dp/B01JUNHBRM

Product Title: kawaii pastel tops tees baby blue flower design
Euclidean Distance from input image: 47.90206
Amazon Url: www.amzon.com/dp/B071SBCY9W

Product Title: edv cheetah run purple multi xl
Euclidean Distance from input image: 48.046482
Amazon Url: www.amzon.com/dp/B01CUPYBM0

Product Title: danskin womens vneck loose performance tee xsmall pink ombre
Euclidean Distance from input image: 48.101837
Amazon Url: www.amzon.com/dp/B01F7PHXY8

Product Title: summer alpaca 3d pastel casual loose tops tee design
Euclidean Distance from input image: 48.118866
Amazon Url: www.amzon.com/dp/B01I80A93G

Product Title: miss chievous juniors striped peplum tank top medium shadowpeach
Euclidean Distance from input image: 48.13122
Amazon Url: www.amzon.com/dp/B0177DM70S

Product Title: red pink floral heel sleeveless shirt xl xxl
Euclidean Distance from input image: 48.16945
Amazon Url: www.amzon.com/dp/B00JV63QQE

Product Title: moana logo adults hot v neck shirt black xxl
Euclidean Distance from input image: 48.256786
Amazon Url: www.amzon.com/dp/B01LX6H43D

Product Title: abaday multicolor cartoon cat print short sleeve longline shirt large
Euclidean Distance from input image: 48.265686
Amazon Url: www.amzon.com/dp/B01CR57YY0

Product Title: kawaii cotton pastel tops tees peach pink cactus design
Euclidean Distance from input image: 48.362602
Amazon Url: www.amzon.com/dp/B071WYLBZS

Product Title: chicago chicago 18 shirt women pink
Euclidean Distance from input image: 48.383606
Amazon Url: www.amzon.com/dp/B01GXAZTRY

Product Title: yichun womens tiger printed summer tshirts tops
Euclidean Distance from input image: 48.449356
Amazon Url: www.amzon.com/dp/B010NN9RXO

Product Title: nancy lopez whimsy short sleeve whiteblacklemon drop xs
Euclidean Distance from input image: 48.47889
Amazon Url: www.amzon.com/dp/B01MPX6IDX

Product Title: womens tops tees pastel peach ice cream cone print
Euclidean Distance from input image: 48.557957
Amazon Url: www.amzon.com/dp/B0734GRKZL

Product Title: uswomens mary j blige without tshirts shirt
Euclidean Distance from input image: 48.614372
Amazon Url: www.amzon.com/dp/B01M0XXFKK

Assignment

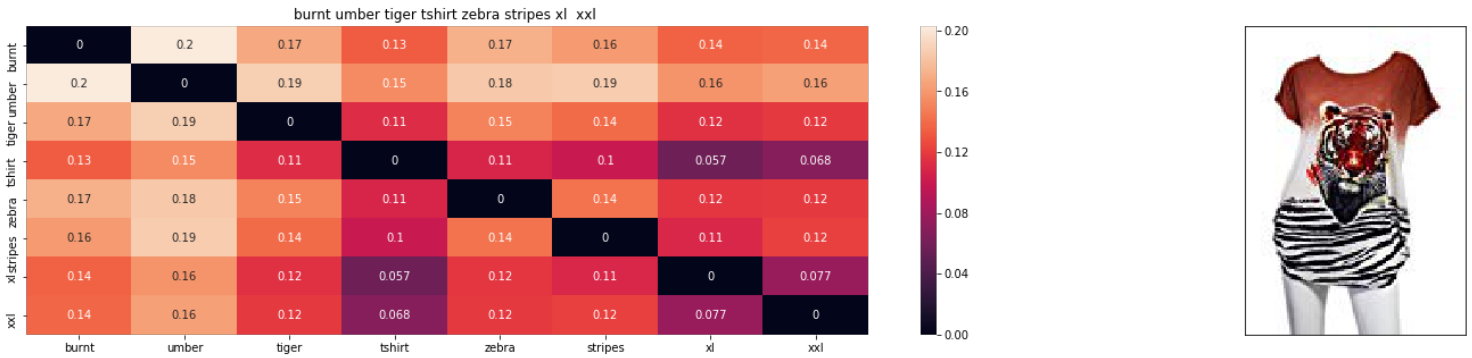
In [53]:

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

```
    # pairwise_dist will store the distance from given input apparel to all remaining 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
    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]) # 3 items [brand + type(shirt/top) + color]
    image = pairwise_distances(bottleneck_features_train, bottleneck_features_train[doc_id].reshape(1,-1))
    pairwise_dist   = (w1 * idf_w2v_dist +  w2 * ex_feat_dist + w3 * image) / float(w1 + w2 + w3)

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

    #data frame indices of the 20 smallest distace's
    df_indices = list(data.index[indices])

    for i in range(0, len(indices)):
        # sentence1, sentence2, url, doc_id1, doc_id2, df_id1, df_id2, model
        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])

        rows = data[['medium_image_url','title']].loc[data['asin']==asins[indices[i]]]
        for indx, row in rows.iterrows():
            display(Image(url=row['medium_image_url'], embed=True))
            print('Product Title: ', row['title'])
            print('Euclidean Distance from input image:', pdists[i])
            print('Amazon Url: www.amzon.com/dp/'+ asins[indices[i]])
        print('='*125)
```

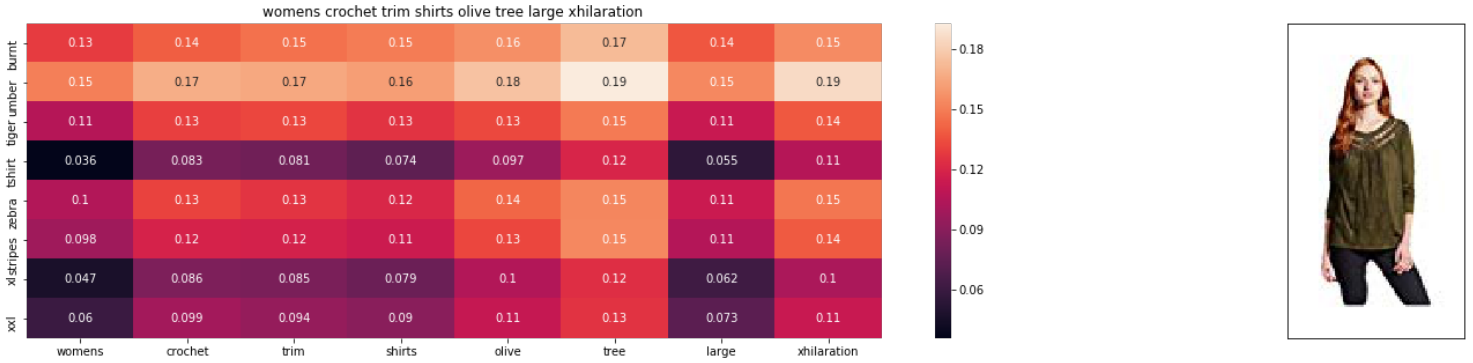
In [54]:

```
# https://stackoverflow.com/a/10749029/10219869
import PIL.Image

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

# same imp to 'idf' and '[brand + type(shirt/top) + color]' and 'Image'

idf_exfeat_image(12566, 5, 5, 5, 10)
```



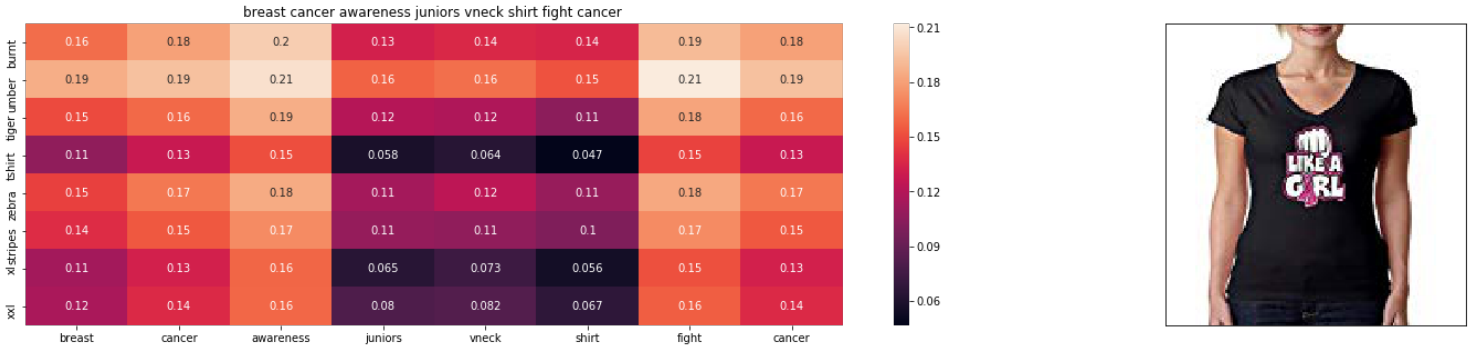
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 2.5030795465378714e-06

Product Title: foxcroft nyc womens pinpoint oxford shirt noniron stretch poplin blouse xlarge white
Euclidean Distance from input image: 2.5030795465378714e-06
Amazon Url: www.amzon.com/dp/B072277HVB
==========



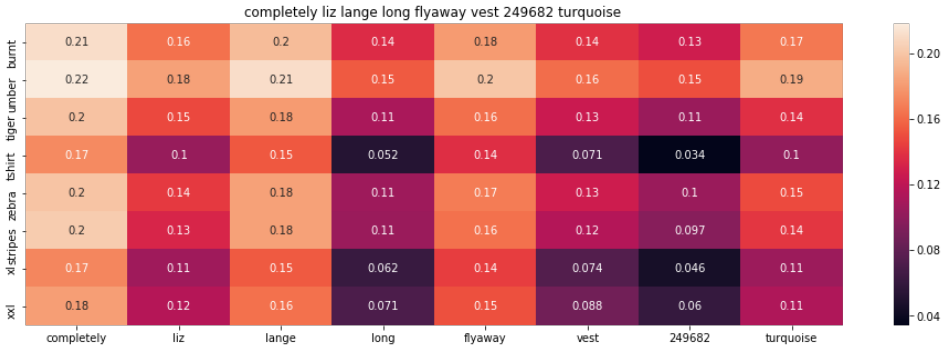
ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 13.352600416782018

Product Title: kiind longsleeve swing top white
Euclidean Distance from input image: 13.352600416782018
Amazon Url: www.amzon.com/dp/B0142Q3C6G
==========



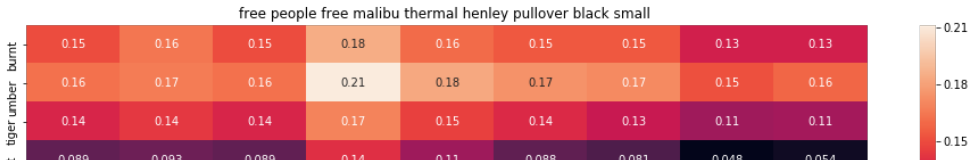
ASIN : B016CU40IY
Brand : Juiceclouds
euclidean distance from input : 13.671048305988586

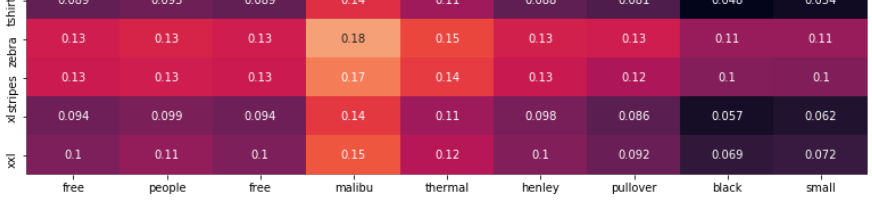
Product Title: size comfortblend shirred crewneck longsleeve tshirt white 2x
Euclidean Distance from input image: 13.671048305988586
Amazon Url: www.amzon.com/dp/B01N5YHV9R
==========



ASIN : B074LTBWSW
Brand : Liz Lange
euclidean distance from input : 14.080883070074037

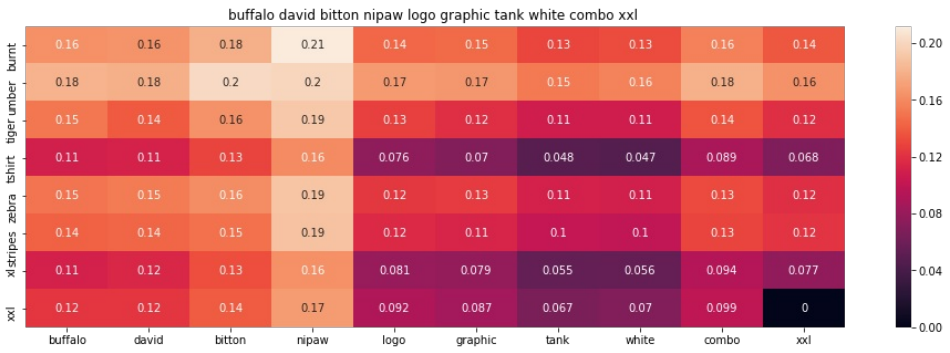
Product Title: karen scott womens plus short sleeve solid henley top white 1x
Euclidean Distance from input image: 14.080883070074037
Amazon Url: www.amzon.com/dp/B0196H3Z2W
==========





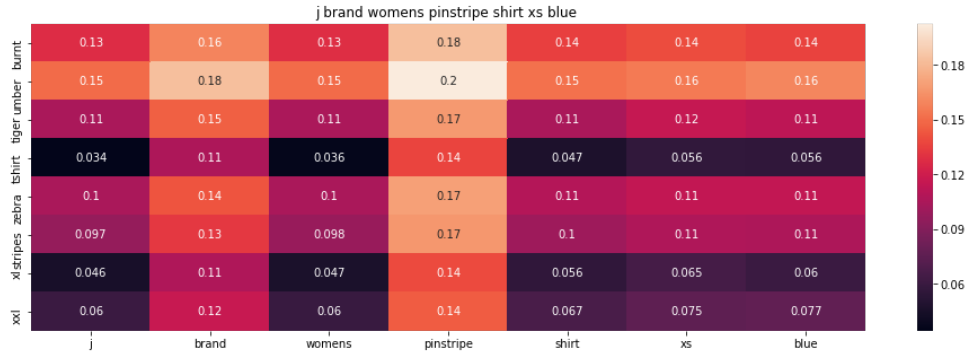
ASIN : B074MXY984
Brand : We The Free
euclidean distance from input : 14.136804455518723

Product Title: karen scott wmen short sleeve henley top plus size 0x bright white
Euclidean Distance from input image: 14.136804455518723
Amazon Url: www.amzon.com/dp/B0196H7P5A
==========



ASIN : B018H5AZXQ
Brand : Buffalo
euclidean distance from input : 14.144593731803214

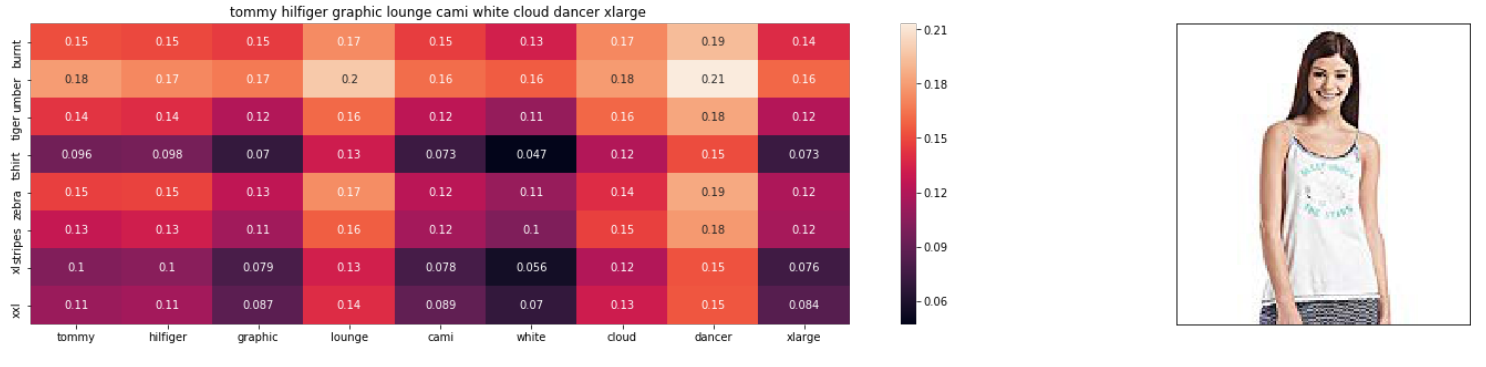
Product Title: free people scoopneck highlow thermal tunic oatmeal heather
Euclidean Distance from input image: 14.144593731803214
Amazon Url: www.amzon.com/dp/B00ZH2X45E
==========



ASIN : B06XYP1X1F
Brand : J Brand Jeans
euclidean distance from input : 14.264338535201937

Product Title: masion jules longsleeve paneled sweatshirt xxl grey
Euclidean Distance from input image: 14.264338535201937
Amazon Url: www.amzon.com/dp/B06X16WTFX

==========

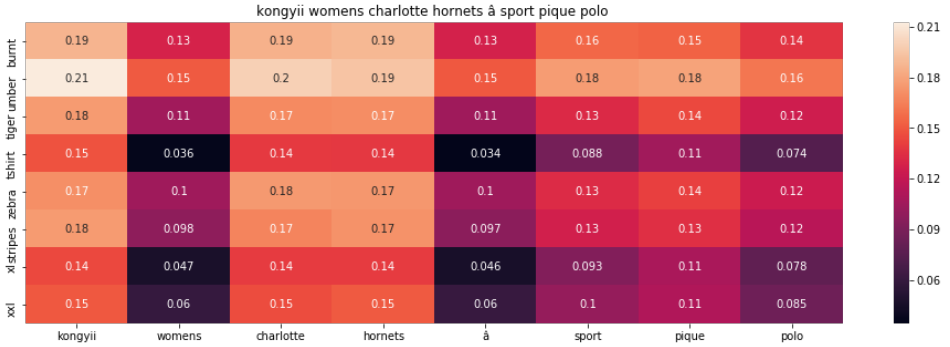


ASIN : B01BMSFYW2
Brand : igertommy hilf
euclidean distance from input : 14.313836552830924

Product Title: vitamina usa soft vneckline belted blouse st6356 wht
Euclidean Distance from input image: 14.313836552830924
Amazon Url: www.amzon.com/dp/B0155ICSDS

==========

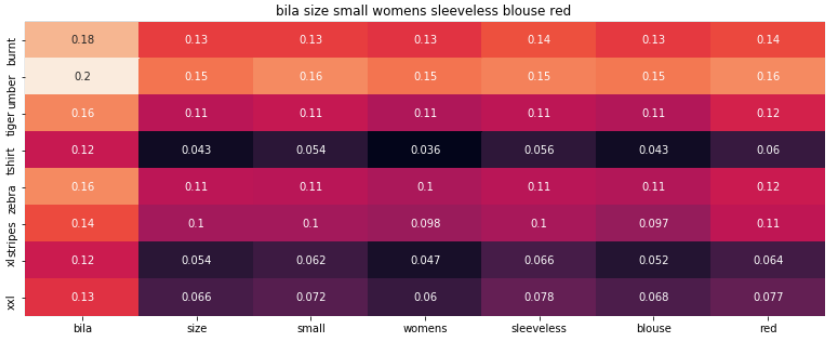
==========



ASIN : B01FJVZST2
Brand : KONGYII
euclidean distance from input : 14.47299332873054

Product Title: dg2 diane gilman turtleneck fleece pullover pockets small gray
Euclidean Distance from input image: 14.47299332873054
Amazon Url: www.amzon.com/dp/B01MSP0KAM

==========

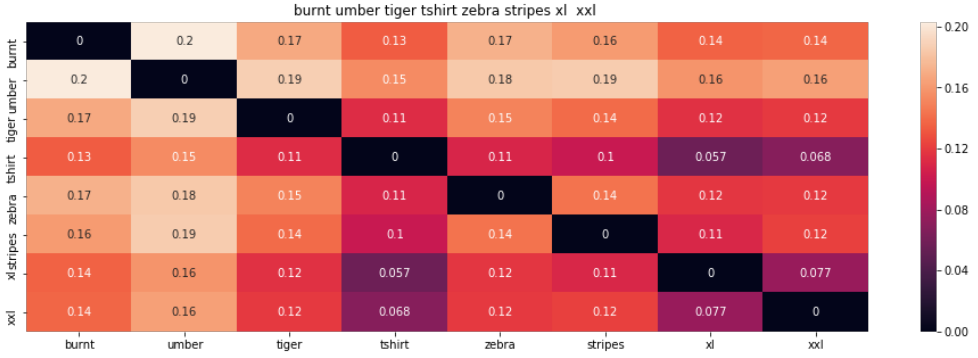


ASIN : B01L7ROZNC
Brand : Bila
euclidean distance from input : 14.573514301141378

Product Title: jessica simpson womens frida peasant top antiquewhite x small
Euclidean Distance from input image: 14.573514301141378
Amazon Url: www.amzon.com/dp/B01KW297J0
==========

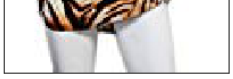
In [55]:

```
# More imp to 'idf' and '[brand + type(shirt/top) + color]' and less importance to 'Image'

idf_exfeat_image(12566, 50, 50, 5, 10)
```



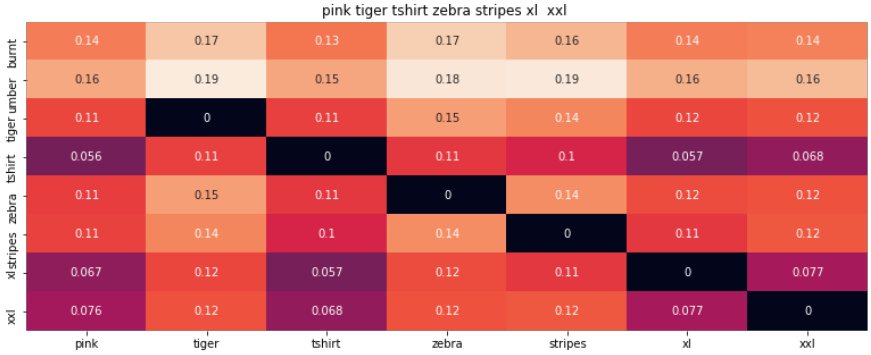
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 3.5758279236255304e-07

Product Title: foxcroft nyc womens pinpoint oxford shirt noniron stretch poplin blouse xlarge white
Euclidean Distance from input image: 3.5758279236255304e-07
Amazon Url: www.amzon.com/dp/B072277HVB
==========



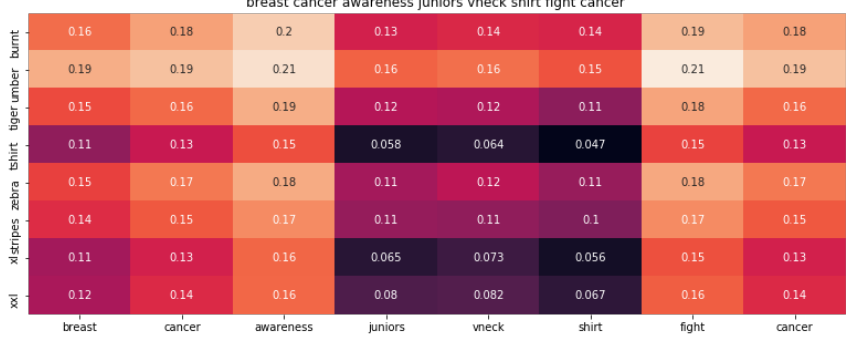
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 2.661051625297183

Product Title: exotic india yellow gray jamawar wrap fauxfur collar
Euclidean Distance from input image: 2.661051625297183
Amazon Url: www.amzon.com/dp/B073ZHRBV8
==========



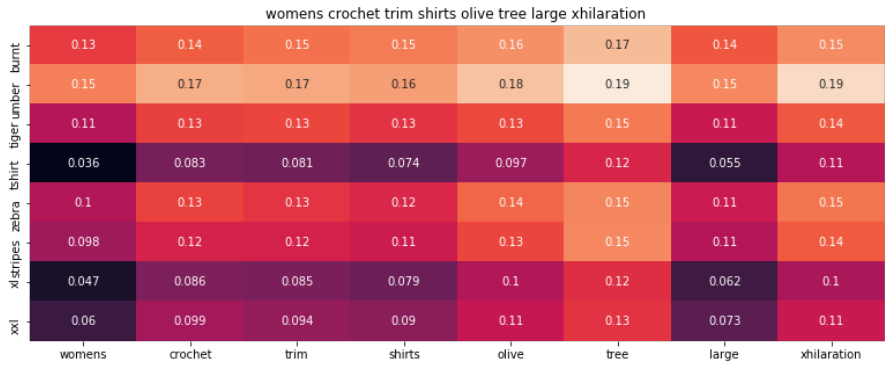
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 2.995462884393842

Product Title: kirkland signature ladies size medium cotton vneck tee white
Euclidean Distance from input image: 2.995462884393842
Amazon Url: www.amzon.com/dp/B073ZRD5FL
==========



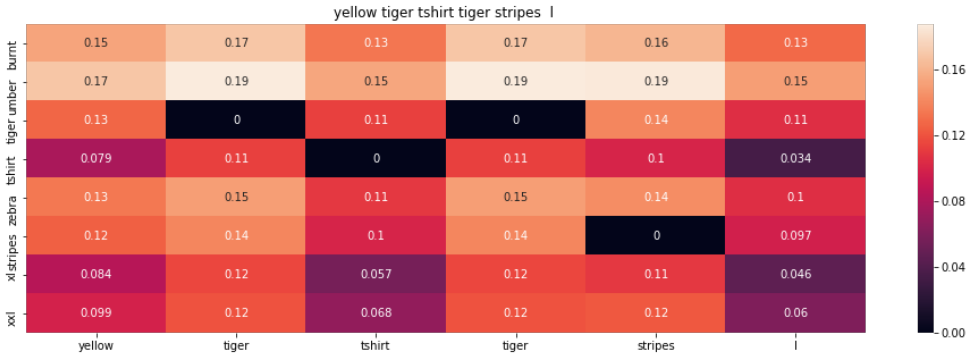
ASIN : B016CU40IY
Brand : Juiceclouds
euclidean distance from input : 3.1101593411540374

Product Title: size comfortblend shirred crewneck longsleeve tshirt white 2x
Euclidean Distance from input image: 3.1101593411540374
Amazon Url: www.amzon.com/dp/B01N5YHV9R
==========



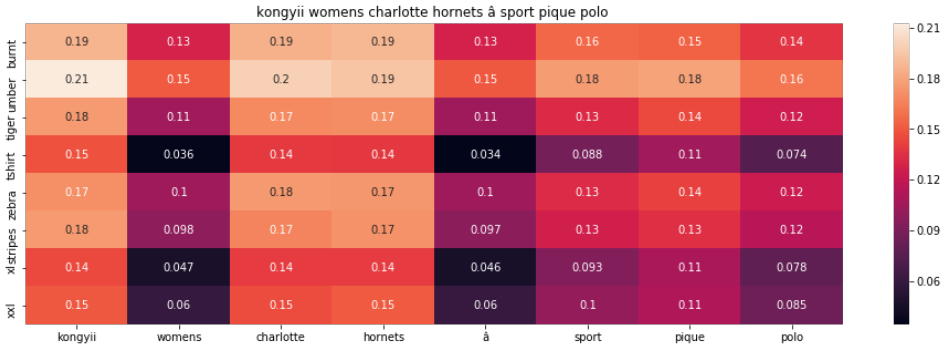
ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 3.1375270483643036

Product Title: kiind longsleeve swing top white
Euclidean Distance from input image: 3.1375270483643036
Amazon Url: www.amzon.com/dp/B0142Q3C6G
==========



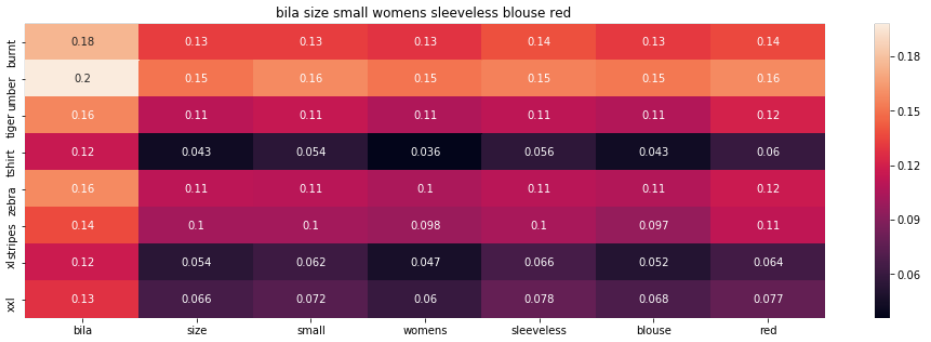
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 3.1772247134521714

Product Title: tahari asl ruffledfront polkadot top blackwhite xl
Euclidean Distance from input image: 3.1772247134521714
Amazon Url: www.amzon.com/dp/B074MZTZS5
==========



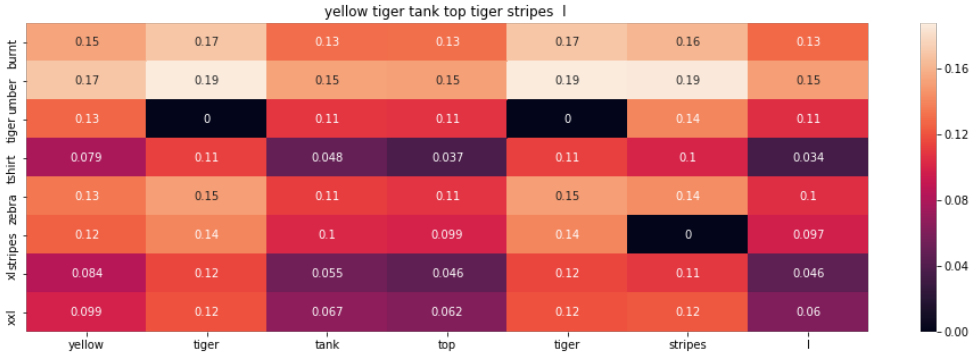
ASIN : B01FJVZST2
Brand : KONGYII
euclidean distance from input : 3.221034693377751

Product Title: dg2 diane gilman turtleneck fleece pullover pockets small gray
Euclidean Distance from input image: 3.221034693377751
Amazon Url: www.amzon.com/dp/B01MSP0KAM
==========



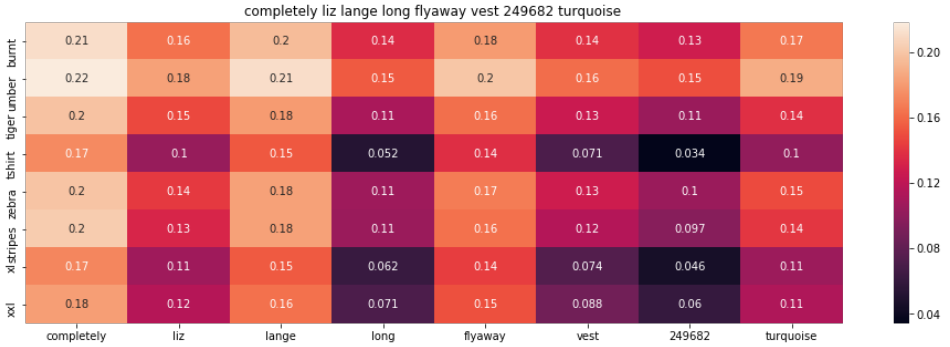
ASIN : B01L7ROZNC
Brand : Bila
euclidean distance from input : 3.2322586771424

Product Title: jessica simpson womens frida peasant top antiquewhite x small
Euclidean Distance from input image: 3.2322586771424
Amazon Url: www.amzon.com/dp/B01KW297J0
==========



ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 3.237676695587036

Product Title: blouseuploter cute cartoon vest summer pajamas
Euclidean Distance from input image: 3.237676695587036
Amazon Url: www.amzon.com/dp/B01LPJRYI0
==========

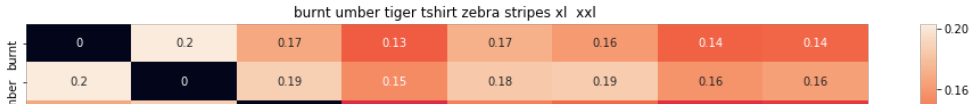


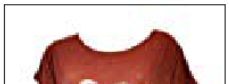
ASIN : B074LTBWSW
Brand : Liz Lange
euclidean distance from input : 3.2451771966652014

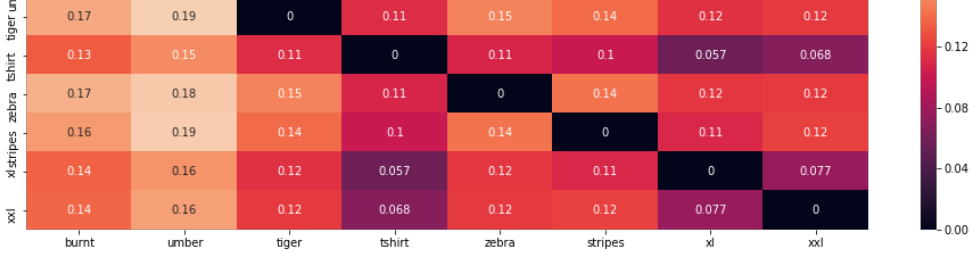
Product Title: karen scott womens plus short sleeve solid henley top white 1x
Euclidean Distance from input image: 3.2451771966652014
Amazon Url: www.amzon.com/dp/B0196H3Z2W
==========

In [56]:

```
# Less imp to 'idf' and '[brand + type(shirt/top) + color]' and More importance to 'Image'

idf_exfeat_image(12566, 5, 5, 50, 10)
```



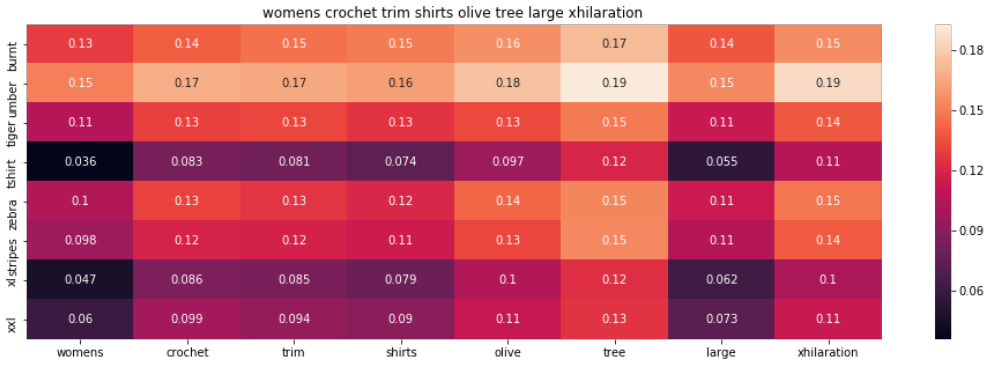




ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 6.257698987610638e-06

Product Title: foxcroft nyc womens pinpoint oxford shirt noniron stretch poplin blouse xlarge white
Euclidean Distance from input image: 6.257698987610638e-06
Amazon Url: www.amzon.com/dp/B072277HVB

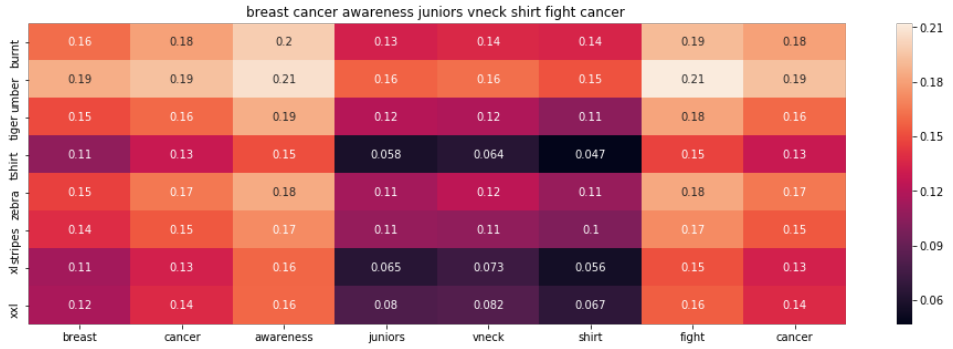
==========



ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 31.22897830040319

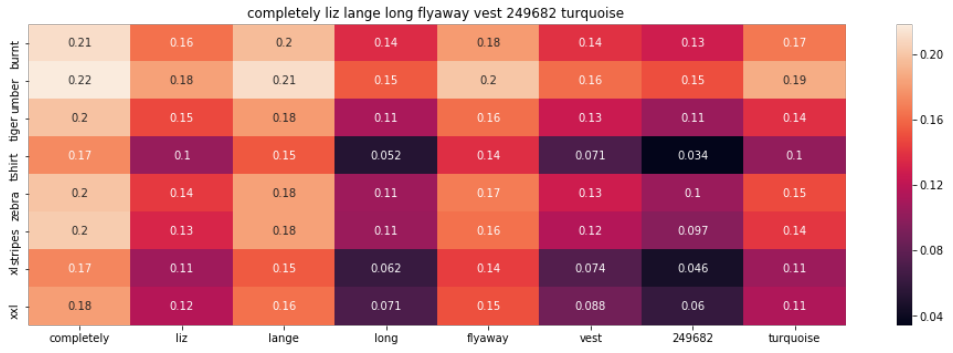
Product Title: kiind longsleeve swing top white
Euclidean Distance from input image: 31.22897830040319
Amazon Url: www.amzon.com/dp/B0142Q3C6G

==========



ASIN : B016CU40IY
Brand : Juiceclouds
euclidean distance from input : 32.152603995442455

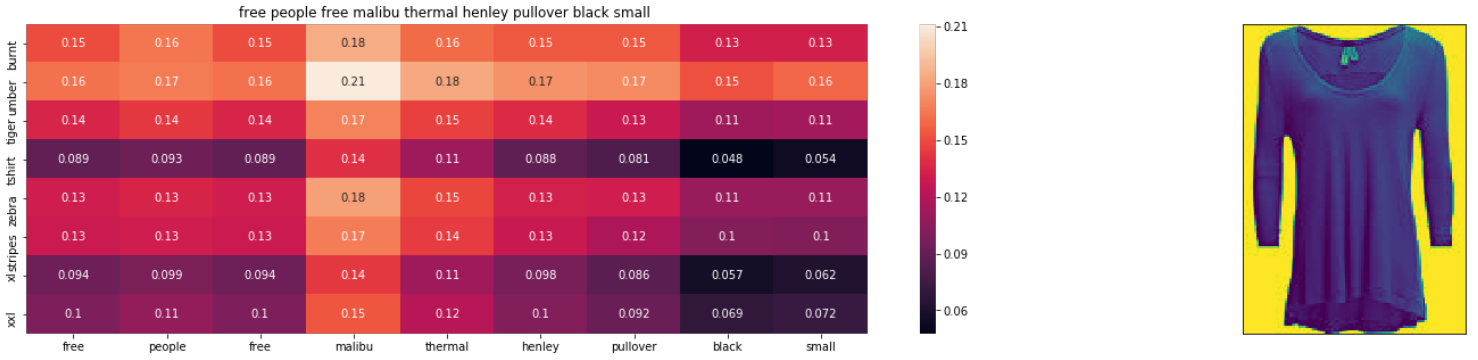
Product Title: size comfortblend shirred crewneck longsleeve tshirt white 2x
Euclidean Distance from input image: 32.152603995442455
Amazon Url: www.amzon.com/dp/B01N5YHV9R
==========



ASIN : B074LTBWSW
Brand : Liz Lange
euclidean distance from input : 33.04336835052632

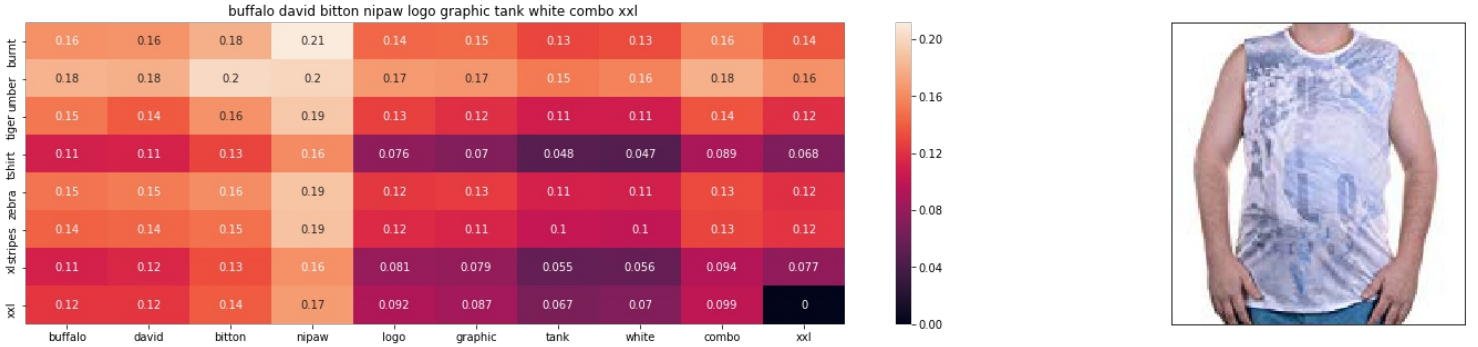
Product Title: karen scott womens plus short sleeve solid henley top white 1x

Euclidean Distance from input image: 33.04336835052632
Amazon Url: www.amzon.com/dp/B0196H3Z2W
==========



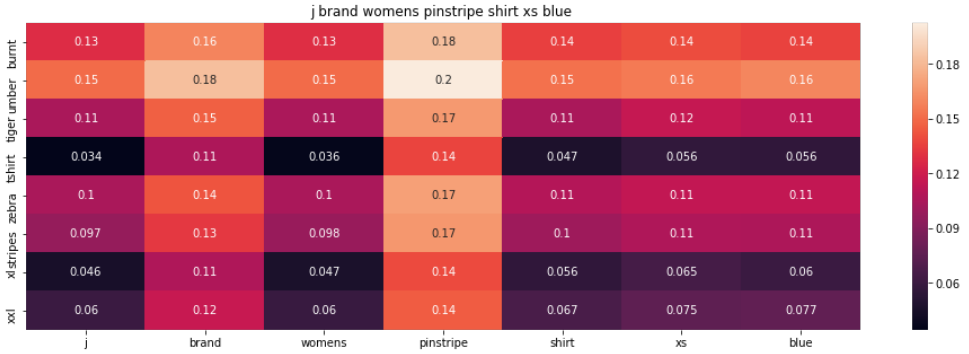
ASIN : B074MXY984
Brand : We The Free
euclidean distance from input : 33.057348696887495

Product Title: karen scott wmen short sleeve henley top plus size 0x bright white
Euclidean Distance from input image: 33.057348696887495
Amazon Url: www.amzon.com/dp/B0196H7P5A
==========



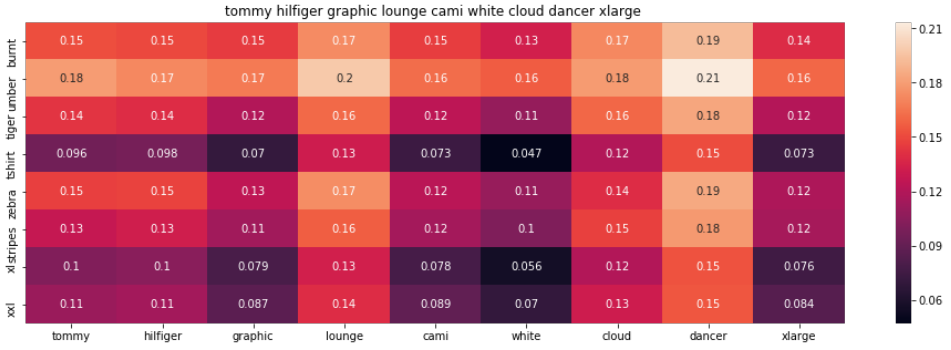
ASIN : B018H5AZXQ
Brand : Buffalo
euclidean distance from input : 33.207708593269814

Product Title: free people scoopneck highlow thermal tunic oatmeal heather
Euclidean Distance from input image: 33.207708593269814
Amazon Url: www.amzon.com/dp/B00ZH2X45E
==========



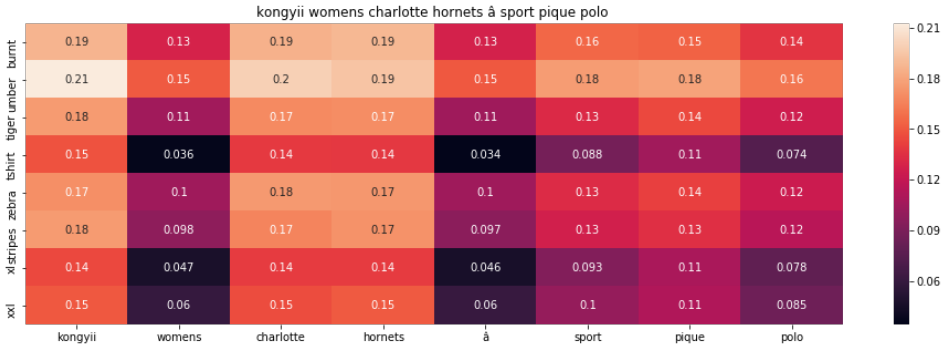
ASIN : B06XYP1X1F
Brand : J Brand Jeans
euclidean distance from input : 33.25903863363772

Product Title: masion jules longsleeve paneled sweatshirt xxl grey
Euclidean Distance from input image: 33.25903863363772
Amazon Url: www.amzon.com/dp/B06X16WTFX
==========



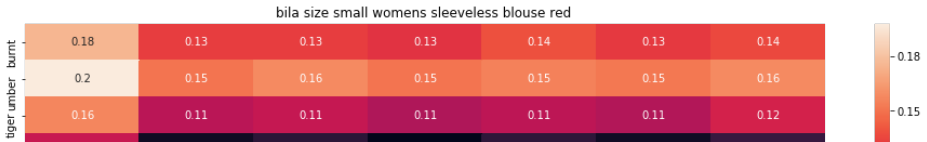
ASIN : B01BMSFYW2
Brand : igertommy hilf
euclidean distance from input : 33.37870734784315

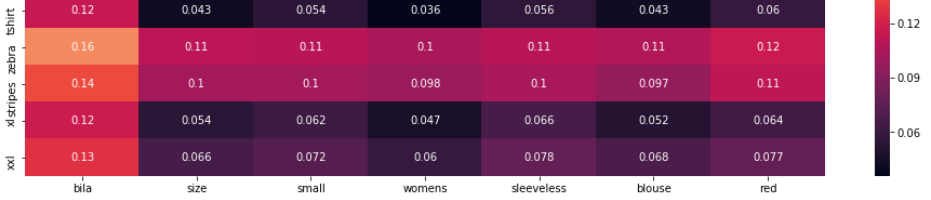
Product Title: vitamina usa soft vneckline belted blouse st6356 wht
Euclidean Distance from input image: 33.37870734784315
Amazon Url: www.amzon.com/dp/B0155ICSDS
==========

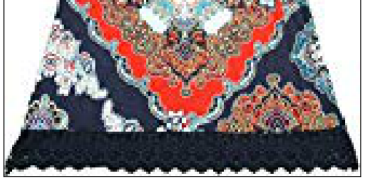


ASIN : B01FJVZST2
Brand : KONGYII
euclidean distance from input : 34.1639189050993

Product Title: dg2 diane gilman turtleneck fleece pullover pockets small gray
Euclidean Distance from input image: 34.1639189050993
Amazon Url: www.amzon.com/dp/B01MSP0KAM
==========





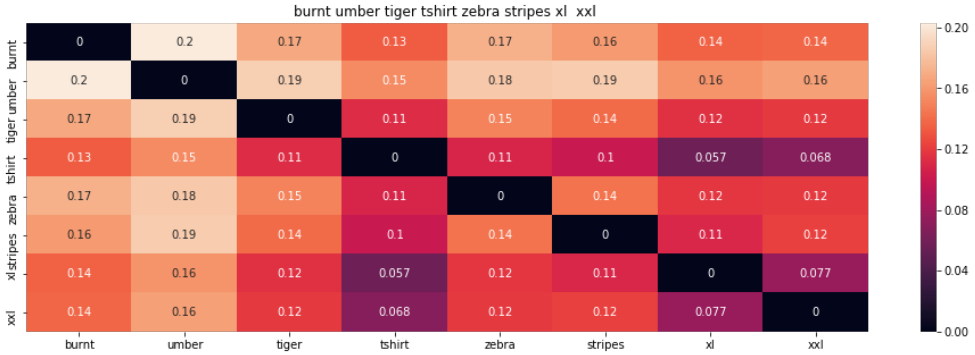


ASIN : B01L7ROZNC
Brand : Bila
euclidean distance from input : 34.4207126613856

Product Title: jessica simpson womens frida peasant top antiquewhite x small
Euclidean Distance from input image: 34.4207126613856
Amazon Url: www.amzon.com/dp/B01KW297J0
==========

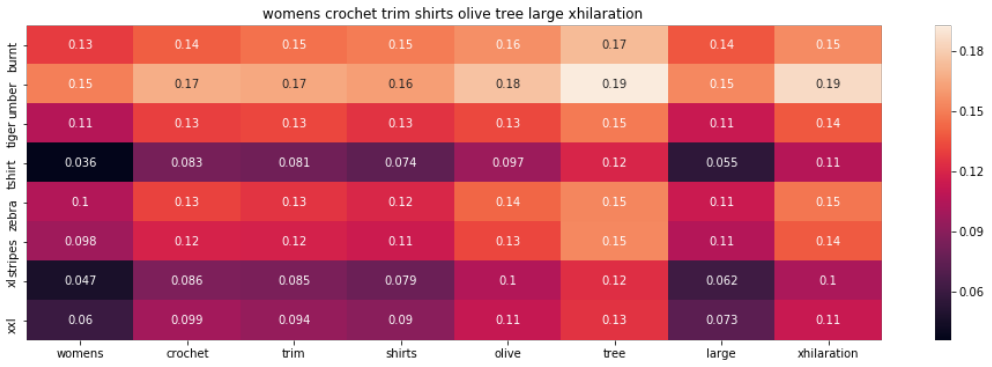
In [57]:

```
# More imp to 'idf' and 'Image' and less importance to '[brand + type(shirt/top) + color]'

idf_exfeat_image(12566, 50, 5, 50, 10)
```



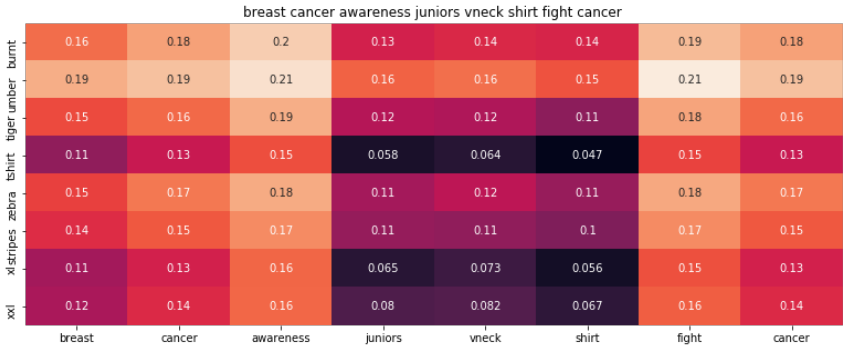
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 3.575827992920365e-06

Product Title: foxcroft nyc womens pinpoint oxford shirt noniron stretch poplin blouse xlarge white
Euclidean Distance from input image: 3.575827992920365e-06
Amazon Url: www.amzon.com/dp/B072277HVB
==========



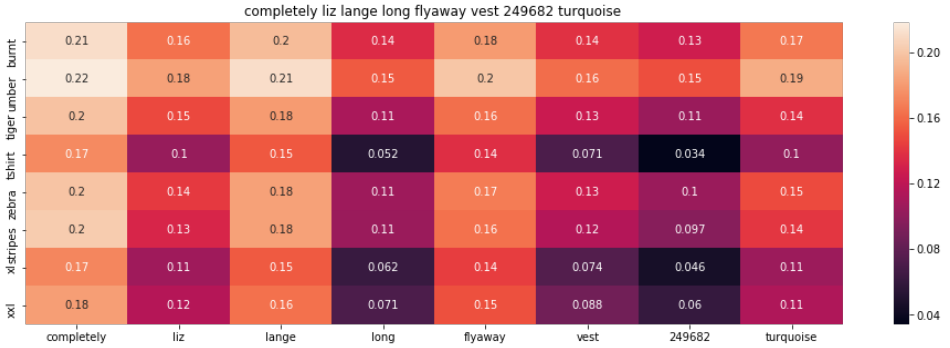
ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 17.862960107020328

Product Title: kiind longsleeve swing top white
Euclidean Distance from input image: 17.862960107020328
Amazon Url: www.amzon.com/dp/B0142Q3C6G
==========



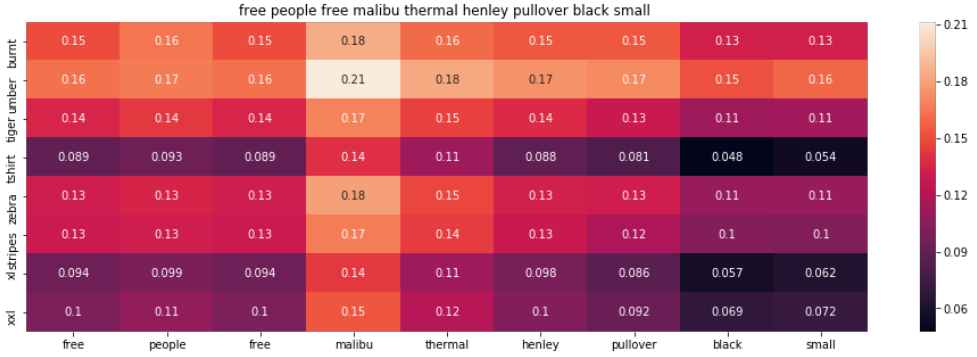
ASIN : B016CU40IY
Brand : Juiceclouds
euclidean distance from input : 18.396175590095105

Product Title: size comfortblend shirred crewneck longsleeve tshirt white 2x
Euclidean Distance from input image: 18.396175590095105
Amazon Url: www.amzon.com/dp/B01N5YHV9R
==========



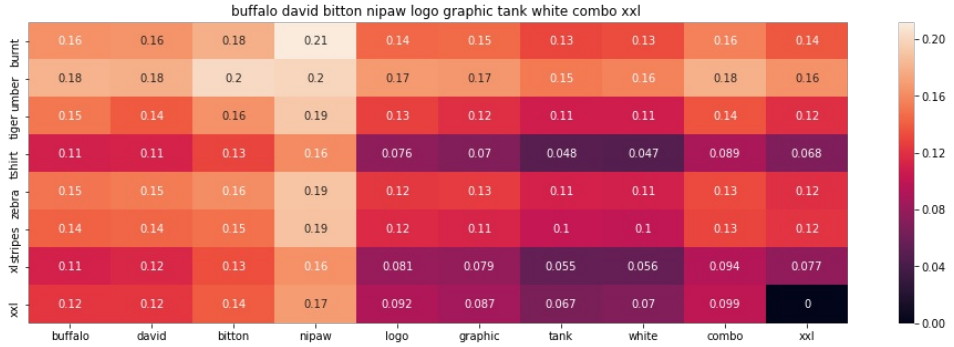
ASIN : B074LTBWSW
Brand : Liz Lange
euclidean distance from input : 18.903364190635585

Product Title: karen scott womens plus short sleeve solid henley top white 1x
Euclidean Distance from input image: 18.903364190635585
Amazon Url: www.amzon.com/dp/B0196H3Z2W
==========



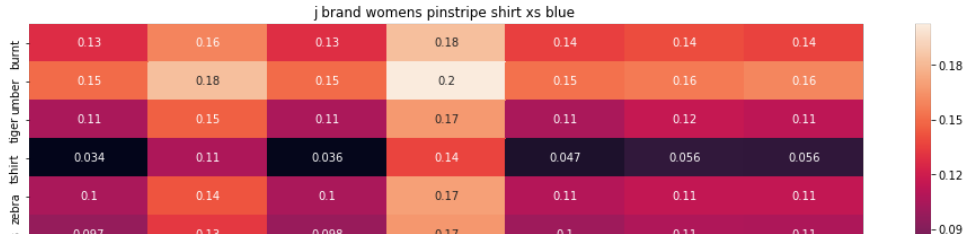
ASIN : B074MXY984
Brand : We The Free
euclidean distance from input : 18.90972065017337

Product Title: karen scott wmen short sleeve henley top plus size 0x bright white
Euclidean Distance from input image: 18.90972065017337
Amazon Url: www.amzon.com/dp/B0196H7P5A
==========



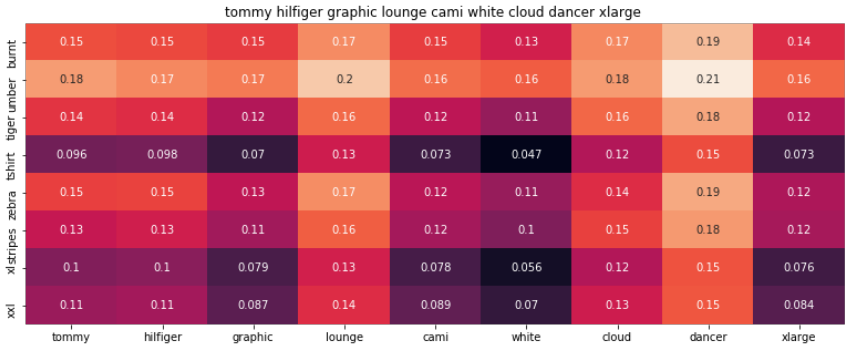
ASIN : B018H5AZXQ
Brand : Buffalo
euclidean distance from input : 18.994380001829143

Product Title: free people scoopneck highlow thermal tunic oatmeal heather
Euclidean Distance from input image: 18.994380001829143
Amazon Url: www.amzon.com/dp/B00ZH2X45E
==========



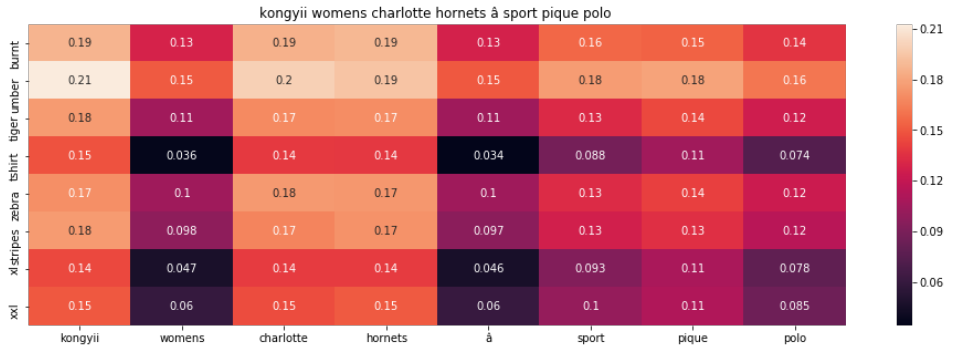
ASIN : B06XYP1X1F
Brand : J Brand Jeans
euclidean distance from input : 19.022364042363336

Product Title: masion jules longsleeve paneled sweatshirt xxl grey
Euclidean Distance from input image: 19.022364042363336
Amazon Url: www.amzon.com/dp/B06X16WTFX
==========



ASIN : B01BMSFYW2
Brand : igertommy hilf
euclidean distance from input : 19.093074914831828

Product Title: vitamina usa soft vneckline belted blouse st6356 wht
Euclidean Distance from input image: 19.093074914831828
Amazon Url: www.amzon.com/dp/B0155ICSDS
==========



ASIN : B01FJVZST2
Brand : KONGYII
euclidean distance from input : 19.541810173159593

Product Title: dg2 diane gilman turtleneck fleece pullover pockets small gray
Euclidean Distance from input image: 19.541810173159593
Amazon Url: www.amzon.com/dp/B01MSP0KAM
==========

ASIN : B01L7ROZNC
Brand : Bila
euclidean distance from input : 19.685413307314867

Product Title: jessica simpson womens frida peasant top antiquewhite x small
Euclidean Distance from input image: 19.685413307314867
Amazon Url: www.amzon.com/dp/B01KW297J0
==========

Conclusions

- When I gave more imp to 'idf' (50) and '[brand + type(shirt/top) + color]' (50) and less importance to 'Image' (5), the output seems better than other wieghtages.

In [ ]:

```
```