

Amazon Apparel Recommendations

[4.2] Data and Code:

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

[4.3] Overview of the data

In [1]:

```
#import all the necessary packages.

from PIL import Image
import requests
from io import BytesIO
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import warnings
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
import math
import time
import re
import os
import seaborn as sns
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import pairwise_distances
from matplotlib import gridspec
from scipy.sparse import hstack
import plotly
import plotly.figure_factory as ff
from plotly.graph_objs import Scatter, Layout

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

In [2]:

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

In [123]:

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

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

Terminology:

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

....

In [124]:

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

Out[124]:

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

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

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

In [125]:

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

In [126]:

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

Number of data points : 183138 Number of features: 7

Out[126]:

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

[5.1] Missing data for various features.

Basic stats for the feature: product_type_name

In [127]:

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

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

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

In [128]:

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

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

In [140]:

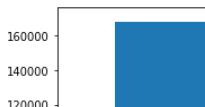
```
# find the 10 most frequent product_type_names.
product_type_count = Counter(list(data['product_type_name']))
product_type_count.most_common(10)
```

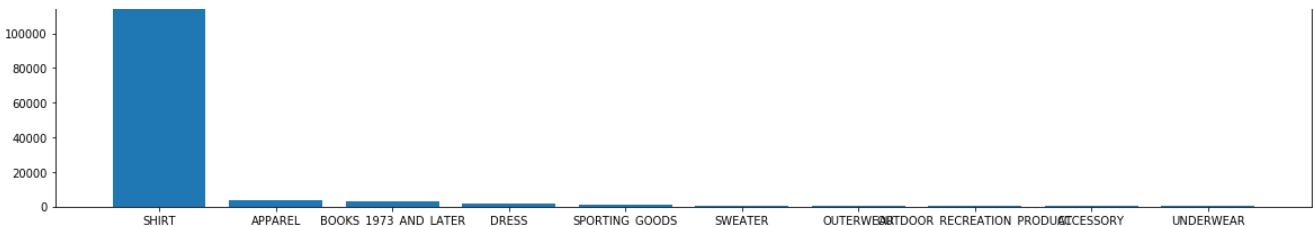
Out[140]:

```
[('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)]
```

In [139]:

```
plt.figure(figsize=(20,5))
x=[]
y=[]
for a ,b in product_type_count.most_common(10):
    x.append(a)
    y.append(b)
plt.bar(x,y)
plt.show()
```



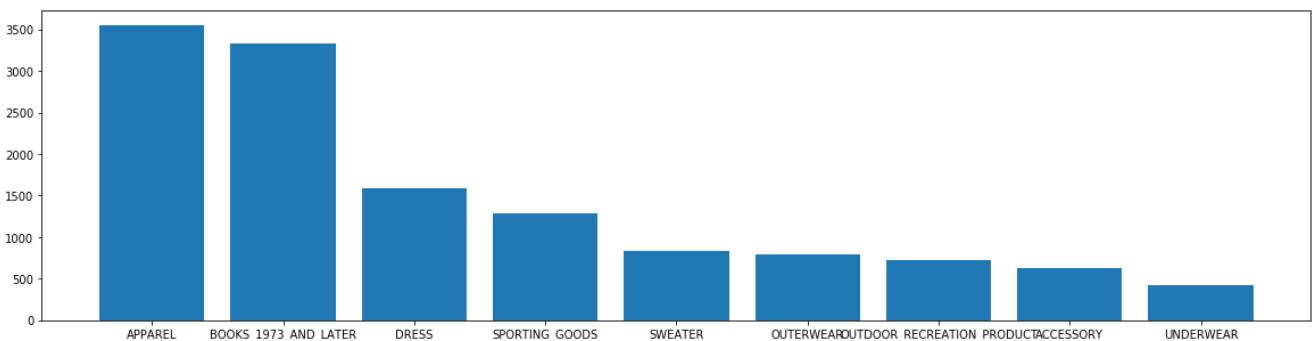


In [141]:

```
x.remove('SHIRT')
y.remove(167794)
```

In [143]:

```
plt.figure(figsize=(20,5))
plt.bar(x,y)
plt.show()
```



Basic stats for the feature: brand

In [144]:

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

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

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

In [145]:

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

Out[145]:

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

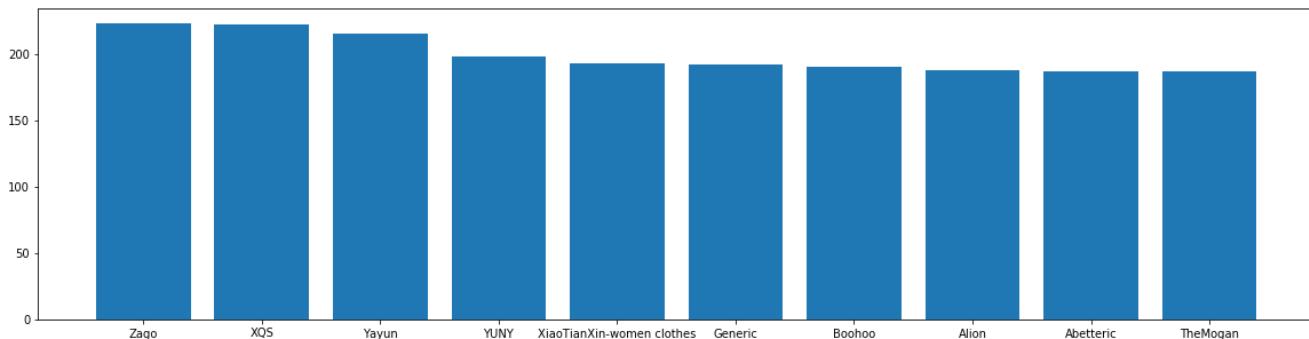
In [147]:

```
plt.figure(figsize=(20,5))
```

```

plt.figure(figsize=(20, 5))
x=[]
y=[]
for a ,b in brand_count.most_common(10):
    x.append(a)
    y.append(b)
plt.bar(x,y)
plt.show()

```



Basic stats for the feature: color

In [148]:

```

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

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

```

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

In [149]:

```

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

```

Out[149]:

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

Basic stats for the feature: formatted_price

In [150]:

```

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

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

```

count 28395
unique 3135
top \$19.99
freq 945

```
freq      77  
Name: formatted_price, dtype: object
```

In [151]:

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

Out[151]:

```
[(None, 154743),  
 ('$19.99', 945),  
 ('$9.99', 749),  
 ('$9.50', 601),  
 ('$14.99', 472),  
 ('$7.50', 463),  
 ('$24.99', 414),  
 ('$29.99', 370),  
 ('$8.99', 343),  
 ('$9.01', 336)]
```

Basic stats for the feature: title

In [16]:

```
print(data['title'].describe())  
  
# All of the products have a title.  
# Titles are fairly descriptive of what the product is.  
# We use titles extensively in this workshop  
# as they are short and informative.
```

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

In [17]:

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

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

In [18]:

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

```
Number of data points After eliminating price=NULL : 28395
```

In [19]:

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

```
Number of data points After eliminating color=NULL : 28385
```

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

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

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

In [20]:

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

In [21]:

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

```
'''  
from PIL import Image  
import requests  
from io import BytesIO  
  
for index, row in images.iterrows():  
    url = row['large_image_url']  
    response = requests.get(url)  
    img = Image.open(BytesIO(response.content))  
    img.save('images/28k_images/'+row['asin']+'.jpeg')  

```

Out[21]:

```
"\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"
```

[5.2] Remove near duplicate items

[5.2.1] Understand about duplicates.

In [17]:

```
# read data from pickle file from previous stage  
data = pd.read_pickle('pickels/28k_apparel_data')  
  
# find number of products that have duplicate titles.  
print(sum(data.duplicated('title')))  
# we have 2325 products which have same title but different color
```

2325

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

:B00AQ4GMCK	:B00AQ4GMTS
:B00AQ4GMLQ	:B00AQ4GN3I

These shirts exactly same except in color

:B00G278GZ6	:B00G278W6O
:B00G278Z2A	:B00G2786X8

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

[5.2.2] Remove duplicates : Part 1

In [18]:

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

In [19]:

```
data.head()
```

Out[19]:

	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 Olympi...	\$9.99
11	B001LOUGE4	Fitness Etc.	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	Ladies Cotton Tank 2x1 Ribbed Tank Top	\$11.99
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	FeatherLite Ladies' Moisture Free Mesh Sport S...	\$20.54
21	B014ICEDNA	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	Supernatural Chibis Sam Dean And Castiel Short...	\$7.50

In [20]:

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

After removal of products with short description: 27949

In [21]:

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

Out[21]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
61973	B06Y1KZ2WB	Éclair	Black/Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	Éclair Women's Printed Thin Strap Blouse Black...	\$24.99
133820	B010RV33VE	xiaoming	Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Womens Sleeveless Loose Long T-shirts...	\$18.19

	asin	brand	color	medium_image_url	product_type_name	xiaoming	title	formatted_price
81461	B01DDSDLNS	xiaoming	White	images-amazon.com/images...	SHIRT	xiaoming	Women's White Long Sleeve Single Brea...	\$21.58
75995	B00X5LYO9Y	xiaoming	Red Anchors	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming	Stripes Tank Patch/Bear Sleeve Anchor...	\$15.91
151570	B00WPJG35K	xiaoming	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming	Sleeve Sheer Loose Tassel Kimono Woma...	\$14.32

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

Titles 1:

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

Title 2:

- 25. tokidoki The Queen of Diamonds Women's Shirt X-Large
- 26. tokidoki The Queen of Diamonds Women's Shirt Small
- 27. tokidoki The Queen of Diamonds Women's Shirt Large

Title 3:

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

In [22]:

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

In [28]:

```
import itertools
stage1_deduplicate_asins = []
i = 0
j = 0
num_data_points = data_sorted.shape[0]
while i < num_data_points and j < num_data_points:

    previous_i = i

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

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

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

```

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

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

# itertools.zip_longest(a,b): will map the corresponding words in both strings, it will
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 t
    hose two apperals are different
    # if the number of words in which both strings differ are < 2 , we are considering it as t
    hose two apperals are same, hence we are ignoring them
    if (length - count) > 2: # number of words in which both sensences differ
        # if both strings are differ by more than 2 words we include the 1st string index
        stage1_dedupe_asins.append(data_sorted['asin'].loc[indices[i]])

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

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

```

In [29]:

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

We removed the dupliactes which differ only at the end.

In [30]:

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

Number of data points : 17593

In [31]:

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

[5.2.3] Remove duplicates : Part 2

In the previous cell, we sorted whole data in alphabetical order of titles. Then, we removed titles which are adjacent and very similar title

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

Examples:

Titles-1

86261. UltraClub Women's Classic Wrinkle-Free Long Sleeve Oxford Shirt, Pink, XX-Large
115042. UltraClub Ladies Classic Wrinkle-Free Long-Sleeve Oxford Light Blue XXL

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

In [32]:

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

In [33]:

```
# This code snippet takes significant amount of time.  
# O(n^2) time.  
# Takes about an hour to run on a decent computer.  
  
indices = []  
for i, row in data.iterrows():  
    indices.append(i)  
  
stage2_deduplicate_asins = []  
while len(indices) != 0:  
    i = indices.pop()  
    stage2_deduplicate_asins.append(data['asin'].loc[i])  
    # consider the first apparel's title  
    a = data['title'].loc[i].split()  
    # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'X-Large']  
    for j in indices:  
  
        b = data['title'].loc[j].split()  
        # store the list of words of jth string in b, ex: b = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'X-Large']  
  
        length = max(len(a), len(b))  
  
        # count is used to store the number of words that are matched in both strings  
        count = 0  
  
        # itertools.zip_longest(a,b): will map the corresponding words in both strings, it will  
        appended None in case of unequal strings  
        # example: a = ['a', 'b', 'c', 'd']  
        # b = ['a', 'b', 'd']  
        # itertools.zip_longest(a,b): will give [('a', 'a'), ('b', 'b'), ('c', 'd'), ('d', None)]  
        for k in itertools.zip_longest(a, b):  
            if (k[0] == k[1]):  
                count += 1  
  
        # if the number of words in which both strings differ are < 3 , we are considering it as those two apperals are same, hence we are ignoring them  
        if (length - count) < 3:  
            indices.remove(j)
```

In [34]:

```
# from whole previous products we will consider only  
# the products that are found in previous cell  
data = data.loc[data['asin'].isin(stage2_deduplicate_asins)]
```

In [35]:

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

Number of data points after stage two of dedupe: 16435

In [36]:

```
data.to_pickle('pickels/16k_apperal_data')  
# Storing these products in a pickle file  
# candidates who wants to download these files instead
```

```
# of 180K they can download and use them from the Google Drive folder.
```

6. Text pre-processing

In [23]:

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

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

In [5]:

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

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

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

In [6]:

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

4.941327991814034 seconds

In [7]:

```
data.head()
```

Out[7]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
	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 [8]:

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

Stemming

In [9]:

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

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

argu
fish

[8] Text based product similarity

In [10]:

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

Out[10]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
	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

	asin	brand	color	https://images-na.ssl-images-amazon.com/images...	product_type_name	supernatural title	formatted_price
27	B014ICEJ1Q	FNC7C	Purple	images-amazon.com/images...	SHIRT	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 [29]:

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

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

#plotting code to understand the algorithm's decision.
def plot_heatmap(keys, values, labels, url, text):
    # keys: list of words of recommended title
    # values: len(values) == len(keys), values(i) represents the occurrence of the word
    # 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))
    # 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, plotting heat map that represents the count of commonly occurred words in title2
    ax = plt.subplot(gs[0])
    # it displays a cell in white color if the word is intersection(lis of words of title1 and
    # list of words of title2), in black if not
    ax = sns.heatmap(np.array([values]), annot=np.array([labels]))
    ax.set_xticklabels(keys) # set that axis labels as the words of title
    ax.set_title(text) # apparel title

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

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

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

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

    # doc_id : index of the title1
    # vec1 : input apparels's vector, it is of a dict type {word:count}
    # vec2 : recommended apparels's vector, it is of a dict type {word:count}
    # url : apparels image url
    # text: title of recomonded apparel (used to keep title of image)
    # model, it can be any of the models,
    # 1. bag_of_words
    # 2. tfidf
    # 3. idf

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

```

```

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

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

# labels: len(labels) == len(keys), the values of labels depends on the model we are using
# if model == 'bag of words': labels(i) = values(i)
# if model == 'tfidf weighted bag of words':labels(i) = tfidf(keys(i))
# if model == 'idf weighted bag of words':labels(i) = idf(keys(i))

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

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

# this function gets a list of words along with the frequency of each
# word given "text"
def text_to_vector(text):
    word = re.compile(r'\w+')
    words = word.findall(text)
    # words stores list of all words in given string, you can try 'words = text.split()' this will
    also gives same result
    return Counter(words) # Counter counts the occurrence of each word in list, it returns dict
type object {word1:count}

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

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

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

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

```

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

In [12]:

```

from sklearn.feature_extraction.text import CountVectorizer
title_vectorizer = CountVectorizer()
title_features = title_vectorizer.fit_transform(data['title'])

```

```

title_features.get_shape() # get number of rows and columns in feature matrix.
# title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(corpus) returns
# the a sparse matrix of dimensions #data_points * #words_in_corpus

# What is a sparse vector?

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

```

Out[12]:

(16435, 12684)

In [13]:

```

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

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

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

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

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

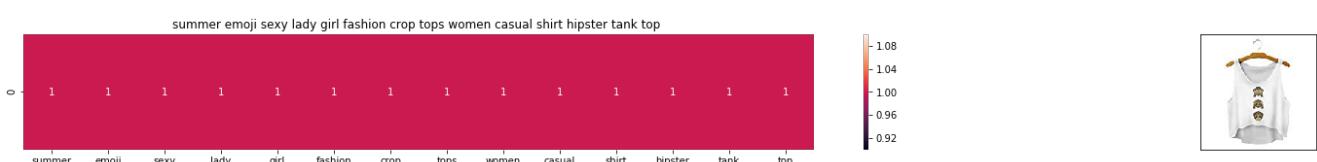
    #call the bag-of-words model for a product to get similar products.
    bag_of_words_model(12566, 20) # change the index if you want to.
    # In the output heat map each value represents the count value
    # of the label word, the color represents the intersection
    # with inputs title.

#try 12566
#try 931

```

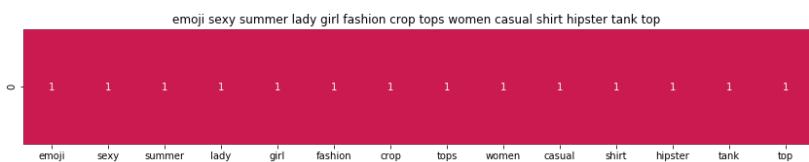


ASIN : B010V3B44G
 Brand: Doxi Supermall
 Title: fashion crop tops women casual summer emoji sexy lady girl shirt hipster tank top
 Euclidean similarity with the query image : 0.0
 =====



ASIN : B010V3BDII
 Brand: Doxi Supermall
 Title: summer emoji sexy lady girl fashion crop tops women casual shirt hipster tank top

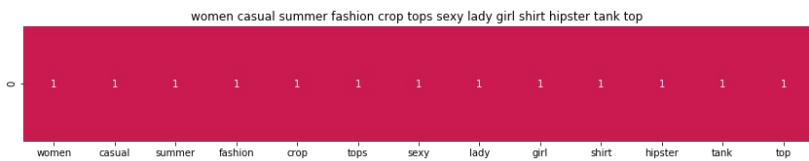
Euclidean similarity with the query image : 0.0



ASIN : B010V3BLWQ

Brand: Doxi Supermall

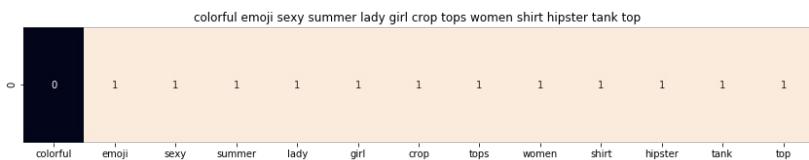
Title: emoji sexy summer lady girl fashion crop tops women casual shirt hipster tank top
Euclidean similarity with the query image : 0.0



ASIN : B010V3AYSS

Brand: Doxi Supermall

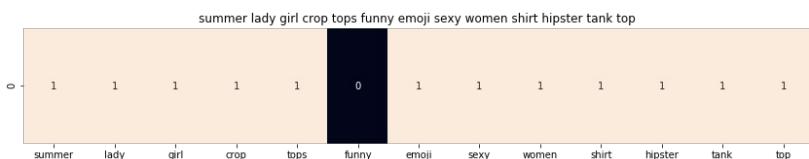
Title: women casual summer fashion crop tops sexy lady girl shirt hipster tank top
Euclidean similarity with the query image : 1.0



ASIN : B010V3BQZS

Brand: Doxi Supermall

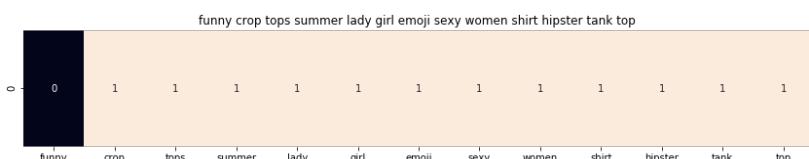
Title: colorful emoji sexy summer lady girl crop tops women shirt hipster tank top
Euclidean similarity with the query image : 1.7320508075688772



ASIN : B010V3BVMQ

Brand: Doxi Supermall

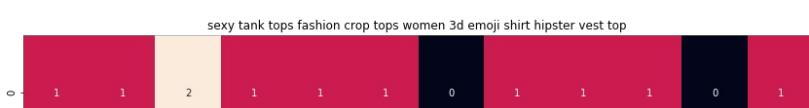
Title: summer lady girl crop tops funny emoji sexy women shirt hipster tank top
Euclidean similarity with the query image : 1.7320508075688772



ASIN : B010V3C116

Brand: Doxi Supermall

Title: funny crop tops summer lady girl emoji sexy women shirt hipster tank top
Euclidean similarity with the query image : 1.7320508075688772



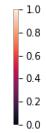
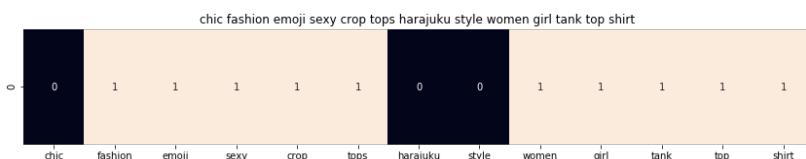


ASIN : B010V3DB9C

Brand: Doxi Supermall

Title: sexy tank tops fashion crop tops women 3d emoji shirt hipster vest top

Euclidean similarity with the query image : 2.6457513110645907

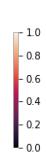
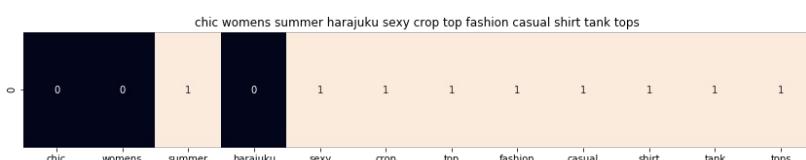


ASIN : B011RCJPR8

Brand: Chiclook Cool

Title: chic fashion emoji sexy crop tops harajuku style women girl tank top shirt

Euclidean similarity with the query image : 2.6457513110645907

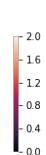
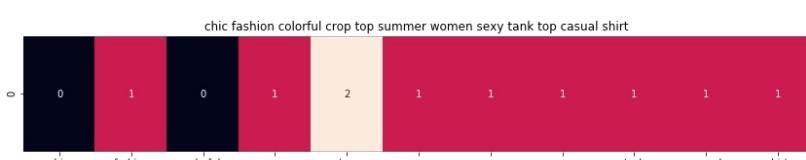


ASIN : B011RCJEMO

Brand: Chiclook Cool

Title: chic womens summer harajuku sexy crop top fashion casual shirt tank tops

Euclidean similarity with the query image : 2.8284271247461903

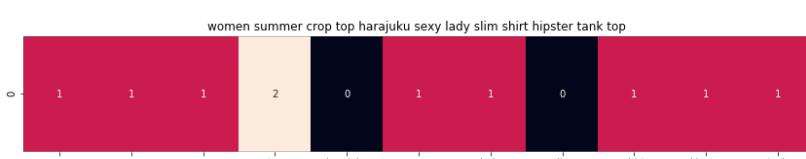


ASIN : B011RCJ6UE

Brand: Chiclook Cool

Title: chic fashion colorful crop top summer women sexy tank top casual shirt

Euclidean similarity with the query image : 2.8284271247461903

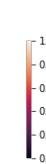
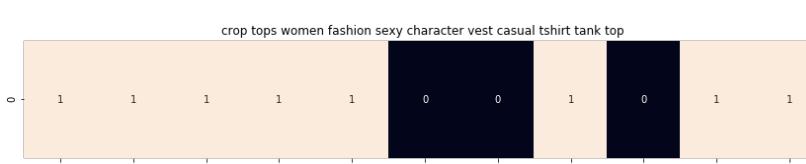


ASIN : B010V3EDEE

Brand: Doxi Supermall

Title: women summer crop top harajuku sexy lady slim shirt hipster tank top

Euclidean similarity with the query image : 2.8284271247461903

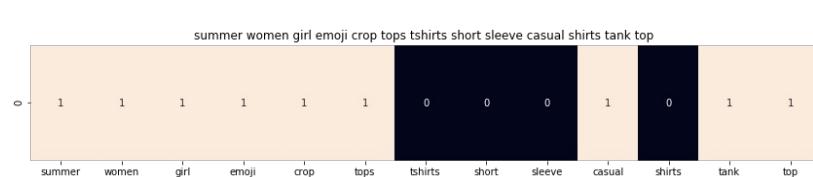


ASIN : B0107UEPVM

Brand: Mang GO

Title: crop tops women fashion sexy character vest casual tshirt tank top

Euclidean similarity with the query image : 3.0

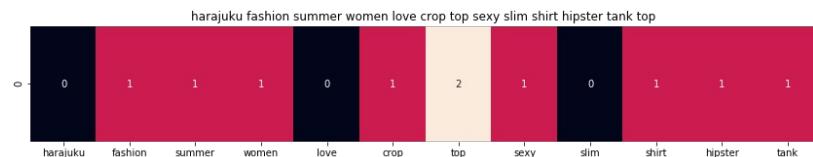


ASIN : B0124ECIU4

Brand: Doxi Supermall

Title: summer women girl emoji crop tops tshirts short sleeve casual shirts tank top

Euclidean similarity with the query image : 3.0

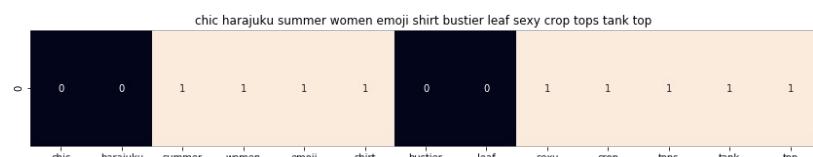


ASIN : B010V350BU

Brand: Doxi Supermall

Title: harajuku fashion summer women love crop top sexy slim shirt hipster tank top

Euclidean similarity with the query image : 3.0

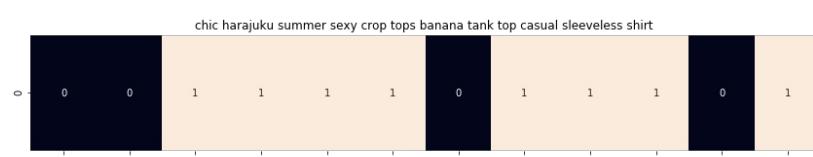


ASIN : B011UEXGH8

Brand: Chiclook Cool

Title: chic harajuku summer women emoji shirt bustier leaf sexy crop tops tank top

Euclidean similarity with the query image : 3.0



ASIN : B011RCIQBE

Brand: Chiclook Cool

Title: chic harajuku summer sexy crop tops banana tank top casual sleeveless shirt

Euclidean similarity with the query image : 3.1622776601683795

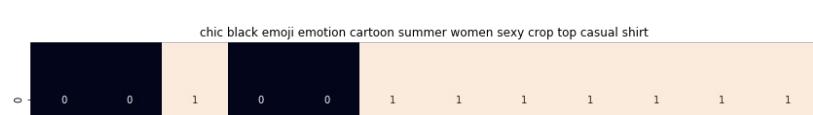


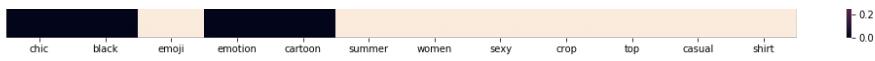
ASIN : B0124E80M4

Brand: Doxi Supermall

Title: black emoji crop top fashion summer women tank top hipster casual top

Euclidean similarity with the query image : 3.1622776601683795





ASIN : B011RCJ4M4

Brand: Chiclook Cool

Title: chic black emoji emotion cartoon summer women sexy crop top casual shirt

Euclidean similarity with the query image : 3.1622776601683795



ASIN : B010V39146

Brand: Doxi Supermall

Title: doxi women sexy crop top harajuku letter summer lady slim shirt hipster tank top

Euclidean similarity with the query image : 3.1622776601683795

[8.5] TF-IDF based product similarity

In [14]:

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

In [48]:

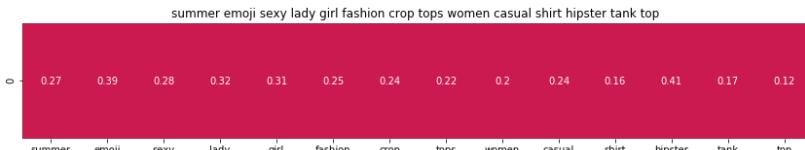
```
def tfidf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

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

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

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

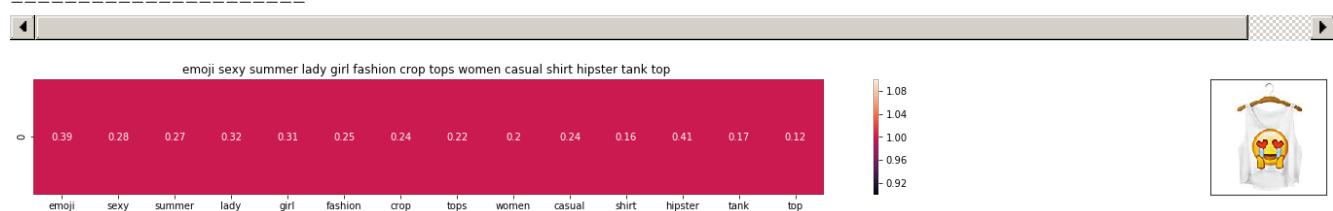
    for i in range(0,len(indices)):
        # we will pass 1. doc_id, 2. title1, 3. title2, url, model
        get_result(indices[i], data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]],
data['medium_image_url'].loc[df_indices[i]], 'tfidfd')
        print('ASIN :',data['asin'].loc[df_indices[i]])
        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 repre
sents the intersection with inputs title
```



ASIN : B010V3BDII

BRAND : Doxi Supermall

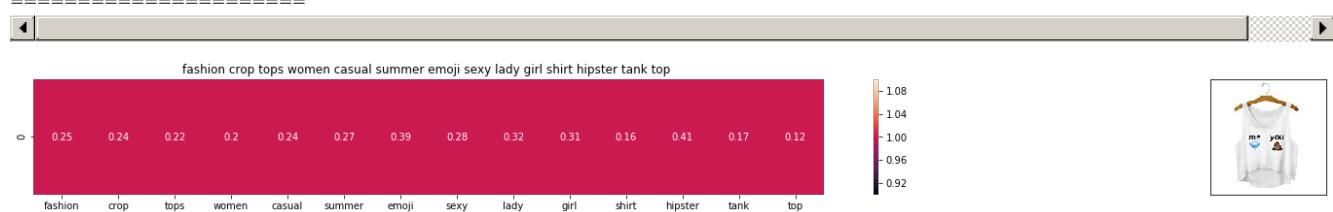
Euclidean distance from the given image : 0.0



ASIN : B010V3BLWQ

BRAND : Doxi Supermall

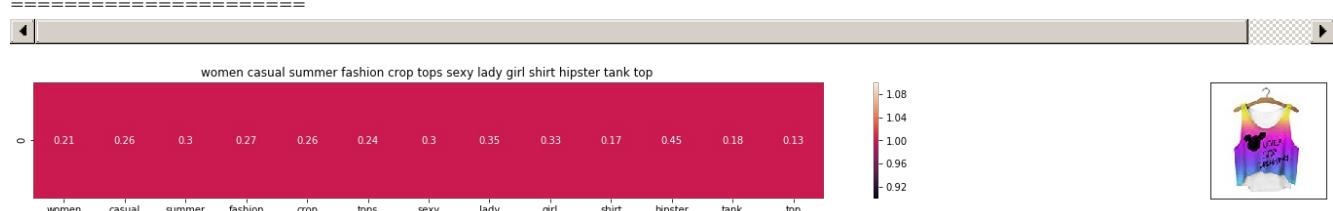
Euclidean distance from the given image : 0.0



ASIN : B010V3B44G

BRAND : Doxi Supermall

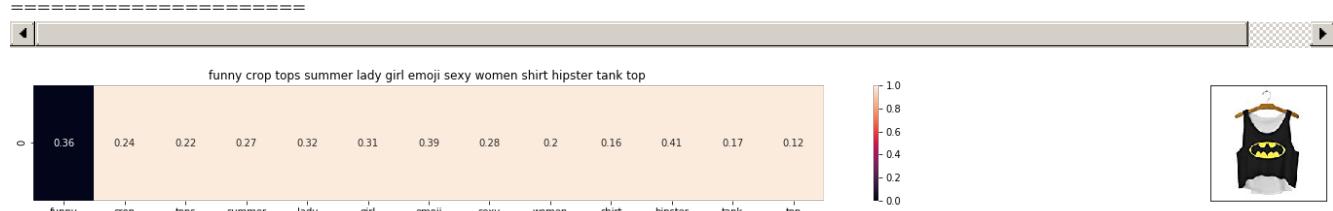
Euclidean distance from the given image : 0.0



ASIN : B010V3AYSS

BRAND : Doxi Supermall

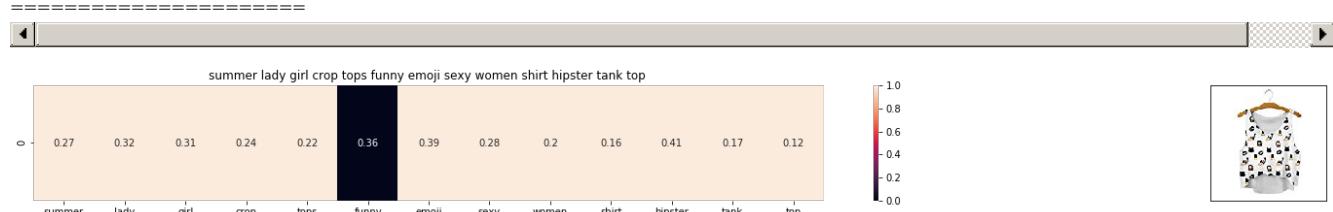
Euclidean distance from the given image : 0.40138594750234946



ASIN : B010V3C116

BRAND : Doxi Supermall

Euclidean distance from the given image : 0.4954421553895553

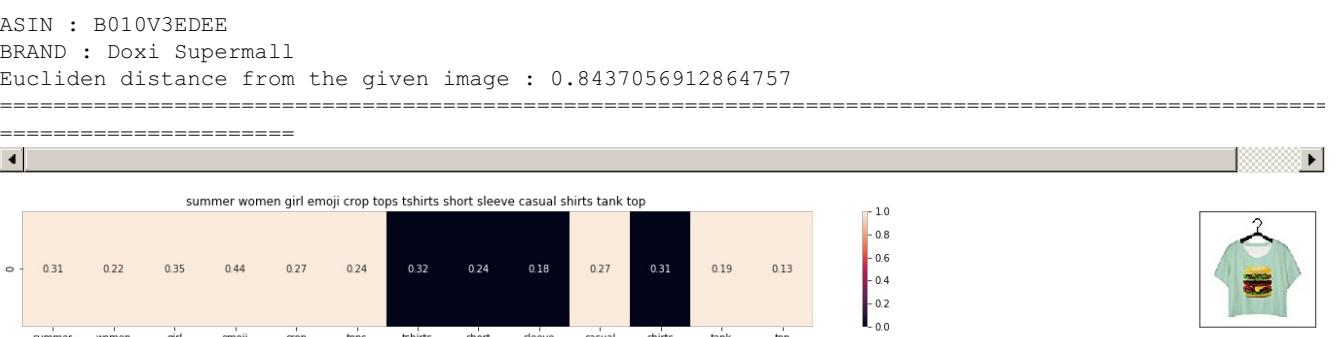
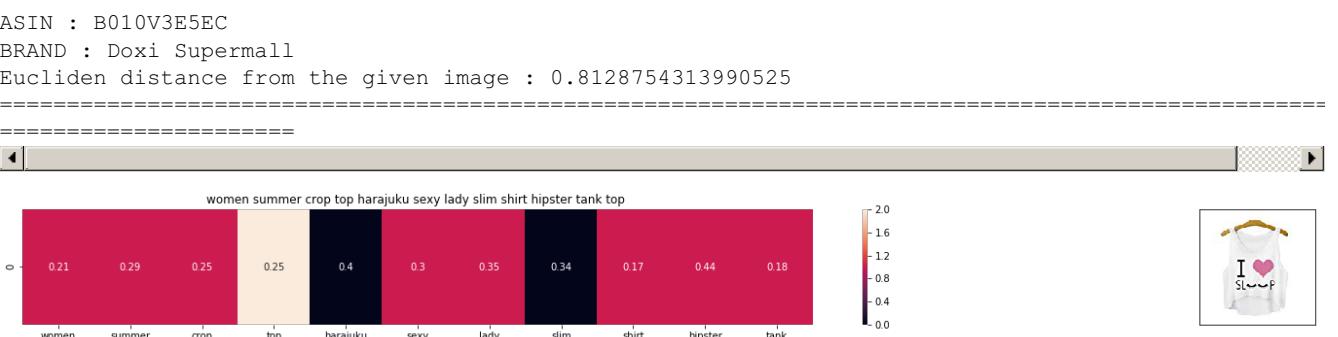
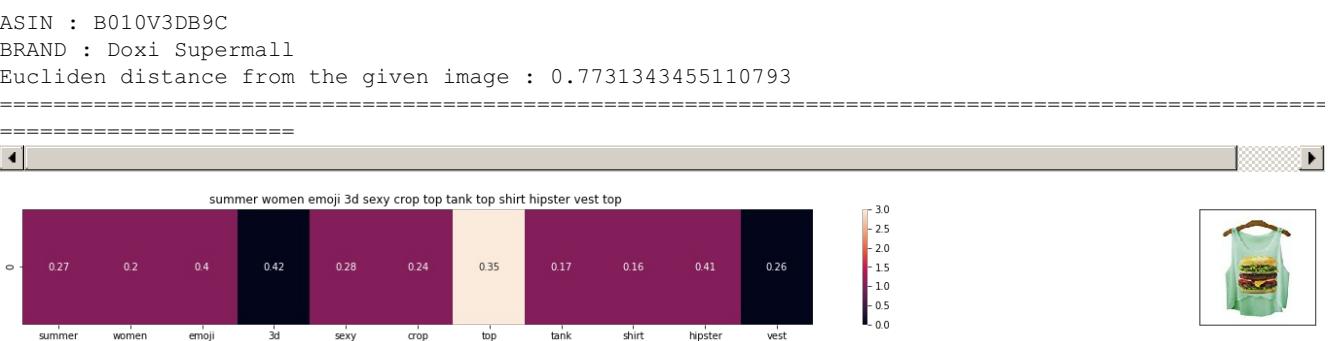
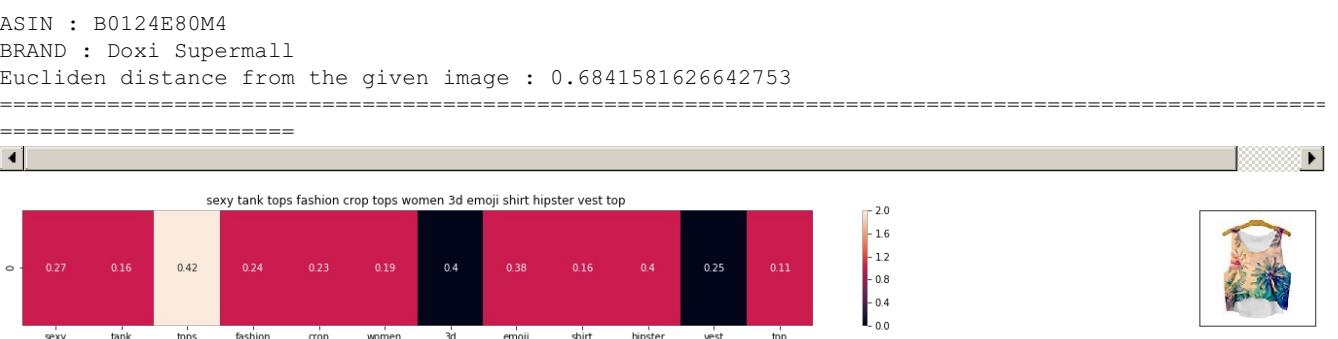
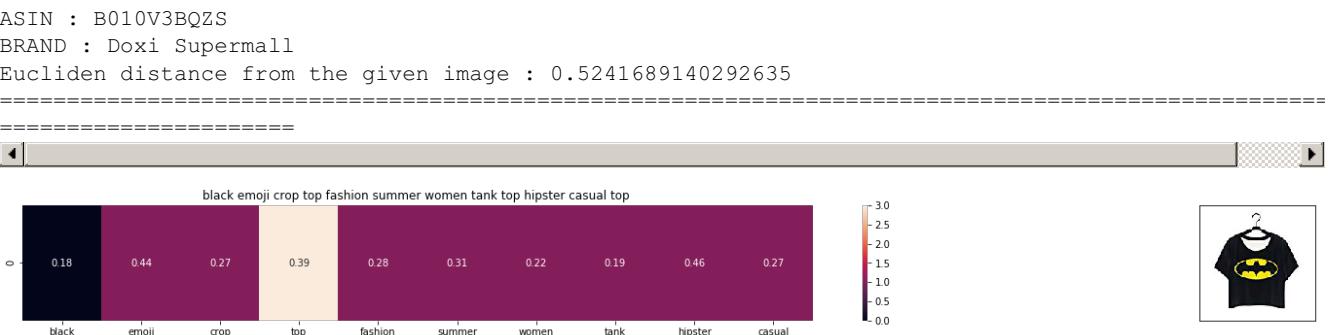
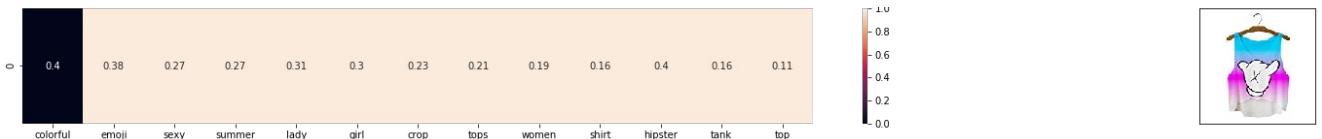


ASIN : B010V3BVMQ

BRAND : Doxi Supermall

Euclidean distance from the given image : 0.4954421553895553



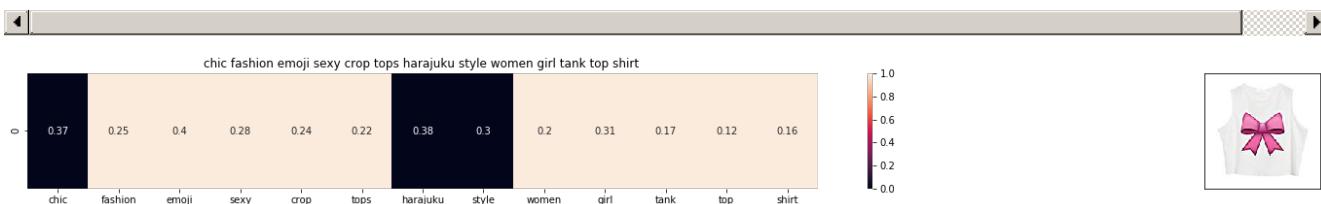


ASIN : B0124E80M4

LMM113 • LMM161 BURMAN

BRAND : Doxi Supermall

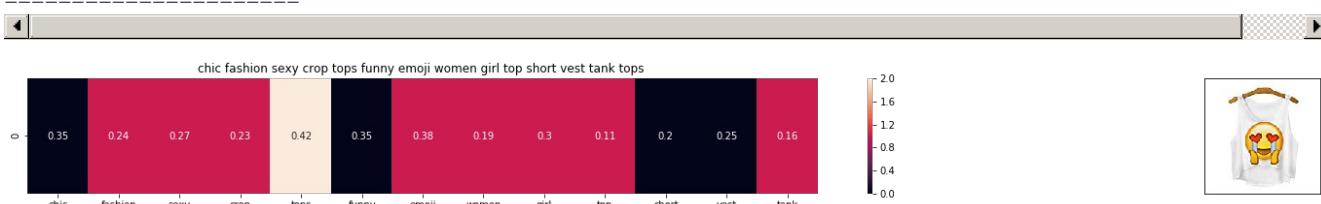
Euclidean distance from the given image : 0.8553483954246196



ASIN : B011RCJPR8

BRAND : Chiclook Cool

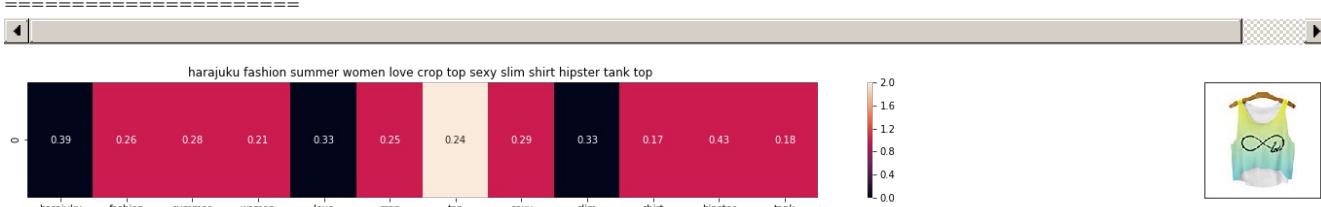
Euclidean distance from the given image : 0.8826321316686155



ASIN : B011RCJH58

BRAND : Chiclook Cool

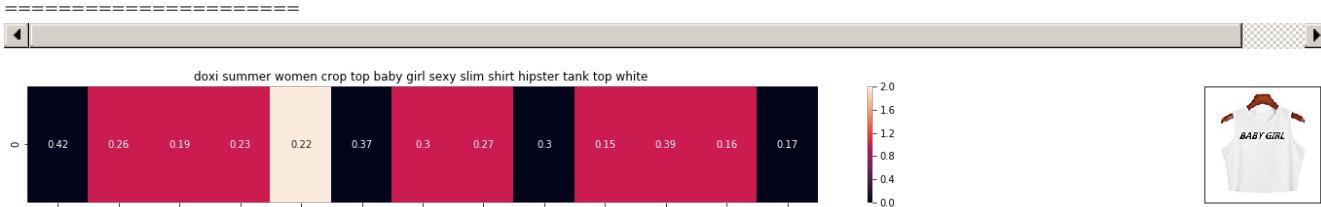
Euclidean distance from the given image : 0.900401746801386



ASTN : B010V350BII

BRAND : Doxi Supermall

Euclidean distance from the given image : 0.9172816572673459



ASTN : B010V3A23II

BRAND : Doxi Supermall

Euclidean distance from the given image : 0.9303848929561549



ACTN - P0124E7MHC

ASIN : B0124E/MHS

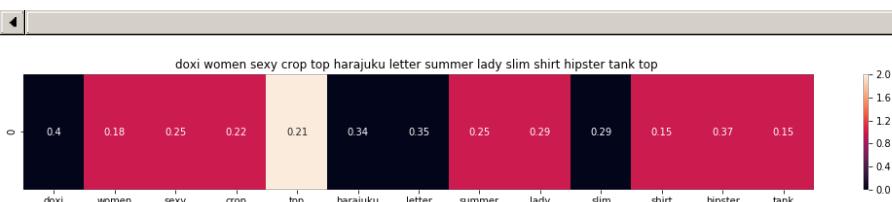
Euclidean distance from the given image : 0.0234421080080748



ASIN : B010V380LQ

BRAND : Doxi Supermall

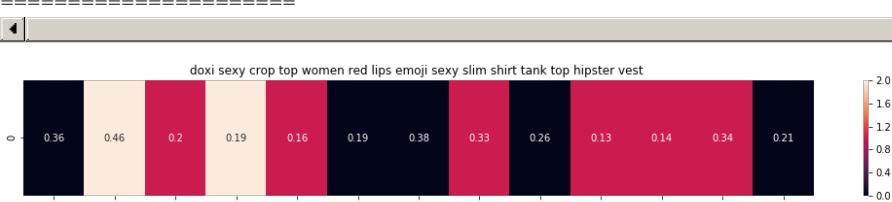
Eucliden distance from the given image : 0.9525344637195033



ASIN : B010V39146

BRAND : Doxi Supermall

Eucliden distance from the given image : 0.9525344637195033



ASIN : B010TKXEHG

BRAND : Doxi Supermall

Eucliden distance from the given image : 0.9560708174154958

[8.5] IDF based product similarity

In [49]:

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

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

In [50]:

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

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

In [51]:

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

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

    # to convert into idf title features we need to multiply the count values with the idf values of +
```

```

# 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 i present
for j in idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0]:
    # we replace the count values of word i in document j with idf_value of word i
    # idf_title_features[doc_id, index_of_word_in_corpus] = idf value of word
    idf_title_features[j,idf_title_vectorizer.vocabulary_[i]] = idf_val

```

In [52]:

```

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(idf_title_features,idf_title_features[doc_id])

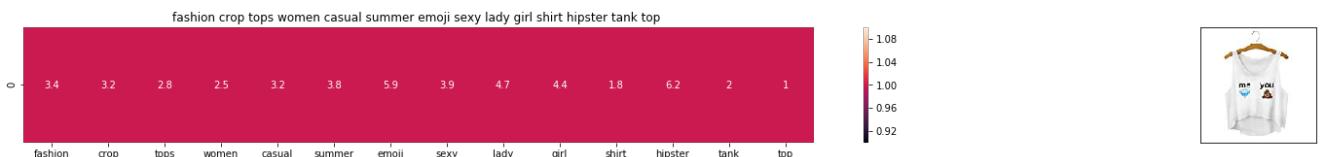
    # np.argsort will return indices of 9 smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists = np.sort(pairwise_dist.flatten())[0:num_results]

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

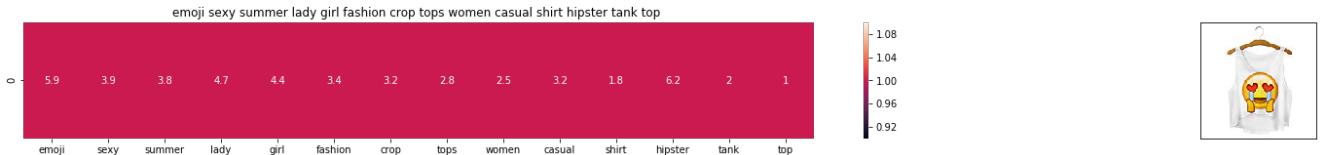
    for i in range(0,len(indices)):
        get_result(indices[i],data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]], 'idf')
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print('Brand :',data['brand'].loc[df_indices[i]])
    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

idf_model(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 : B010V3B44G
 Brand : Doxi Supermall
 euclidean distance from the given image : 0.0



ASIN : B010V3BLWQ
 Brand : Doxi Supermall
 euclidean distance from the given image : 0.0





0.92

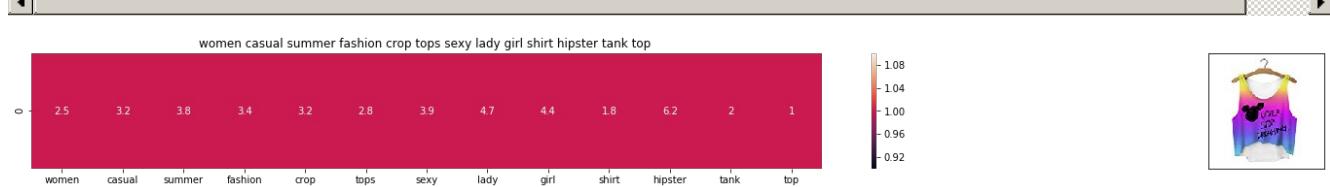


ASIN : B010V3BDII

Brand : Doxi Supermall

euclidean distance from the given image : 0.0

=====

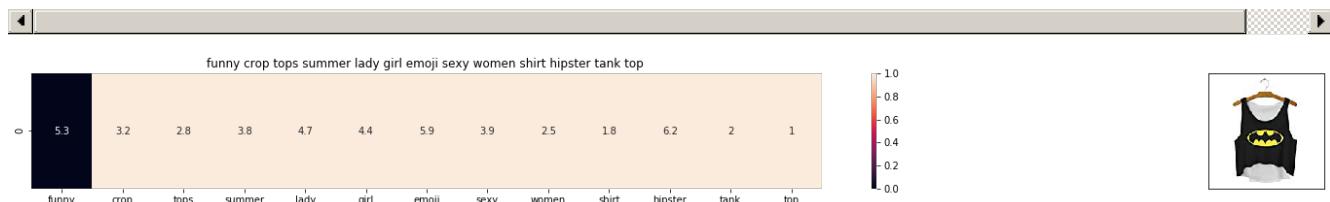


ASIN : B010V3AYSS

Brand : Doxi Supermall

euclidean distance from the given image : 5.922978852180059

=====

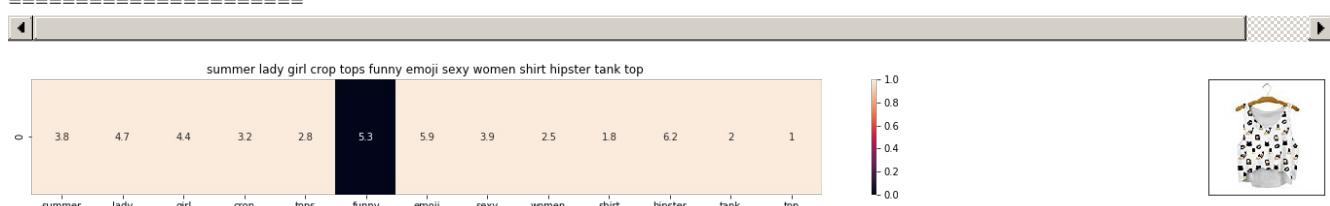


ASIN : B010V3C116

Brand : Doxi Supermall

euclidean distance from the given image : 7.037272185298332

=====

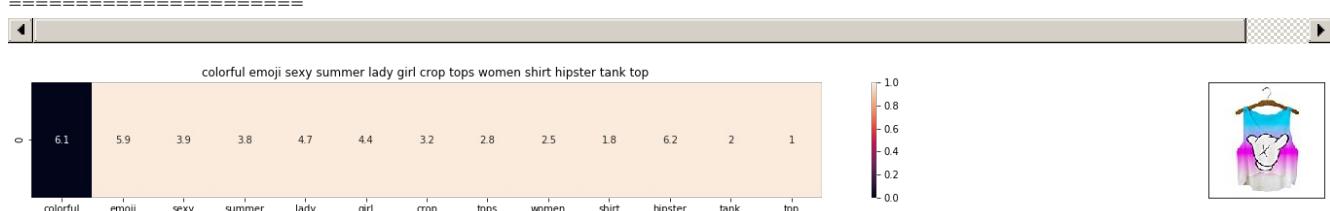


ASIN : B010V3BVMQ

Brand : Doxi Supermall

euclidean distance from the given image : 7.037272185298332

=====

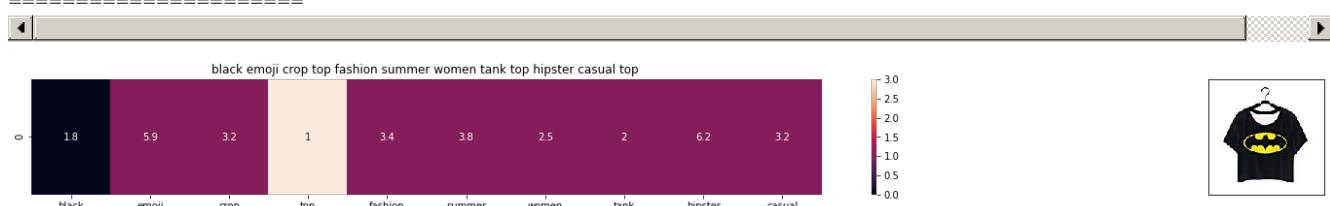


ASIN : B010V3BQZS

Brand : Doxi Supermall

euclidean distance from the given image : 7.667939547088641

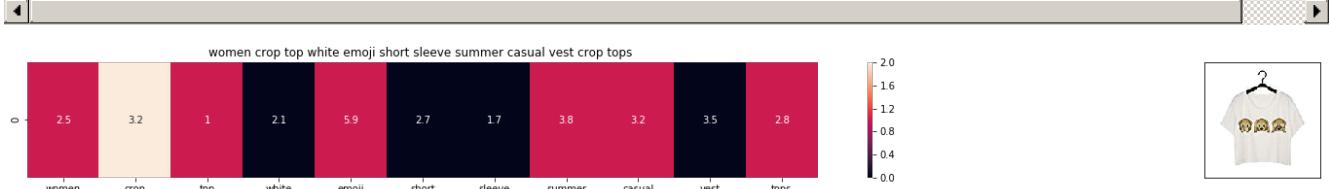
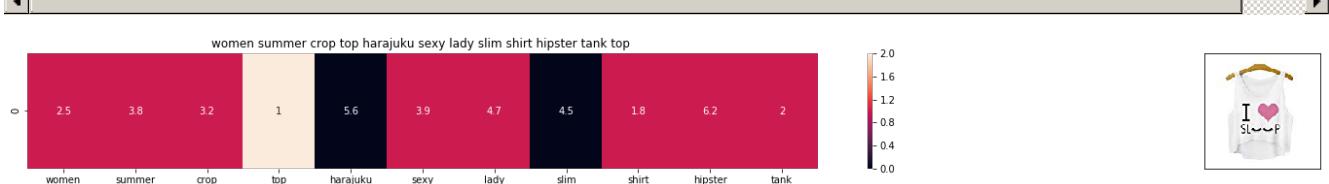
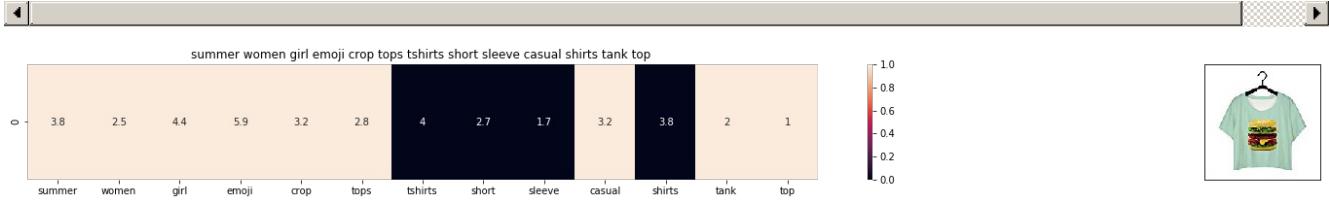
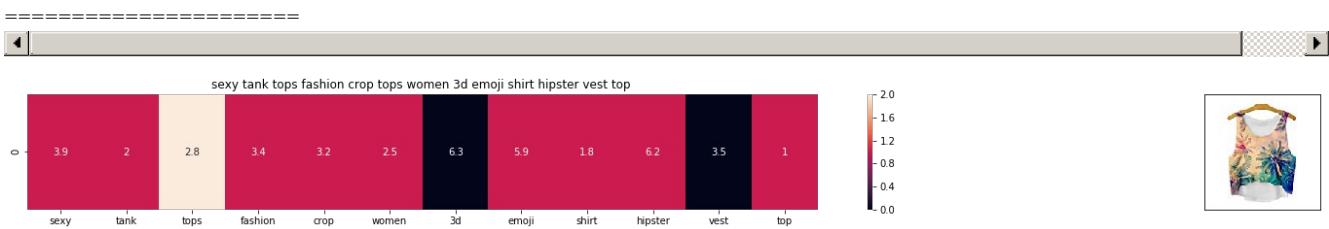
=====



ASIN : B0124E80M4

Brand : Doxi Supermall

euclidean distance from the given image : 8.39863603811248





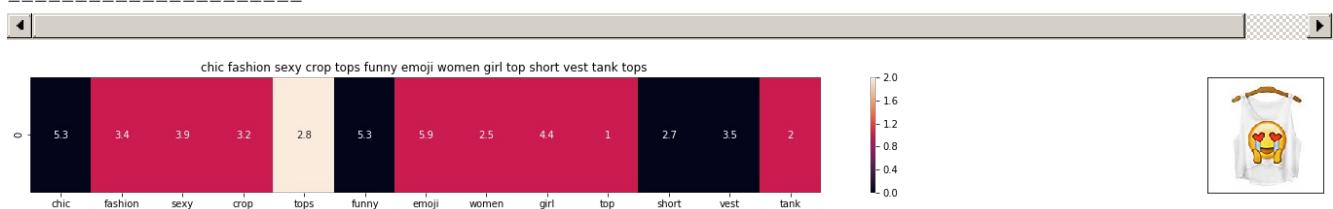
0.0

ASIN : B011RCJPR8

Brand : Chiclook Cool

euclidean distance from the given image : 12.760692810178545

=====

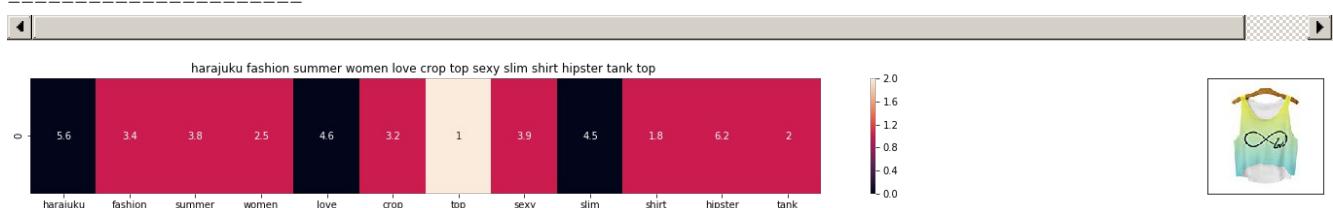


ASIN : B011RCJH58

Brand : Chiclook Cool

euclidean distance from the given image : 12.829128504563066

=====



ASIN : B010V350BU

Brand : Doxi Supermall

euclidean distance from the given image : 12.92180694297083

=====

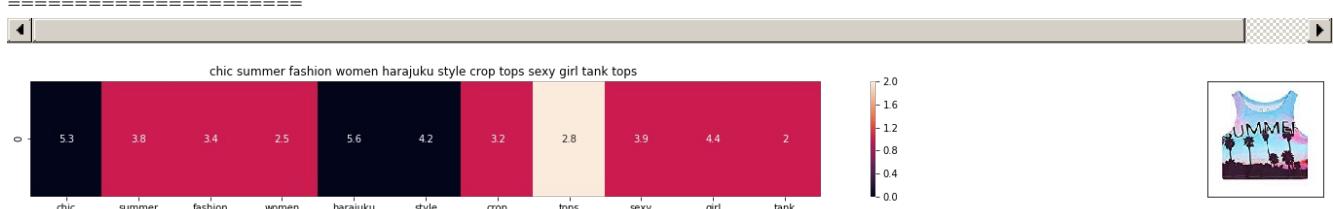


ASIN : B011RCJEMO

Brand : Chiclook Cool

euclidean distance from the given image : 13.474555736106787

=====

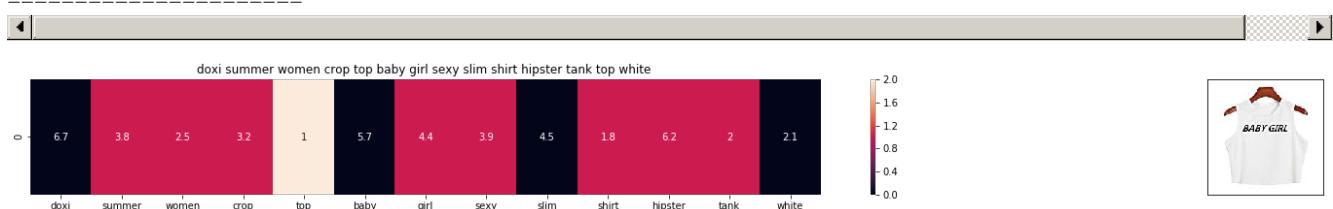


ASIN : B011OU51US

Brand : Chiclook Cool

euclidean distance from the given image : 13.713807644827545

=====

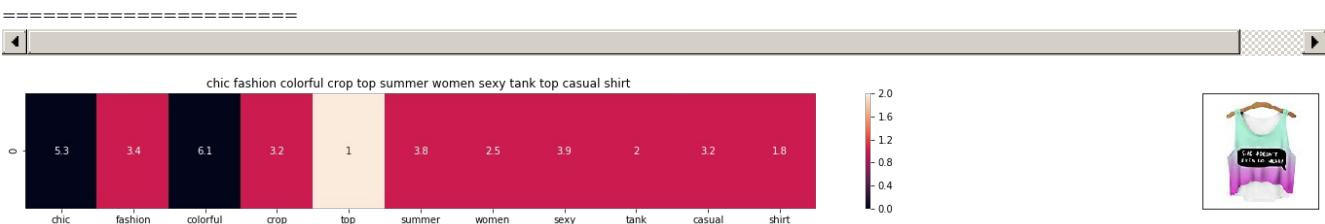


ASIN : B010V3A23U

Brand : Doxi Supermall

euclidean distance from the given image : 13.725325741546714

=====



```
ASIN : B011RCJ6UE
Brand : Chiclook Cool
euclidean distance from the given image : 13.746787336587824
```

[9] Text Semantics based product similarity

In [53]:

```
# 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[53]:

```
'\n# Set values for various parameters\nnum_features = 300      # Word vector dimensionality
\nmin_word_count = 1        # Minimum word count
\n                                num_workers = 4      # Number
of threads to run in parallel\ncontext = 10                # Context window size
\n\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 [35]:

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

# in this project we are using a pretrained model by google
# its 3.3G file, once you load this into your memory
# it occupies ~9Gb, so please do this step only if you have >12G of ram
# we will provide a pickle file which contains a dict ,
# and it contains all our corpus words as keys and model[word] as values
# To use this code-snippet, download "GoogleNews-vectors-negative300.bin"
# from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTtSS21pQmM/edit
# it's 1.9GB in size.

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

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

In [55]:

```
# Utility functions

def get_word_vec(sentence, doc_id, m_name):
    # sentence : title of the apparel
    # doc_id: document id in our corpus
    # m_name: model information it will take two values
        # if m_name == 'avg', we will append the model[i], w2v representation of word i
        # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)
    vec = []
    for i in sentence.split():
        if i in vocab:
            if m_name == 'weighted' and i in idf_title_vectorizer.vocabulary_:
                vec.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[i]] * model[i])
            elif m_name == 'avg':
                vec.append(model[i])
        else:
            # if the word in our corpus is not there in the google word2vec corpus, we are just
            # ignoring it
            vec.append(np.zeros(shape=(300,)))
    # we will return a numpy array of shape (#number of words in title * 300 ) 300 =
    len(w2v_model[word])
    # each row represents the word2vec representation of each word (weighted/avg) in given
    sentence
    return np.array(vec)

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

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

def heat_map_w2v(sentence1, sentence2, url, doc_id1, doc_id2, model):
    # sentance1 : title1, input apparel
    # sentance2 : title2, recommended apparel
    # url: apparel image url
    # doc_id1: document id of input apparel
    # doc_id2: document id of recommended apparel
    # model: it can have two values, 1. avg 2. weighted

    #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/avg) of 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])
# plotting 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 [56]:

```

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

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

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

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

```

[9.2] Average Word2Vec product similarity.

In [57]:

```

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 [58]:

```

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

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

```

```

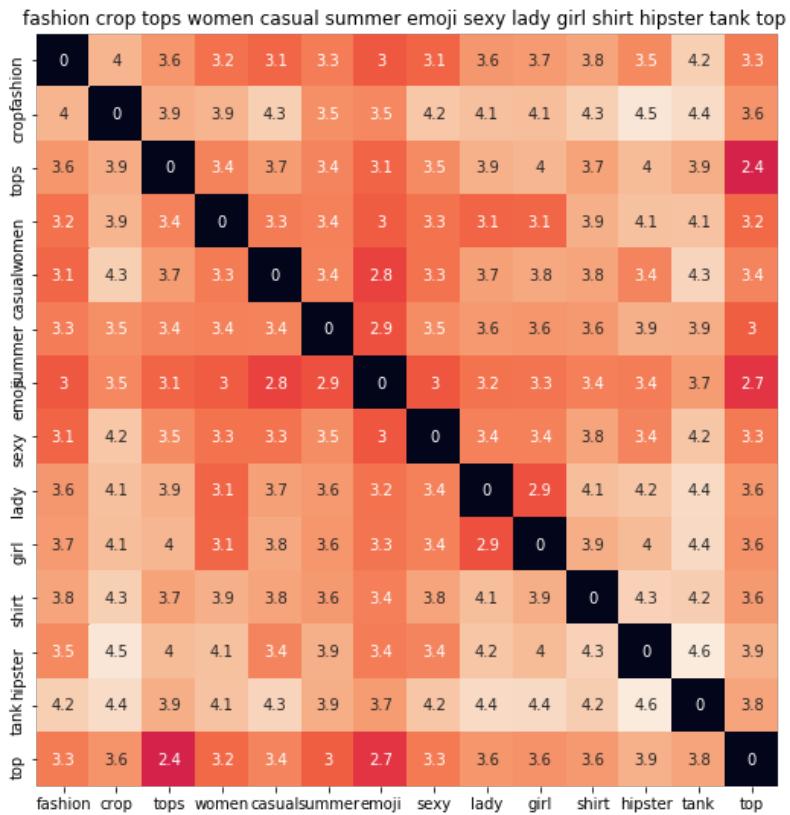
# np.argsort will return indices of 9 smallest distances
indices = np.argsort(pairwise_dist.flatten())[0:num_results]
#pdists will store the 9 smallest distances
pdists = np.sort(pairwise_dist.flatten())[0:num_results]

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

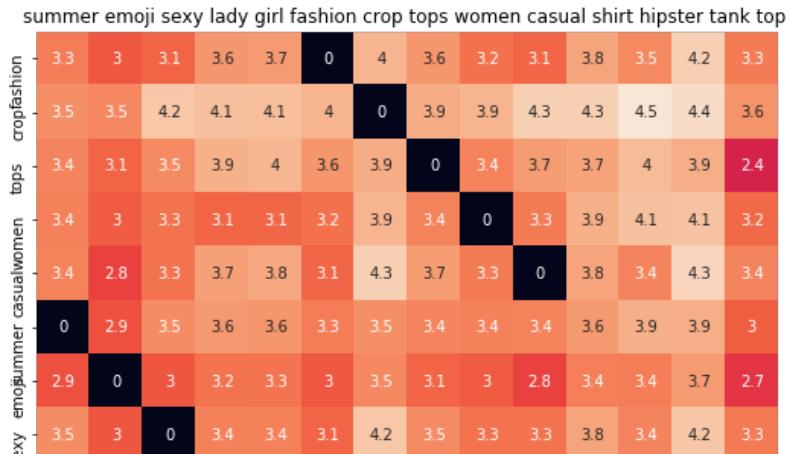
for i in range(0, len(indices)):
    heat_map_w2v(data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i], 'avg')
    print('ASIN :', data['asin'].loc[df_indices[i]])
    print('BRAND :', data['brand'].loc[df_indices[i]])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

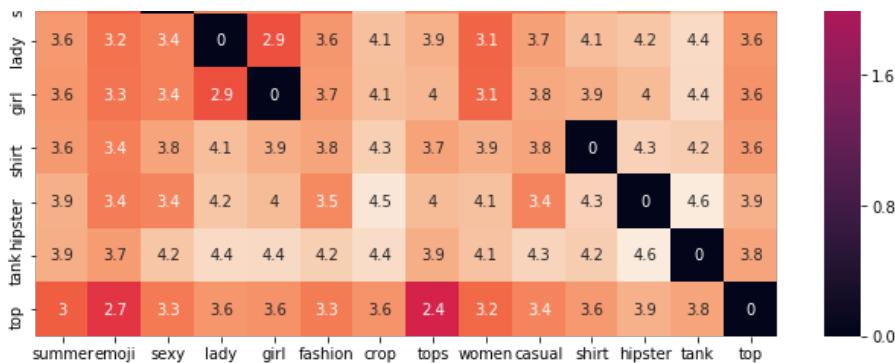
avg_w2v_model(12566, 20)
# in the give heat map, each cell contains the euclidean distance between words i, j

```



ASIN : B010V3B44G
 BRAND : Doxi Supermall
 euclidean distance from given input image : 0.0

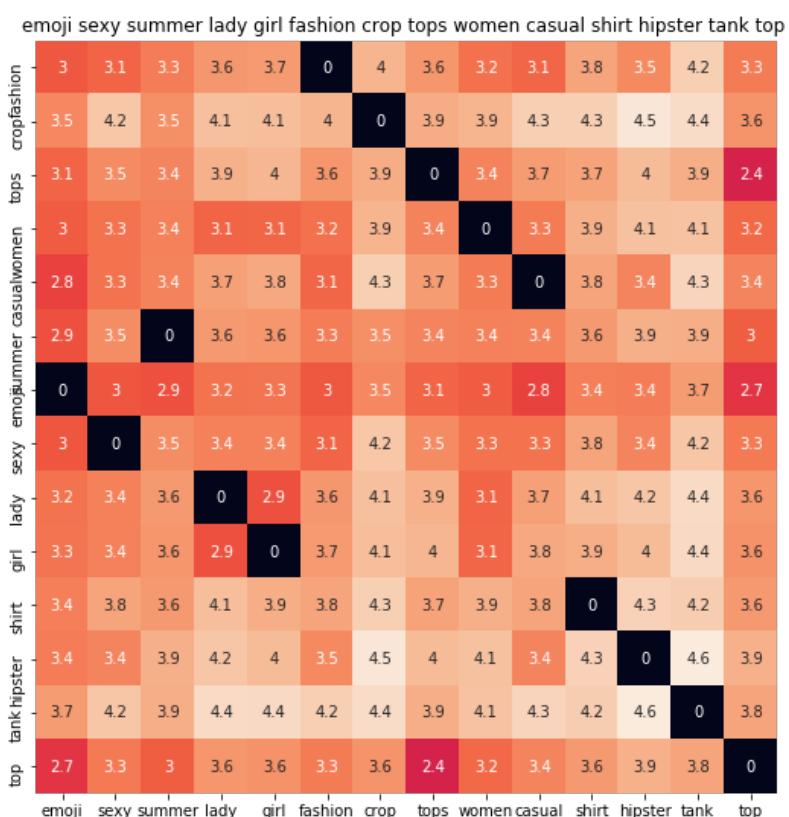




ASIN : B010V3BDII

BRAND : Doxi Supermall

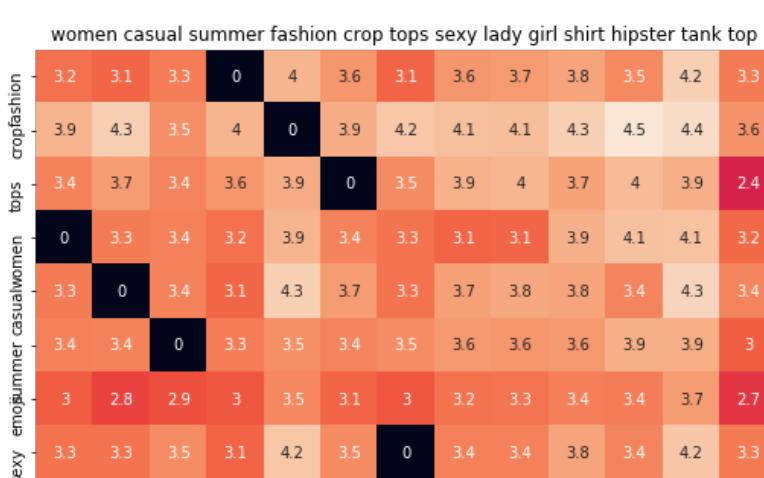
euclidean distance from given input image : 0.0

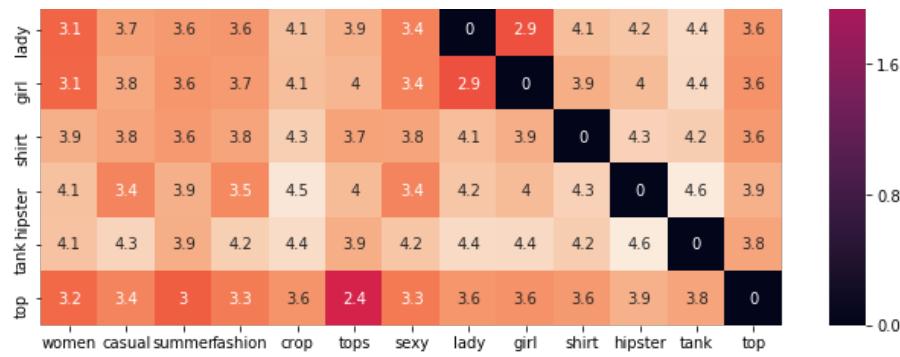


ASIN : B010V3BLWQ

BRAND : Doxi Supermall

euclidean distance from given input image : 0.0

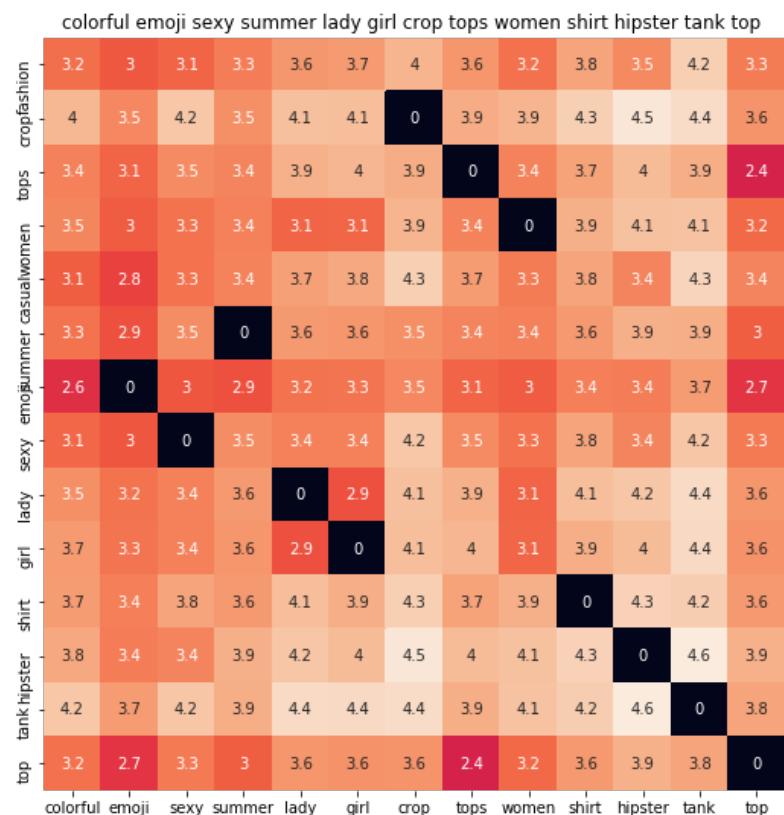




ASIN : B010V3AYSS

BRAND : Doxi Supermall

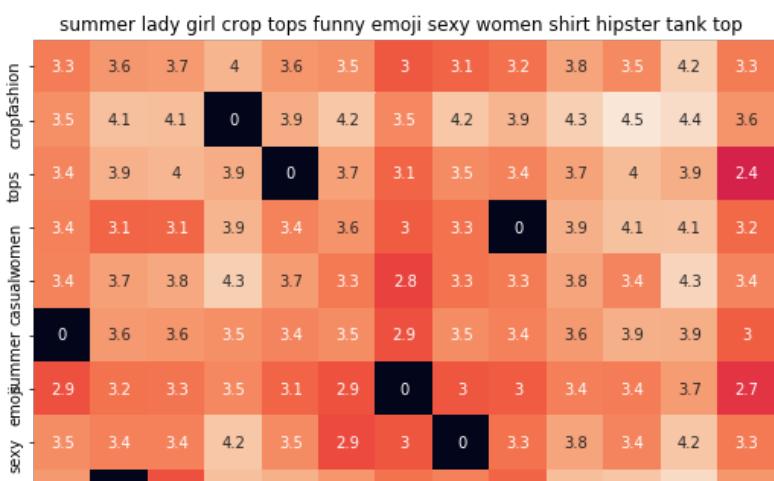
euclidean distance from given input image : 0.1336821

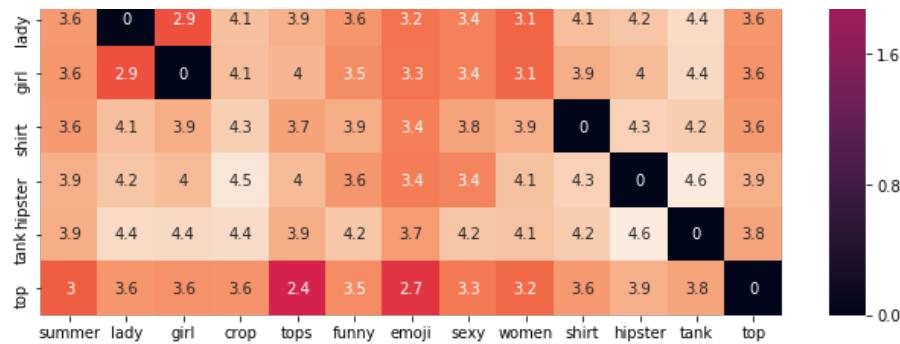


ASIN : B010V3BQZS

BRAND : Doxi Supermall

euclidean distance from given input image : 0.298377

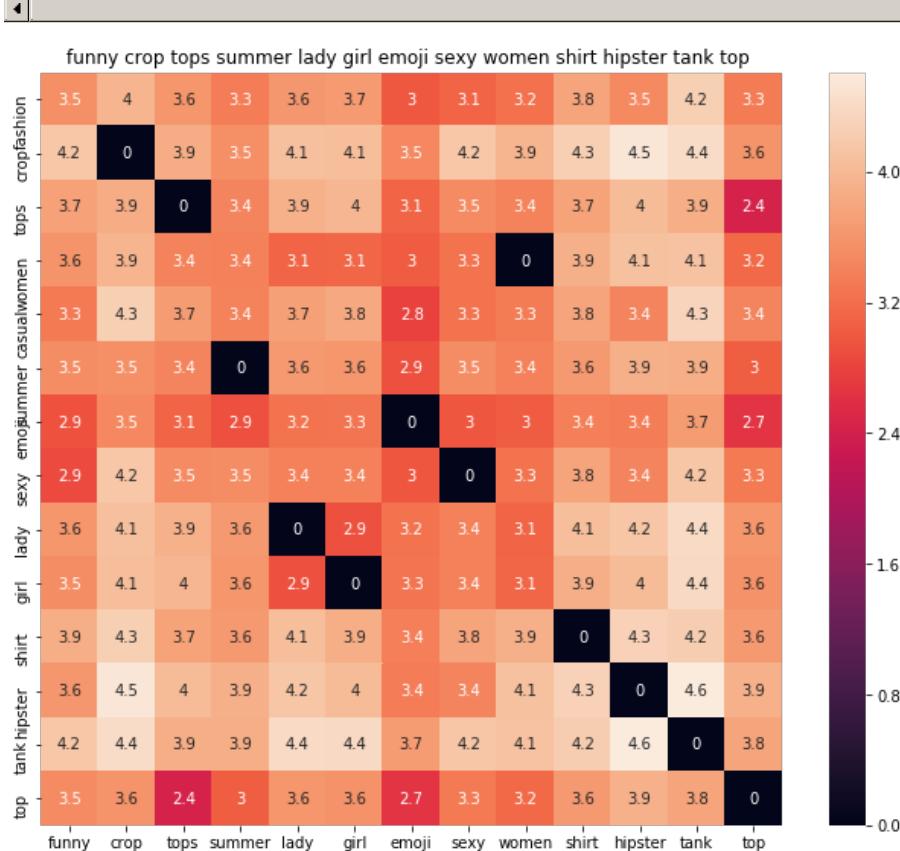




ASIN : B010V3BVMQ

BRAND : Doxi Supermall

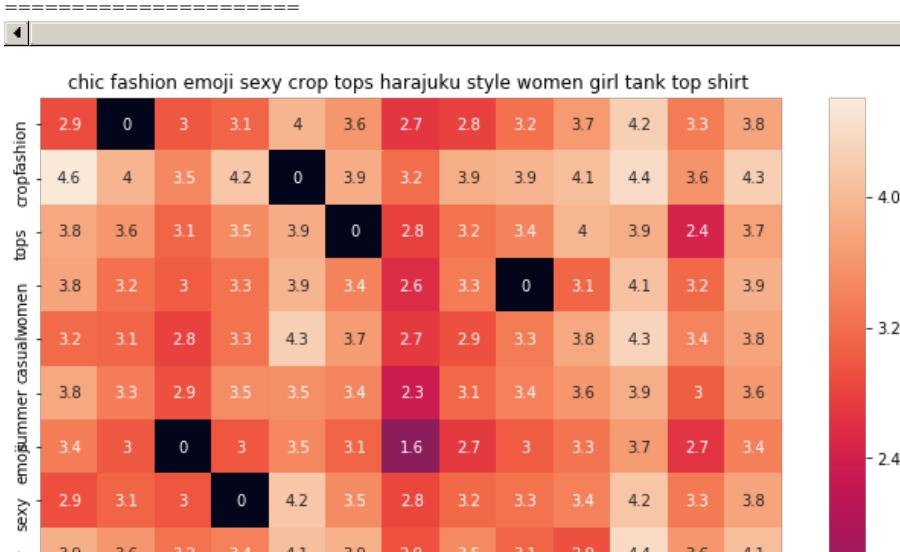
euclidean distance from given input image : 0.32746294

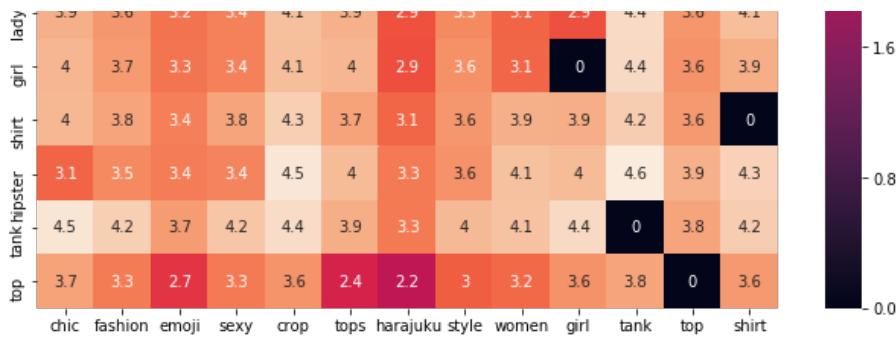


ASIN : B010V3C116

BRAND : Doxi Supermall

euclidean distance from given input image : 0.32746294

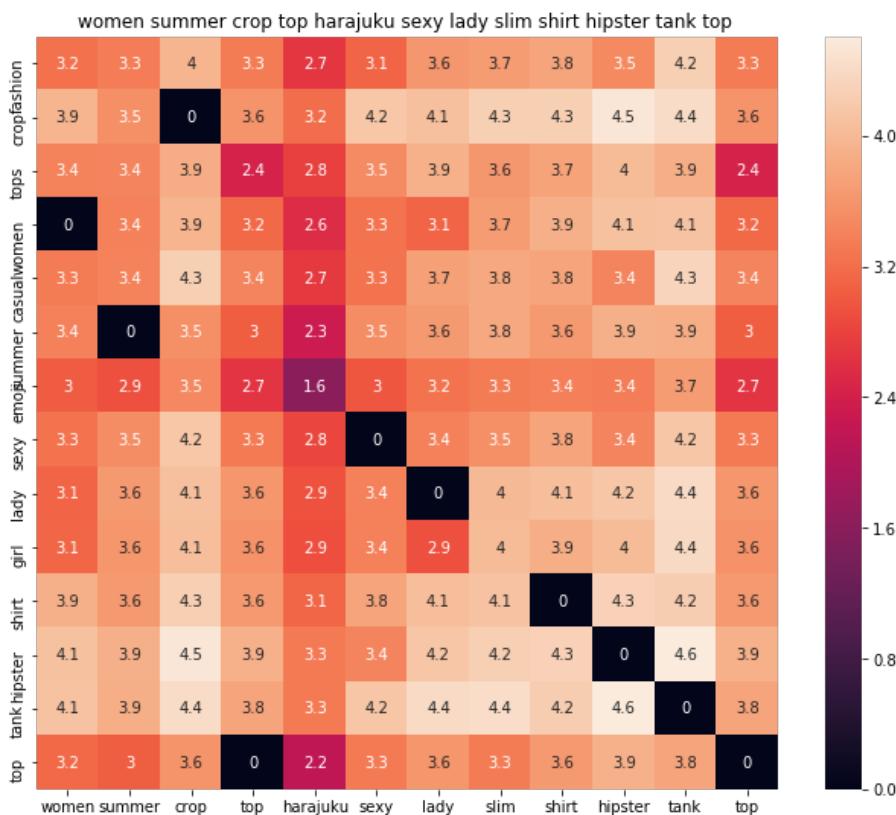




ASIN : B011RCJPR8

BRAND : Chiclook Cool

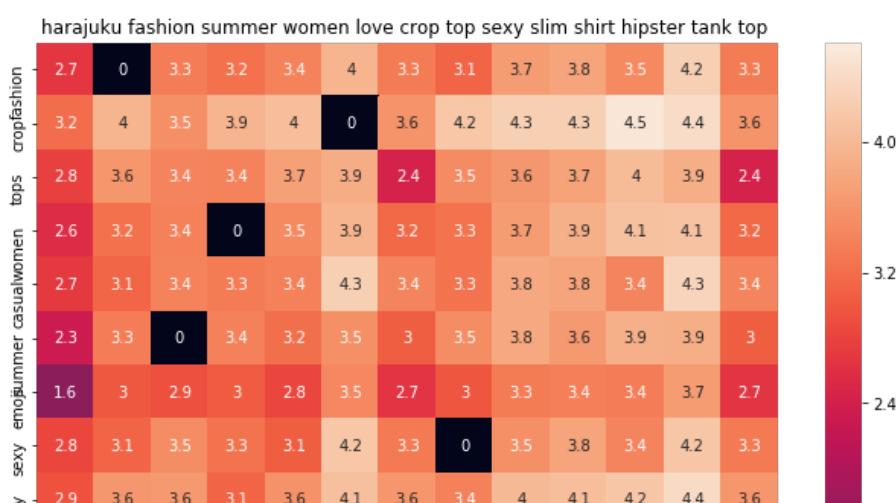
euclidean distance from given input image : 0.4097583



ASIN : B010V3EDEE

BRAND : Doxi Supermall

euclidean distance from given input image : 0.46806592

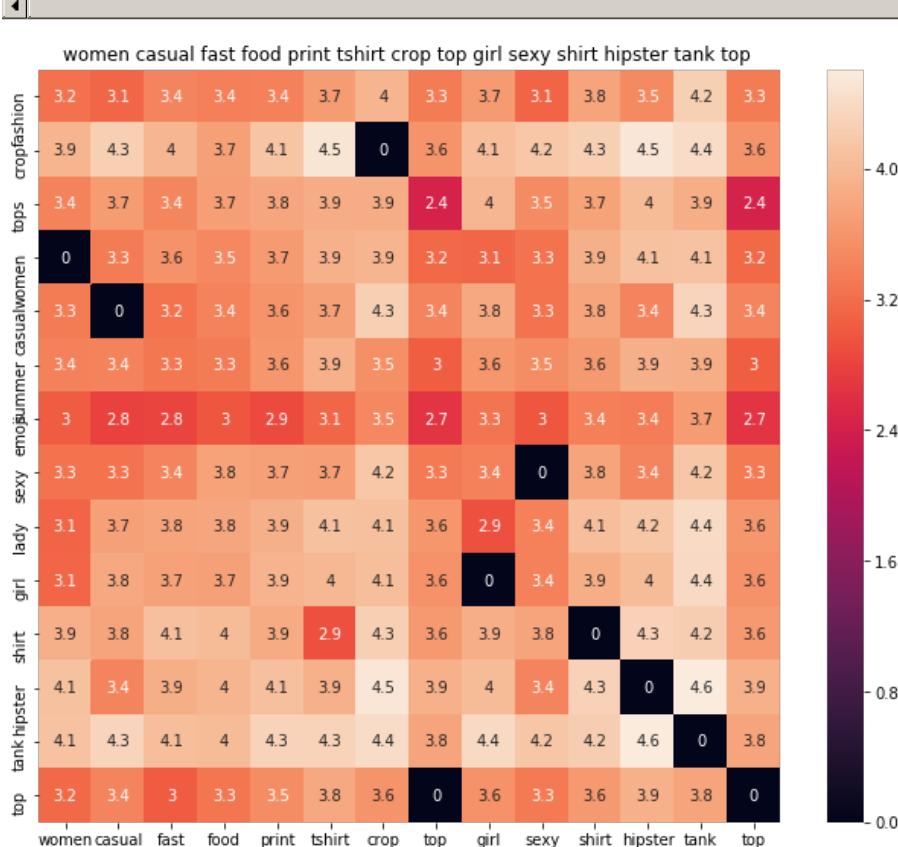




ASIN : B010V350BU

BRAND : Doxi Supermall

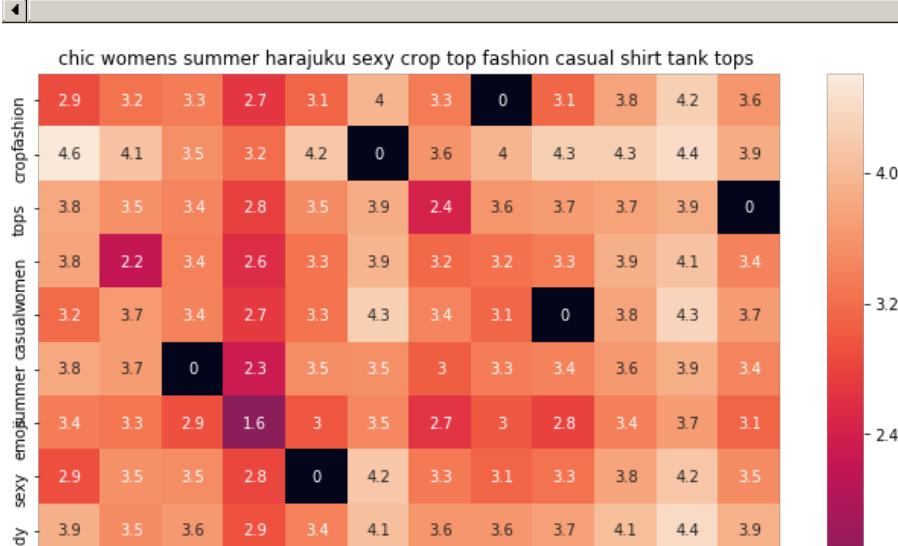
euclidean distance from given input image : 0.4957315

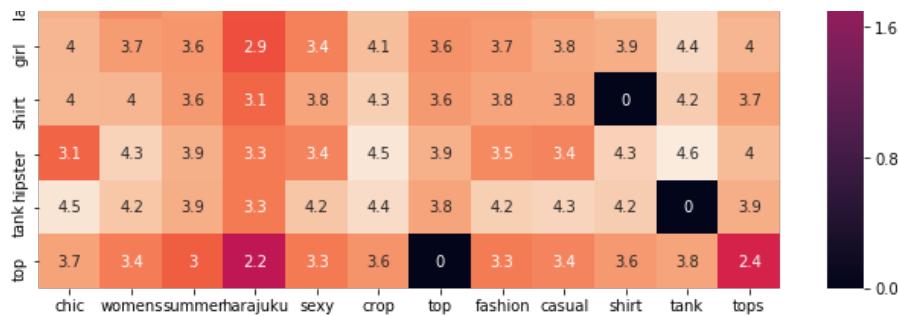


ASIN : B010V3AB50

BRAND : Doxi Supermall

euclidean distance from given input image : 0.50107557





ASIN : B011RCJEMO

BRAND : Chiclook Cool

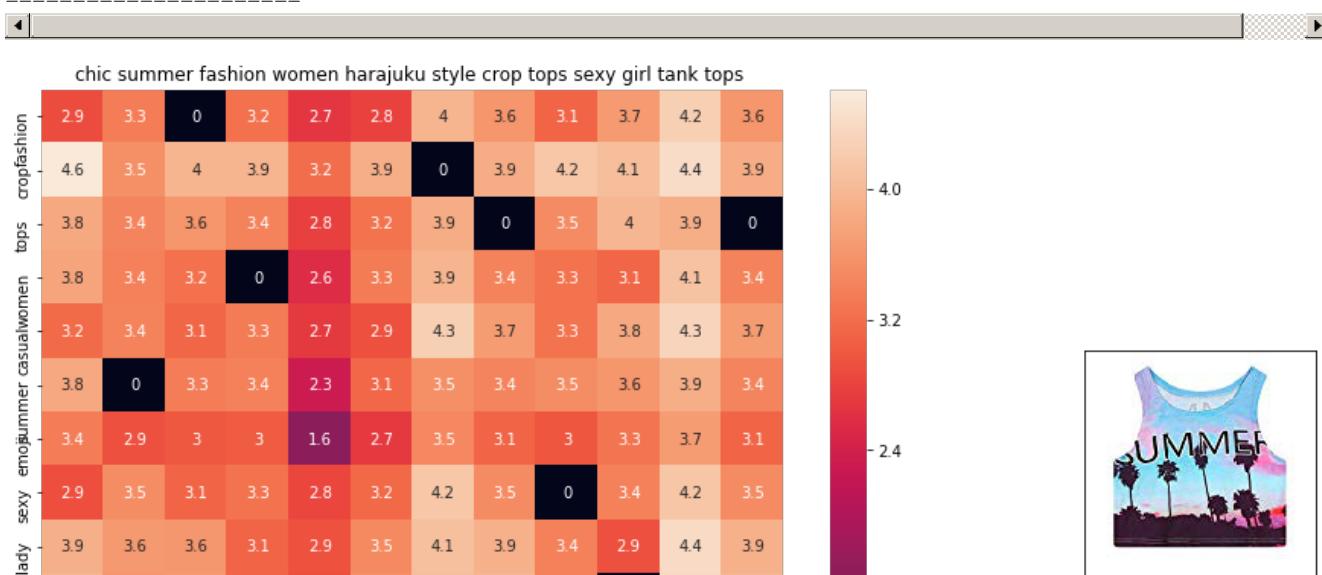
euclidean distance from given input image : 0.50272816

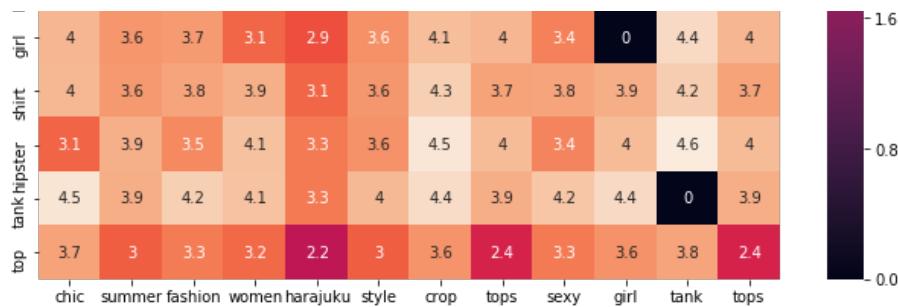


ASIN : B011RCJ6UE

BRAND : Chiclook Cool

euclidean distance from given input image : 0.50736403





ASIN : B011OU51US

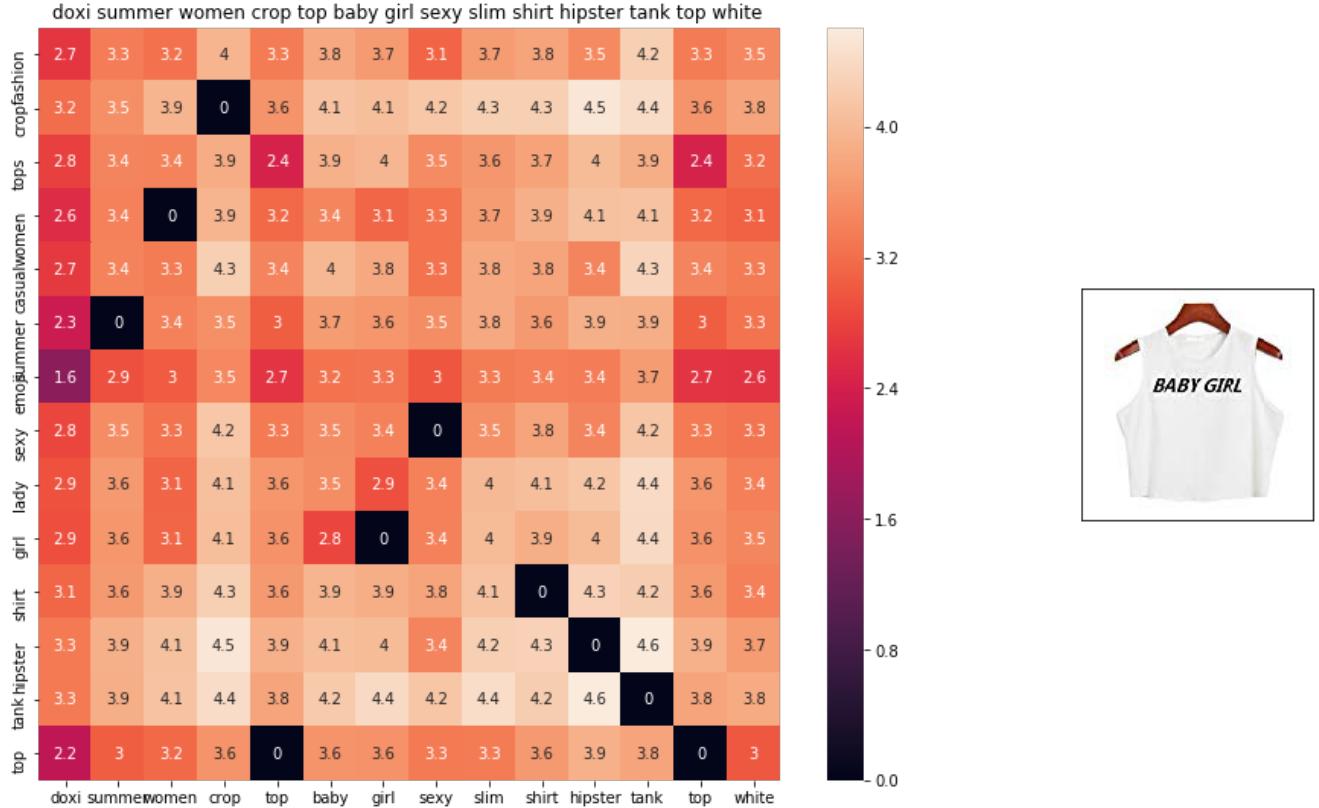
BRAND : Chiclook Cool

euclidean distance from given input image : 0.5160888

=====

=====

=====



ASIN : B010V3A23U

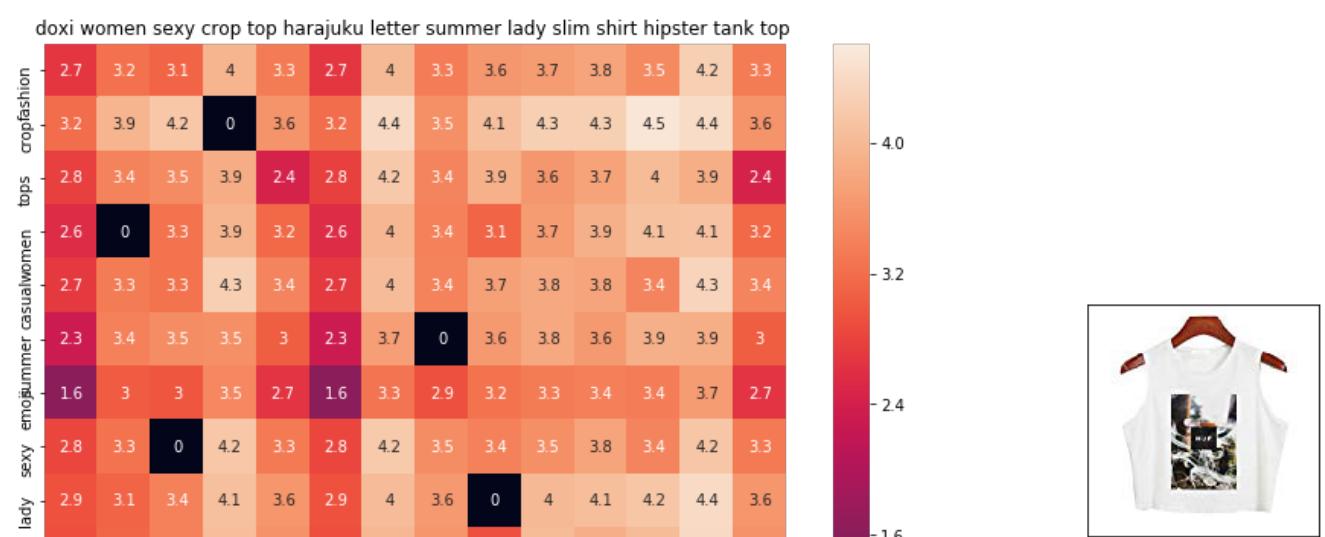
BRAND : Doxi Supermall

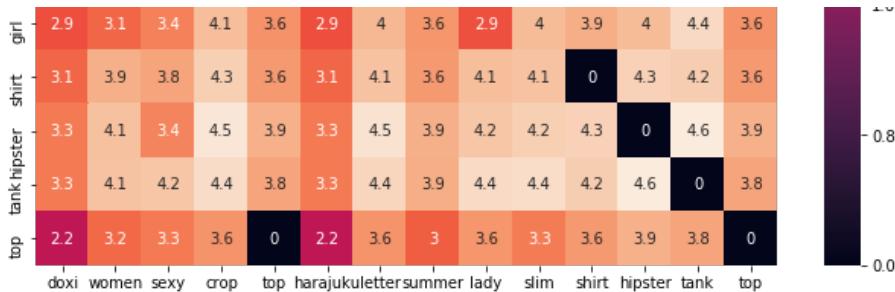
euclidean distance from given input image : 0.5262786

=====

=====

=====





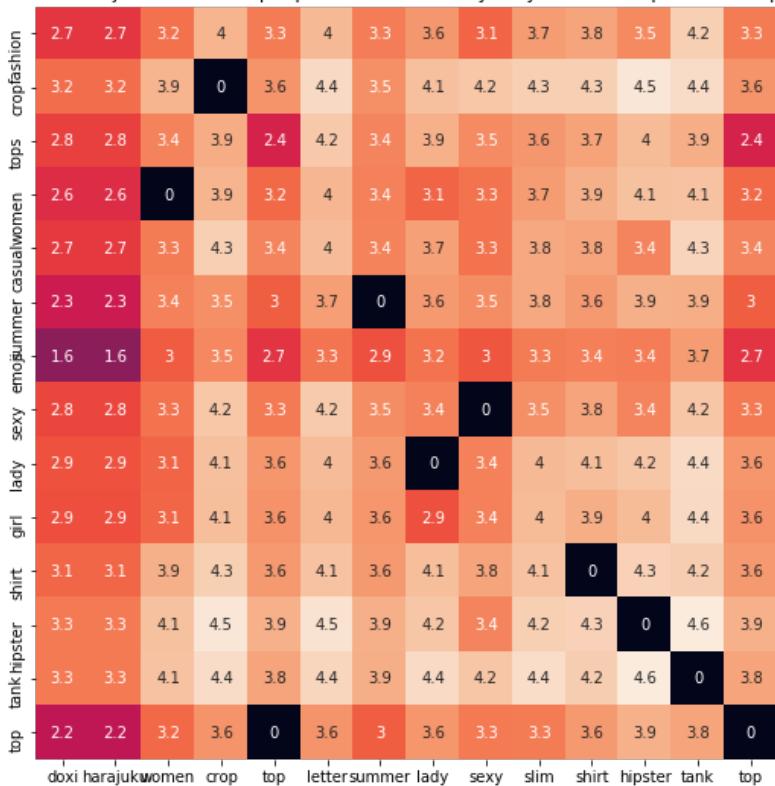
ASIN : B010V39146

BRAND : Doxi Supermall

euclidean distance from given input image : 0.5270717



doxi harajuku women crop top letter summer lady sexy slim shirt hipster tank top



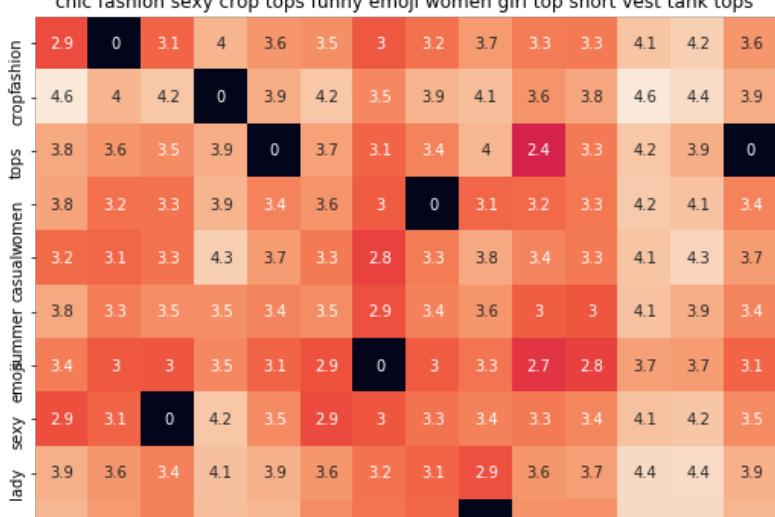
ASIN : B010V380LQ

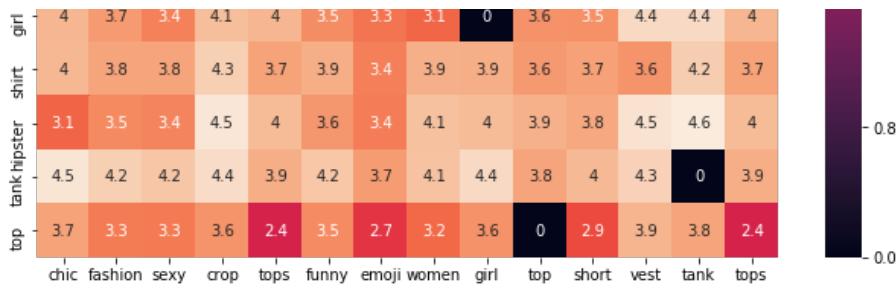
BRAND : Doxi Supermall

euclidean distance from given input image : 0.5270717



chic fashion sexy crop tops funny emoji women girl top short vest tank tops

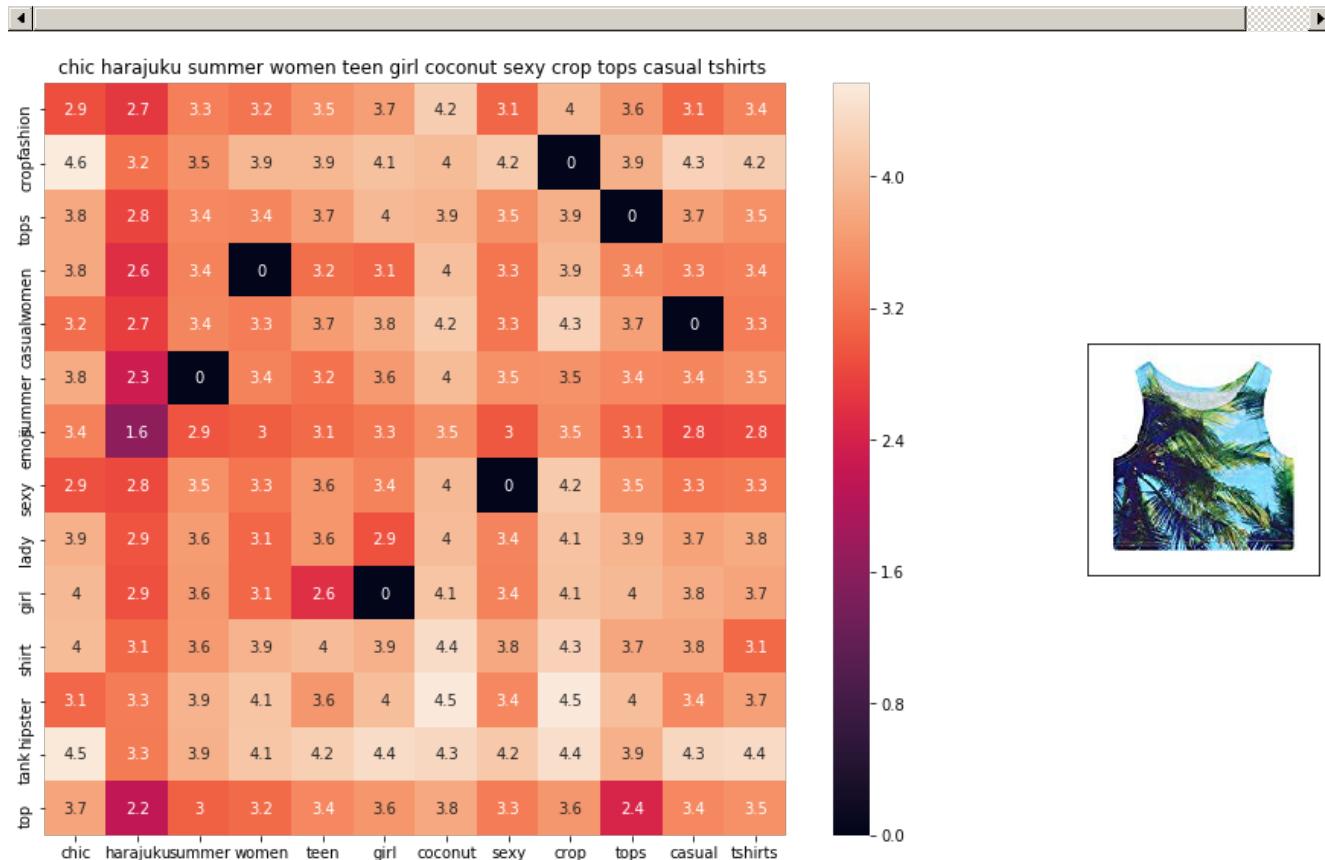




ASIN : B011RCJH58

BRAND : Chiclook Cool

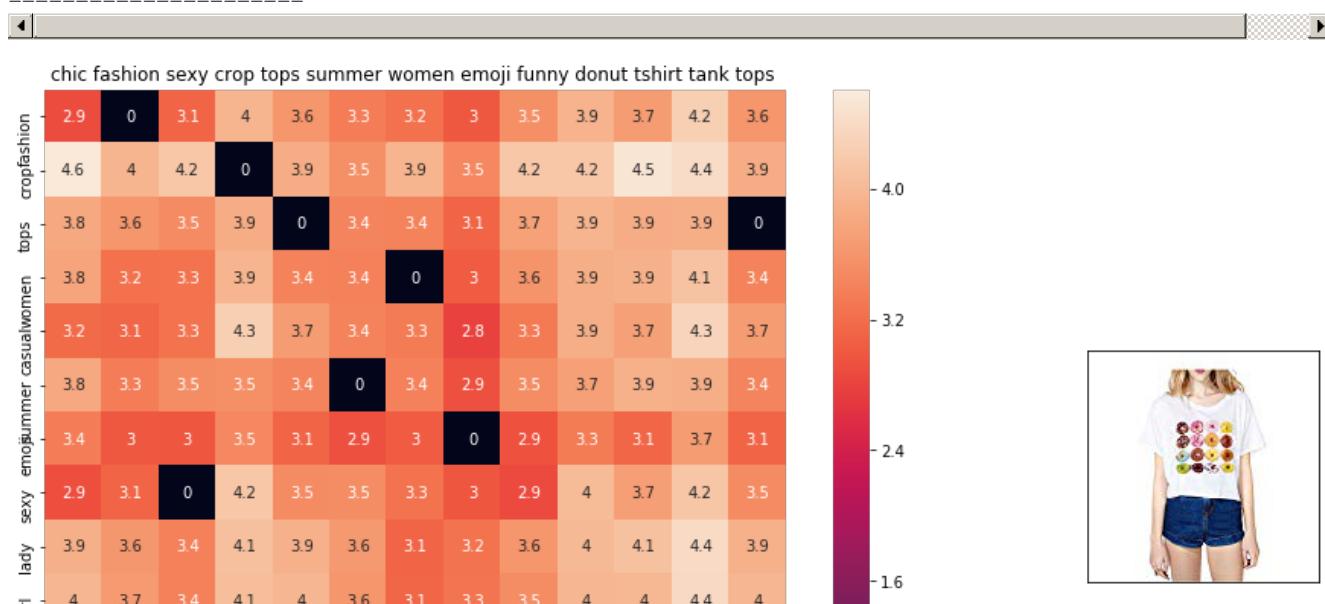
euclidean distance from given input image : 0.5281698

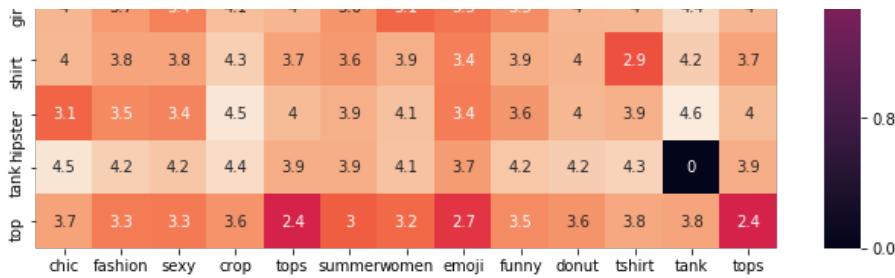


ASIN : B011OU4R08

BRAND : Chiclook Cool

euclidean distance from given input image : 0.55456346





ASIN : B011UEUTQE
 BRAND : Chiclook Cool
 euclidean distance from given input image : 0.5734916

[9.4] IDF weighted Word2Vec for product similarity

In [59]:

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

In [60]:

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

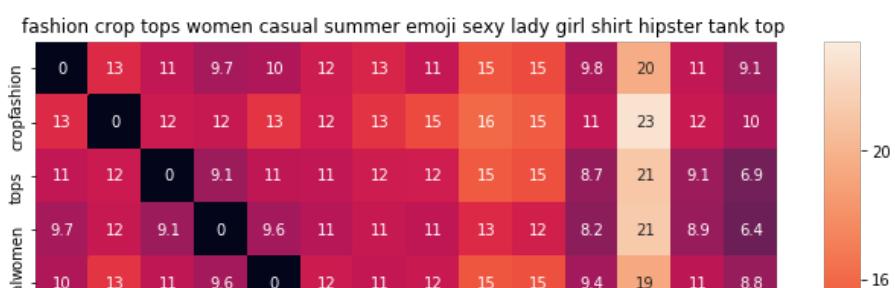
    # pairwise_dist will store the distance from given input apparel to all remaining apparels
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y> / (||X|| * ||Y||)
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = pairwise_distances(w2v_title_weight,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

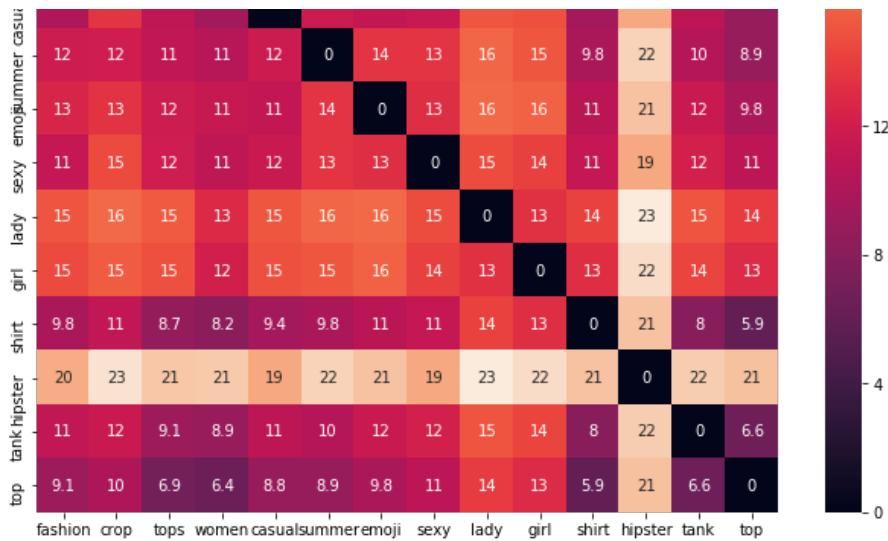
    # np.argsort will return indices of 9 smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists = np.sort(pairwise_dist.flatten())[0:num_results]

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

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

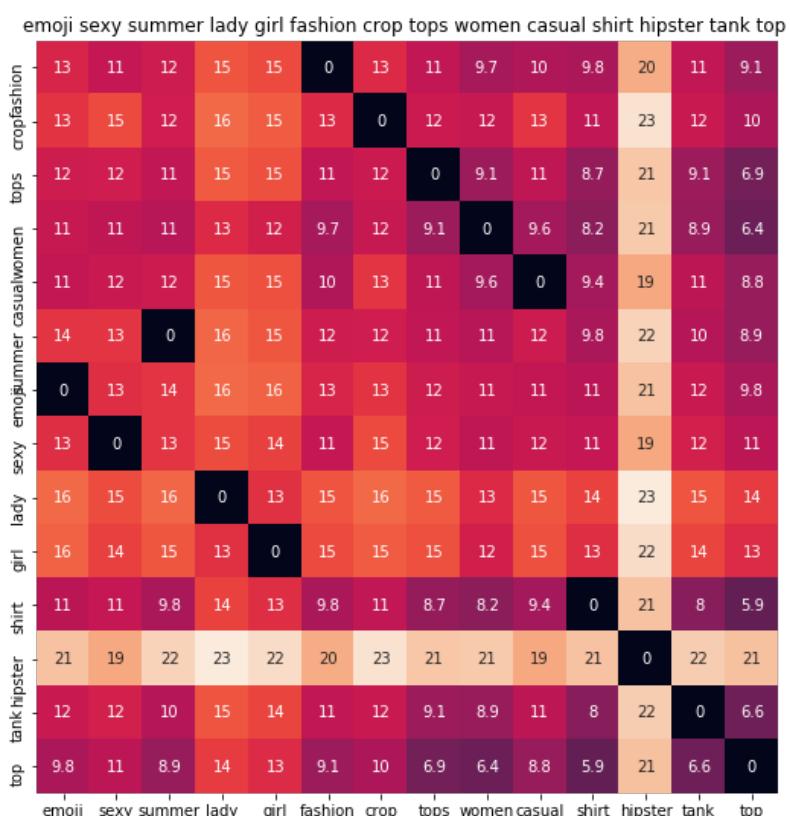




ASIN : B010V3B44G

Brand : Doxi Supermall

euclidean distance from input : 0.001953125

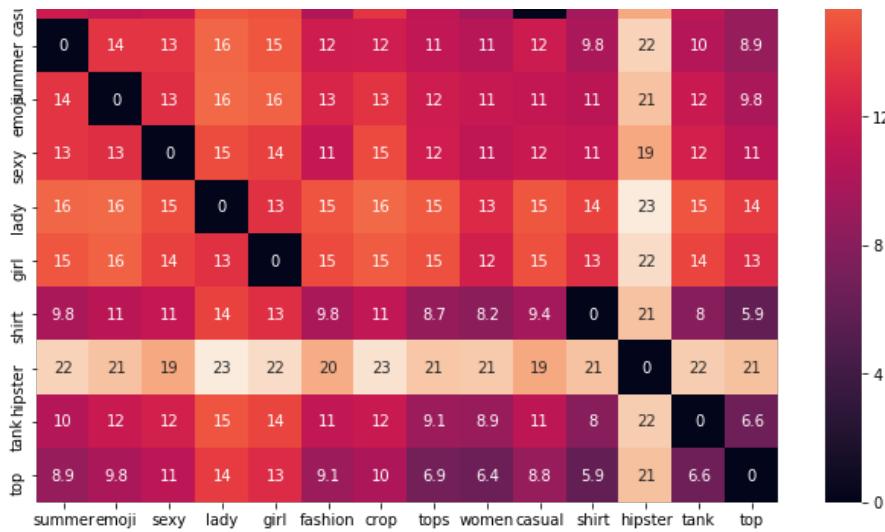


ASIN : B010V3BLWQ

Brand : Doxi Supermall

euclidean distance from input : 0.001953125

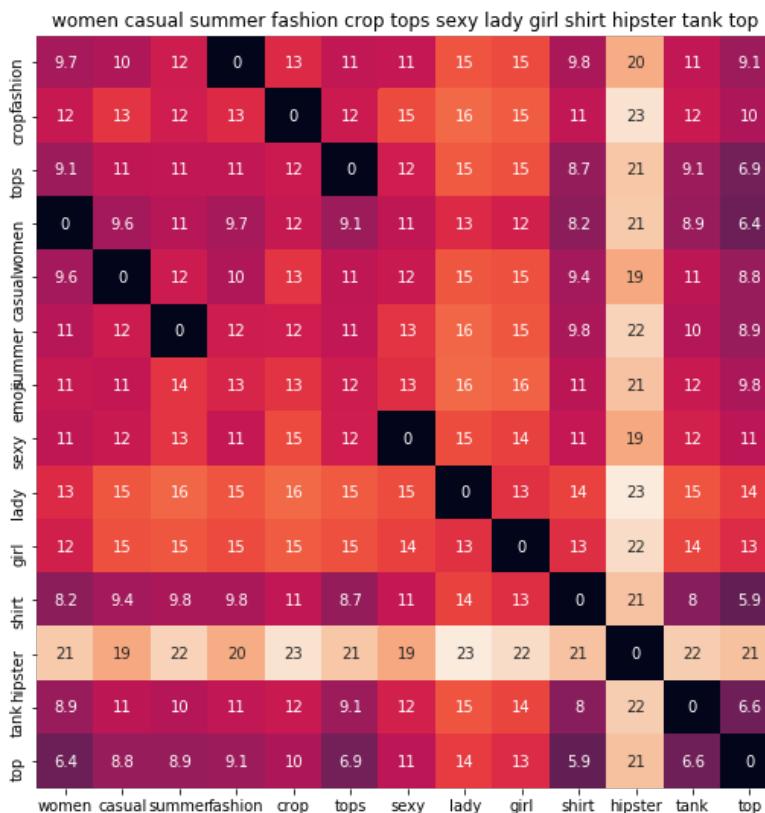




ASIN : B010V3BDII

Brand : Doxi Supermall

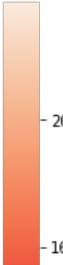
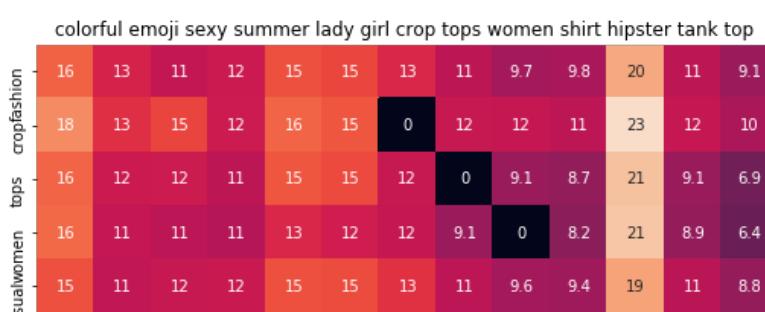
euclidean distance from input : 0.001953125

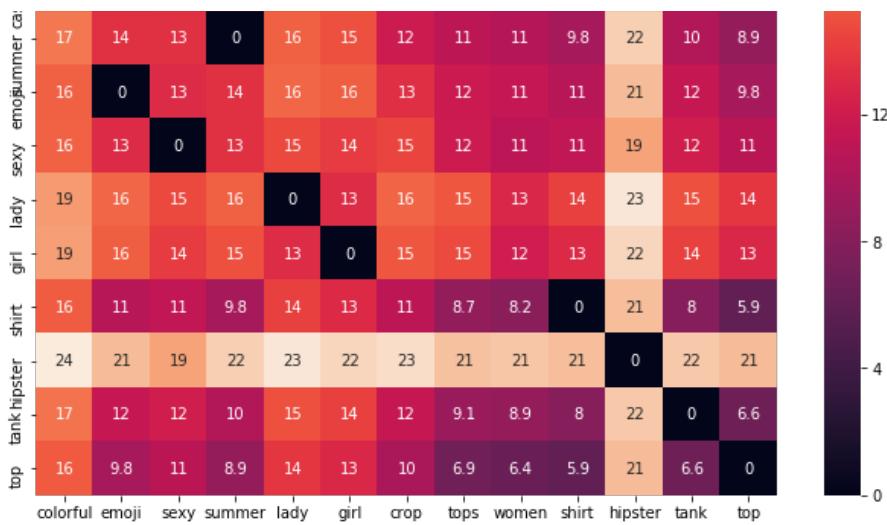


ASIN : B010V3AYSS

Brand : Doxi Supermall

euclidean distance from input : 0.6962308

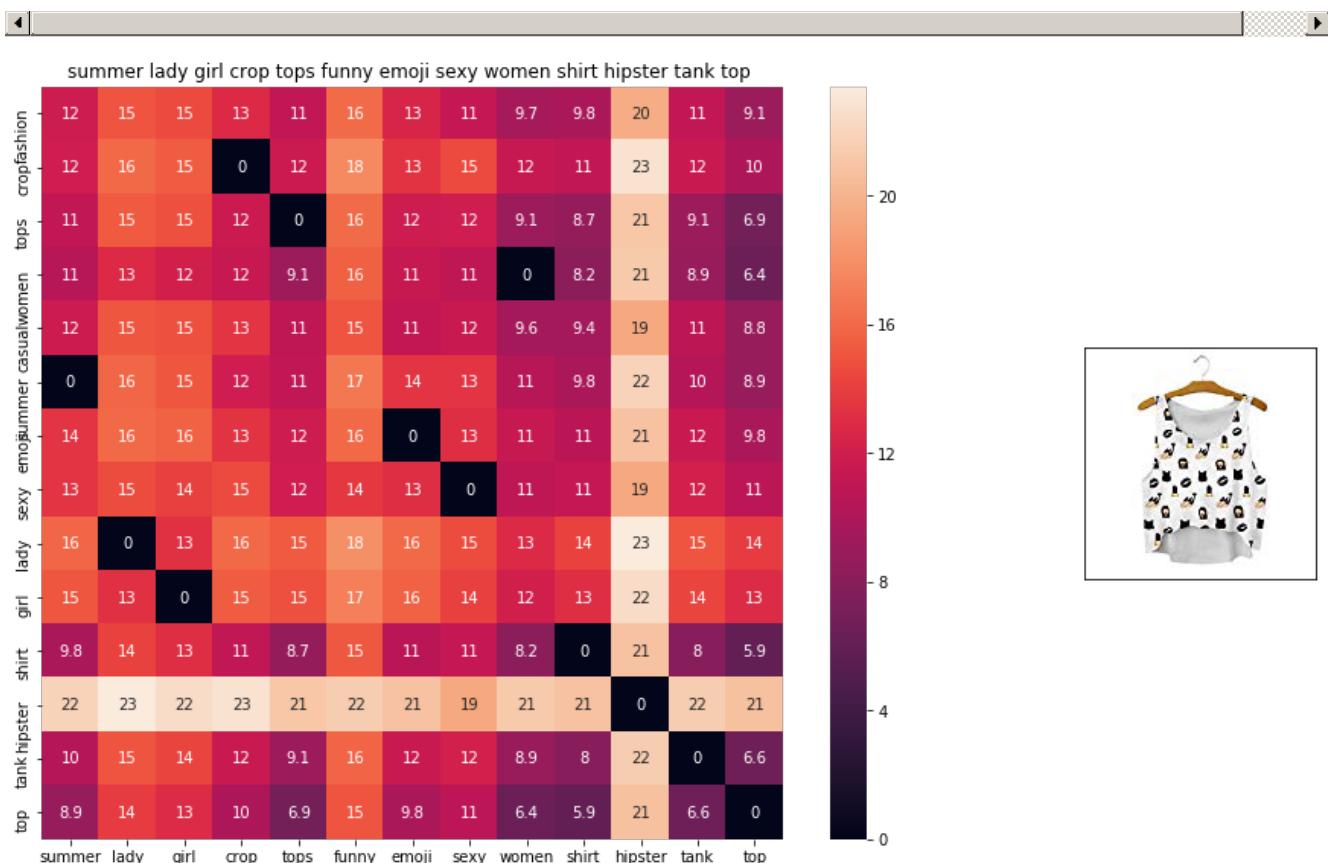




ASIN : B010V3BQZS

Brand : Doxi Supermall

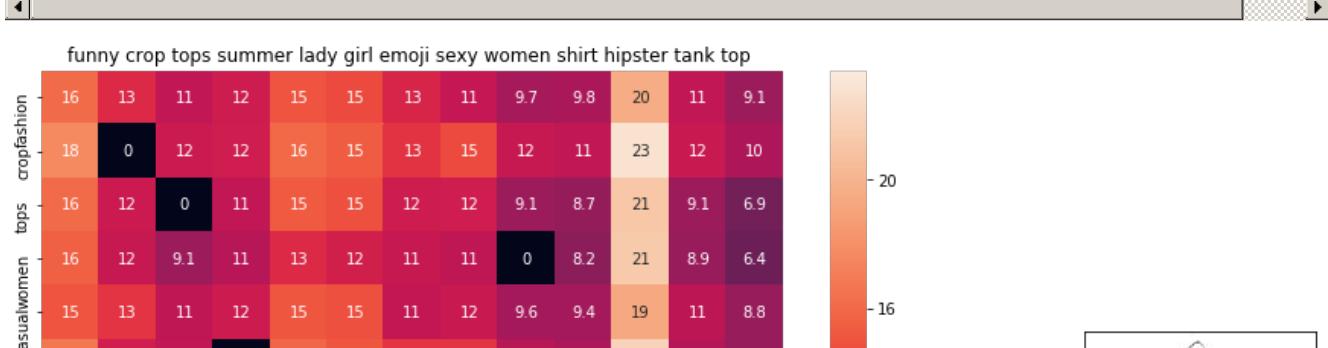
euclidean distance from input : 1.2872716

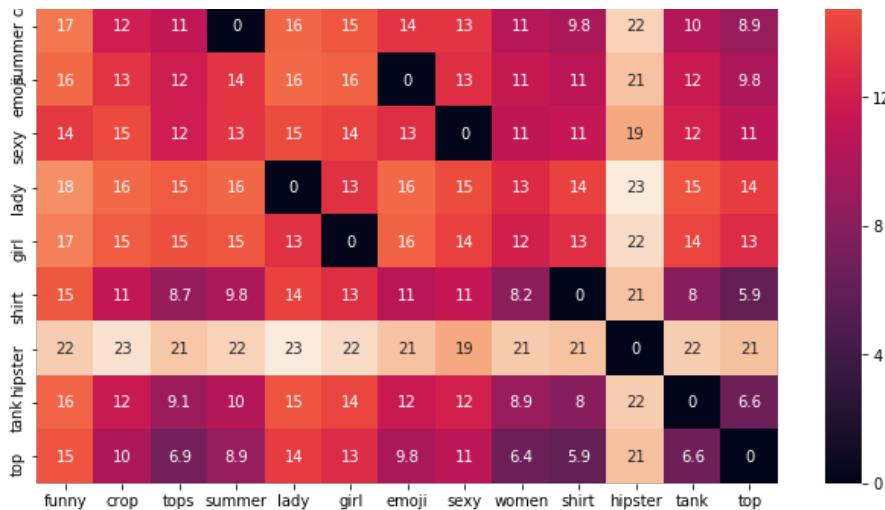


ASIN : B010V3BVMQ

Brand : Doxi Supermall

euclidean distance from input : 1.3530484

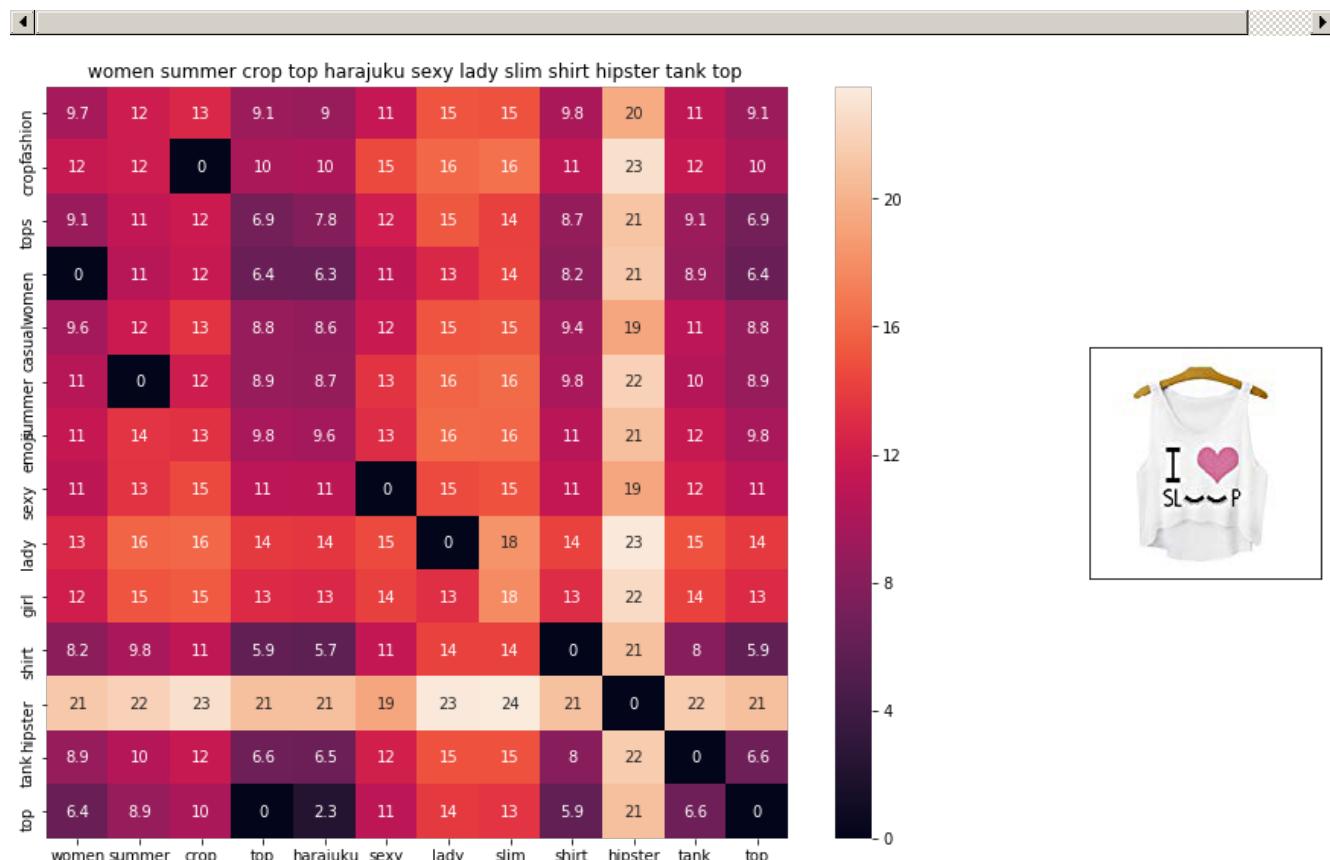




ASIN : B010V3C116

Brand : Doxi Supermall

euclidean distance from input : 1.3530484

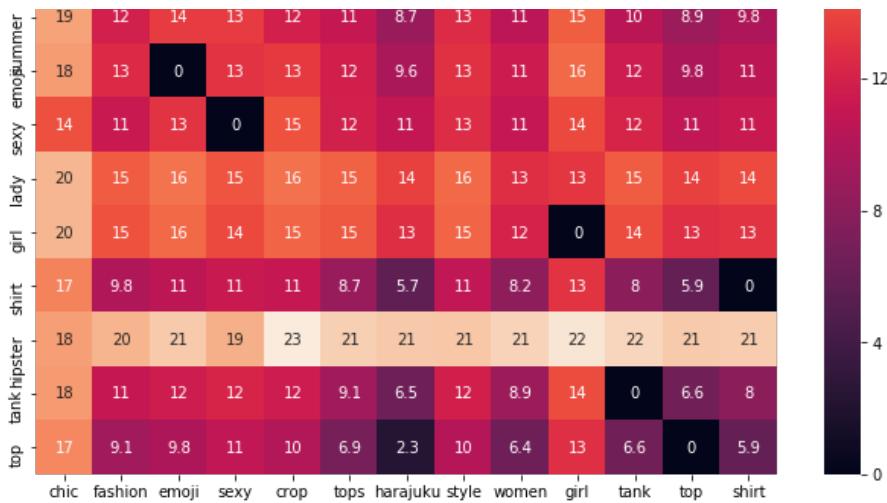


ASIN : B010V3EDEE

Brand : Doxi Supermall

euclidean distance from input : 1.7434149

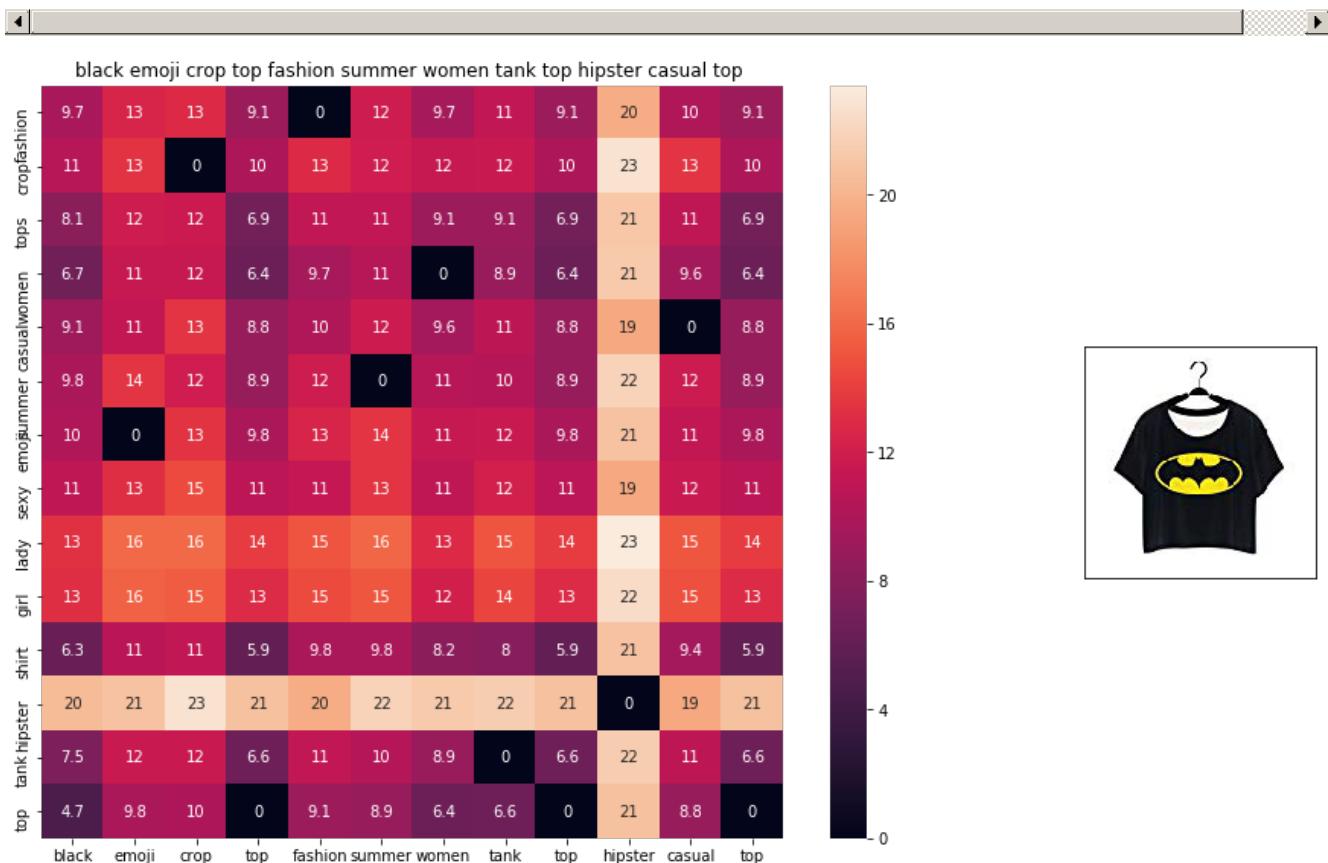




ASIN : B011RCJPR8

Brand : Chiclook Cool

euclidean distance from input : 1.9345262

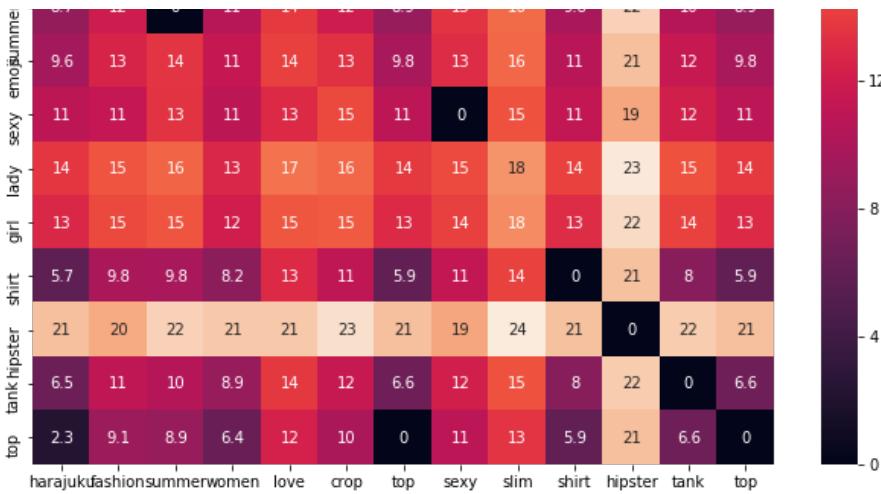


ASIN : B0124E80M4

Brand : Doxi Supermall

euclidean distance from input : 2.0321443

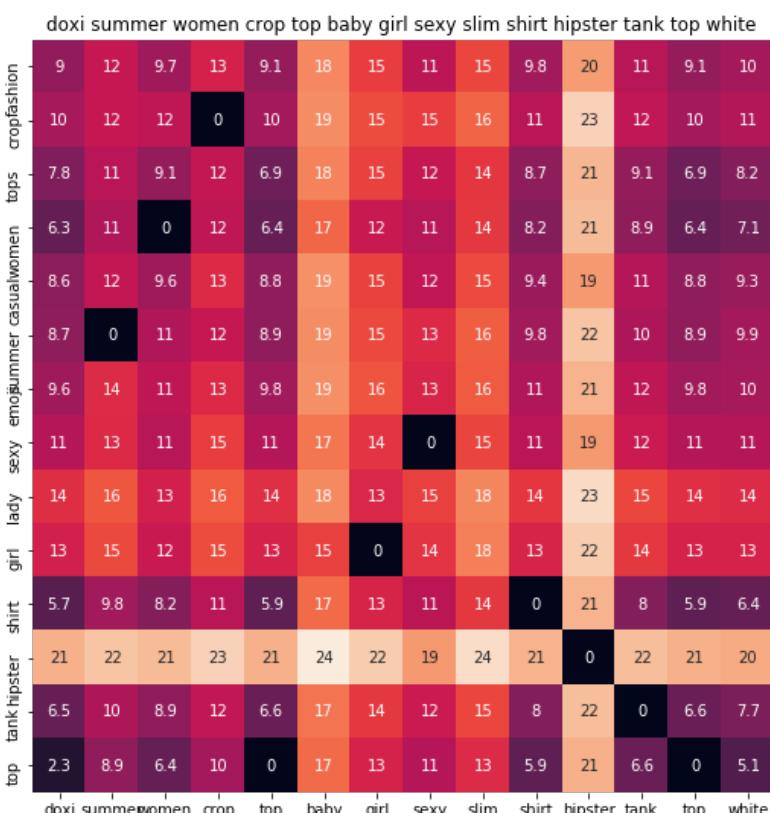




ASIN : B010V350BU

Brand : Doxi Supermall

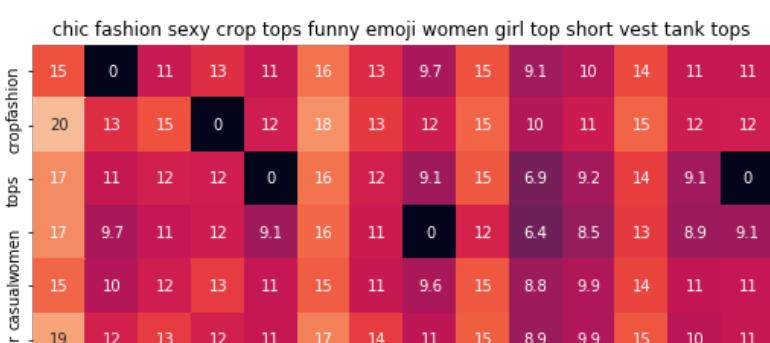
euclidean distance from input : 2.0631943

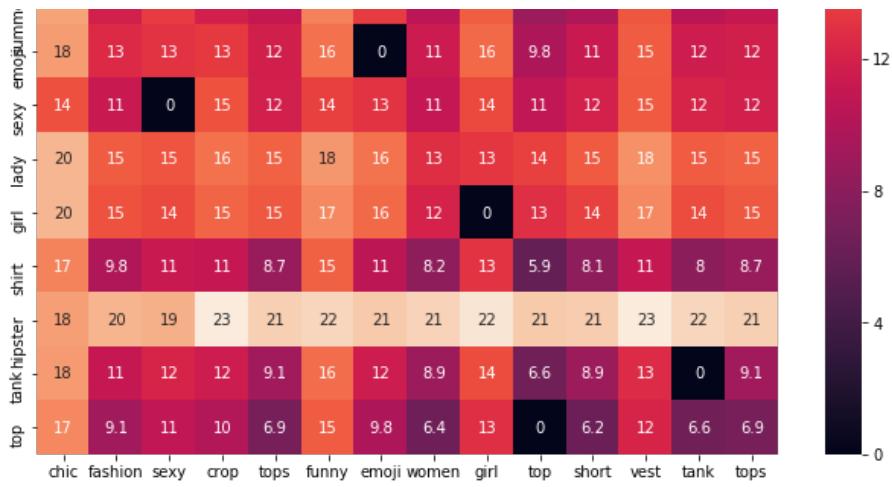


ASIN : B010V3A23U

Brand : Doxi Supermall

euclidean distance from input : 2.124369

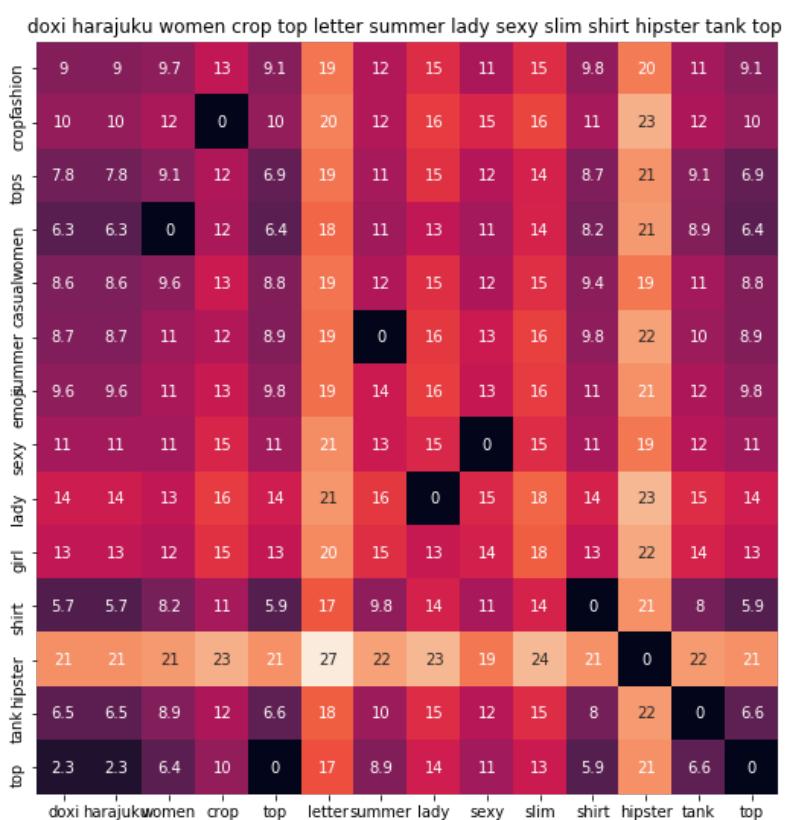




ASIN : B011RCJH58

Brand : Chiclook Cool

euclidean distance from input : 2.1507082



ASIN : B010V380LQ

Brand : Doxi Supermall

euclidean distance from input : 2.1674926

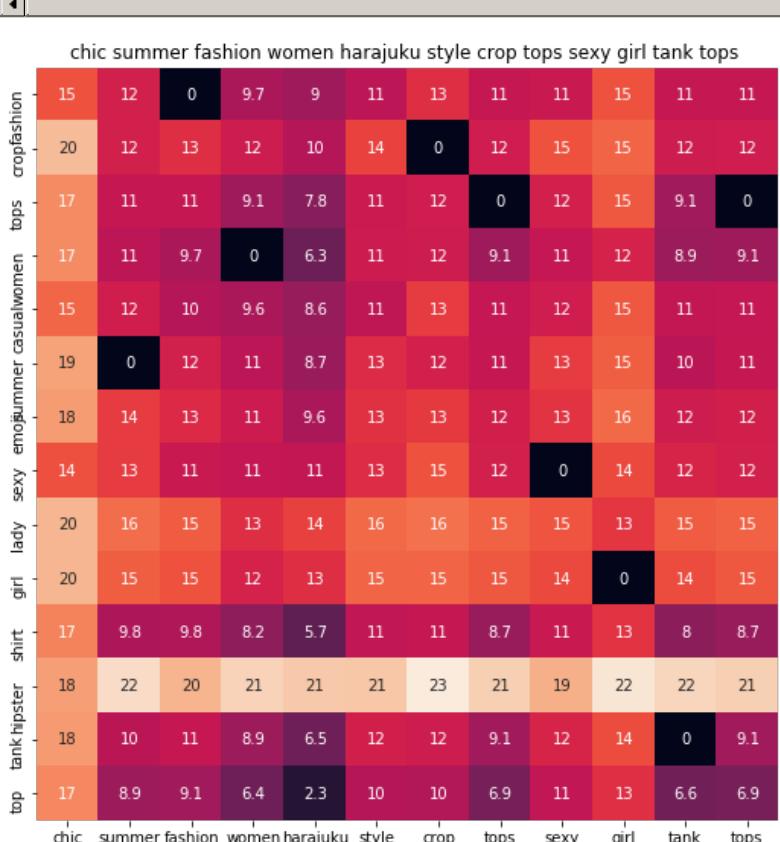




ASIN : B010V39146

Brand : Doxi Supermall

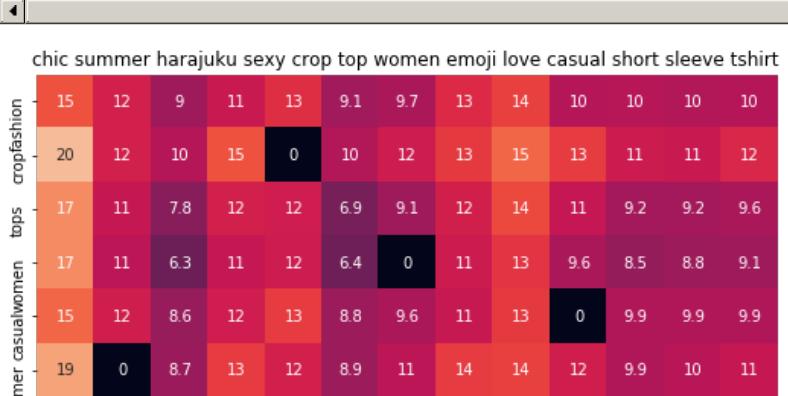
euclidean distance from input : 2.1674926

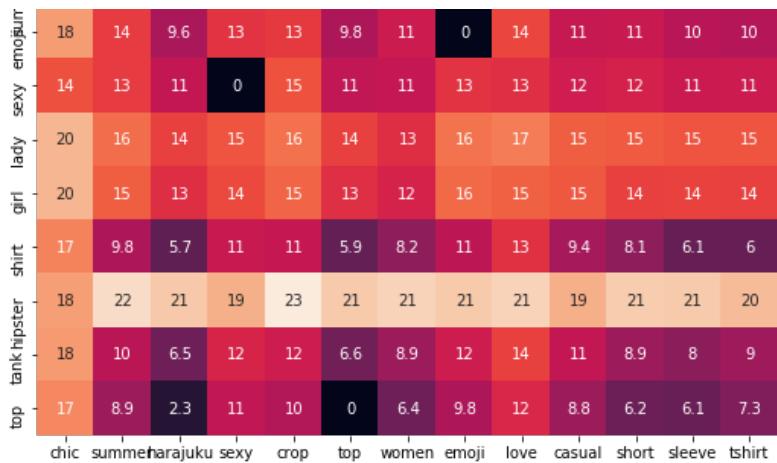


ASIN : B011OU51US

Brand : Chiclook Cool

euclidean distance from input : 2.2320182

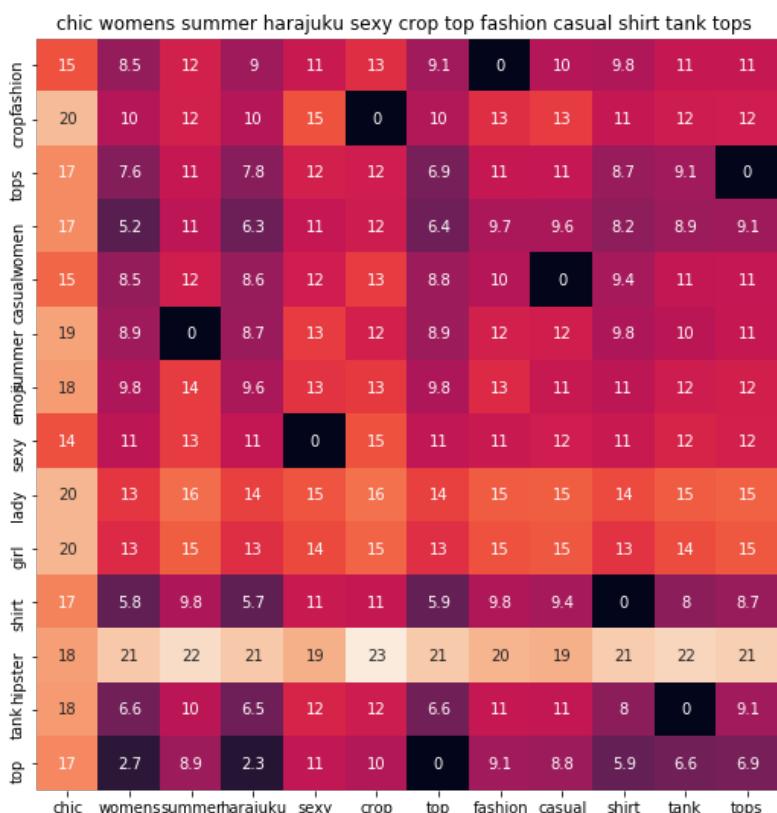




ASIN : B011UEVF40

Brand : Chiclook Cool

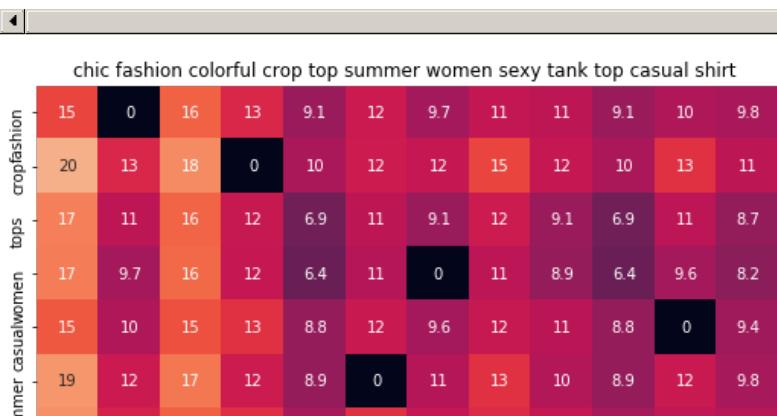
euclidean distance from input : 2.3134534

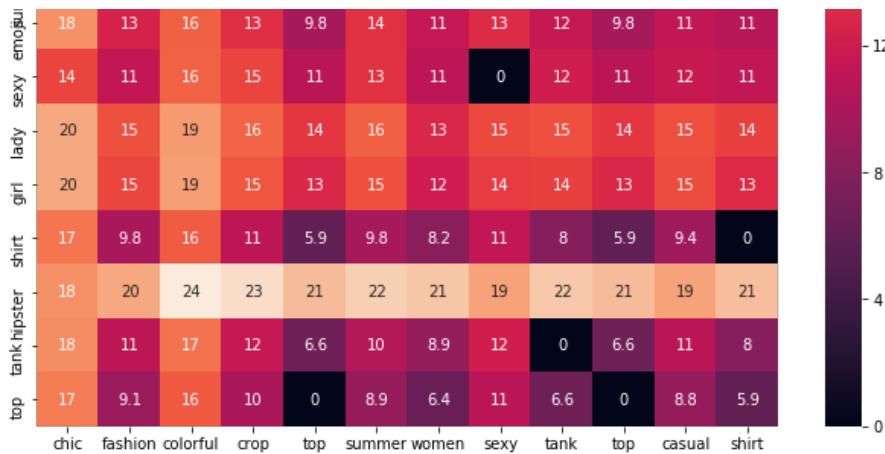


ASIN : B011RCJEMO

Brand : Chiclook Cool

euclidean distance from input : 2.3378801

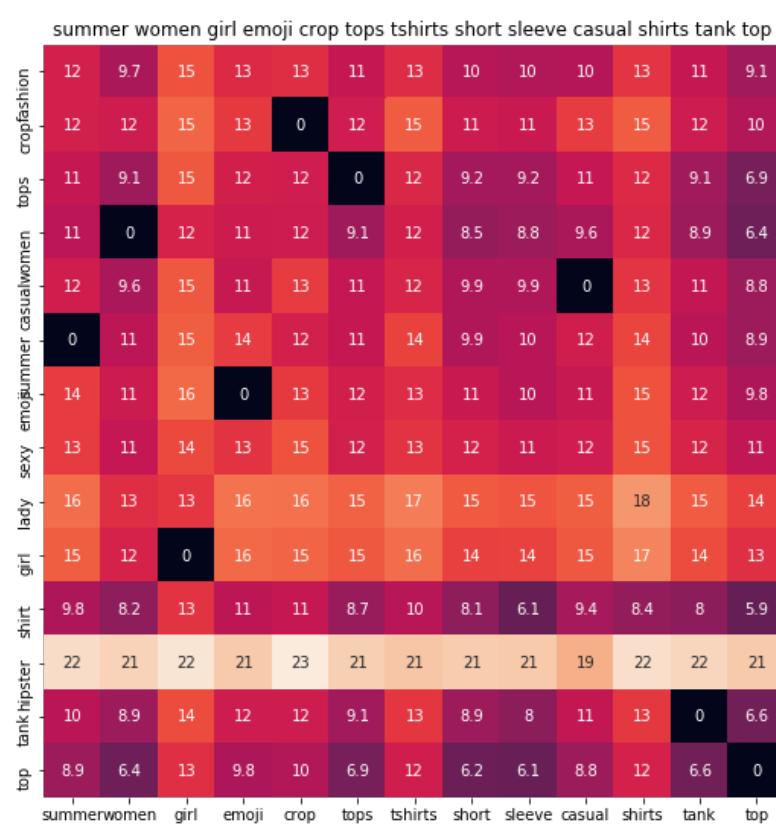




ASIN : B011RCJ6UE

Brand : Chiclook Cool

euclidean distance from input : 2.5151842



ASIN : B0124ECIU4

Brand : Doxi Supermall

euclidean distance from input : 2.5520852

[9.6] Weighted similarity using brand and color.

In [61]:

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

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

```

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

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

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

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

```

In [62]:

```

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

    # sentance1 : title1, input apparel
    # sentance2 : title2, recommended apparel
    # url: apparel image url
    # doc_id1: document id of input apparel

    # doc_id2: document id of recommended apparel
    # df_id1: index of document1 in the data frame
    # df_id2: index of document2 in the data frame
    # model: it can have two values, 1. avg 2. weighted

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

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

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

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

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

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

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

    # in last 25 * 10:15 grids we display image
    ax2 = plt.subplot(gs[:, 10:16])
    # we dont display grid lines 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 [63]:

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

    # pairwise_dist will store the distance from given input apparel to all remaining apparels
    # the metric we used here is cosine, the coside distance is mesured as  $K(X, Y) = \langle X, Y \rangle / (\|X\| * \|Y\|)$ 
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    idf_w2v_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))
    ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id])
    pairwise_dist = (w1 * idf_w2v_dist + w2 * ex_feat_dist)/float(w1 + w2)

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

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

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

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



ASIN : B010V3B44G

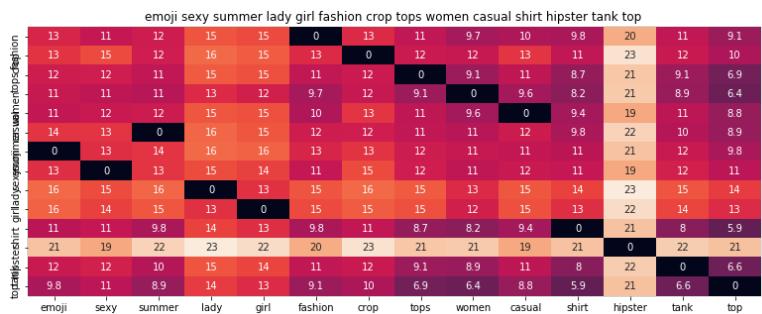
Brand : Doxi Supermall

euclidean distance from input : 0.0009765625

=====

=====

=====



ASIN : B010V3BLWQ

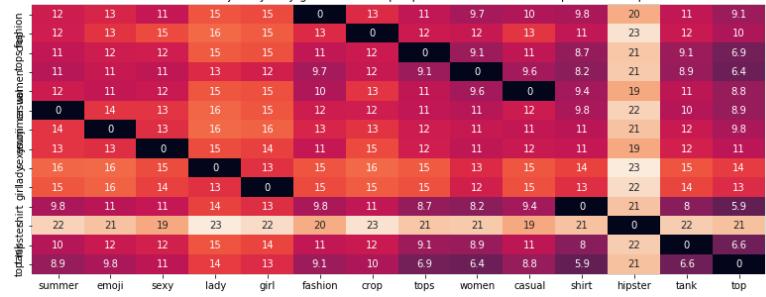
Brand : Doxi Supermall

euclidean distance from input : 0.0009765625

=====

=====

summer emoji sexy lady girl fashion crop tops women casual shirt hipster tank top



ASIN : B010V3BDII

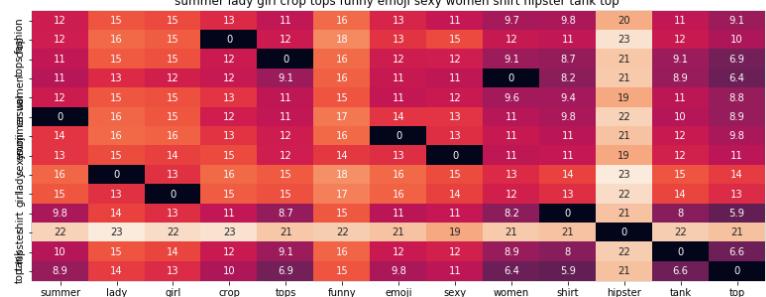
Brand : Doxi Supermall

euclidean distance from input : 0.0009765625

=====

=====

summer lady girl crop tops funny emoji sexy women shirt hipster tank top



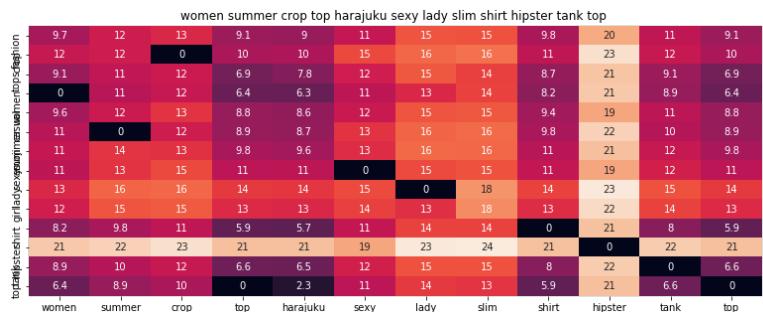
ASIN : B010V3BVMQ

Brand : Doxi Supermall

euclidean distance from input : 0.6765242099761963

=====

=====



ASIN : B010V3EDEE

Brand : Doxi Supermall

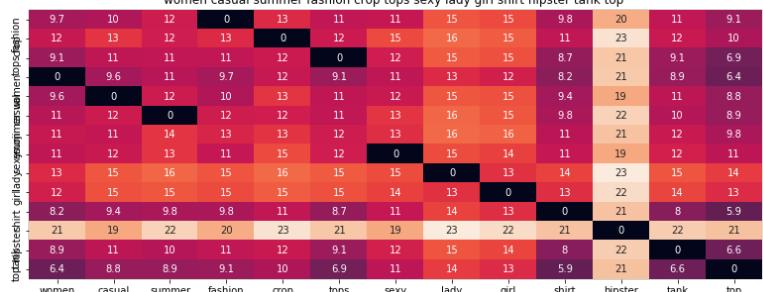
euclidean distance from input : 0.8717074394226074

=====

=====

=====

women casual summer fashion crop tops sexy lady girl shirt hipster tank top



ASIN : B010V3AYSS

Brand : Doxi Supermall

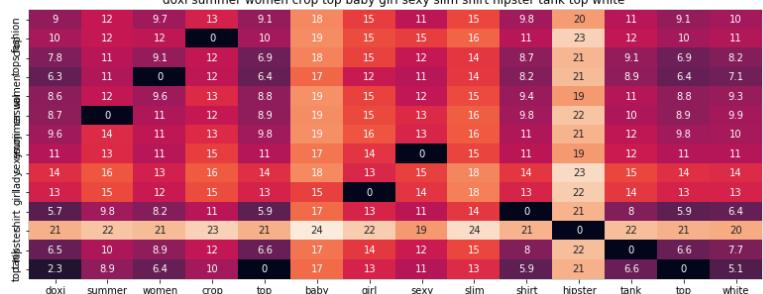
euclidean distance from input : 1.0552222015280392

=====

=====

=====

doxi summer women crop top baby girl sexy slim shirt hipster tank top white



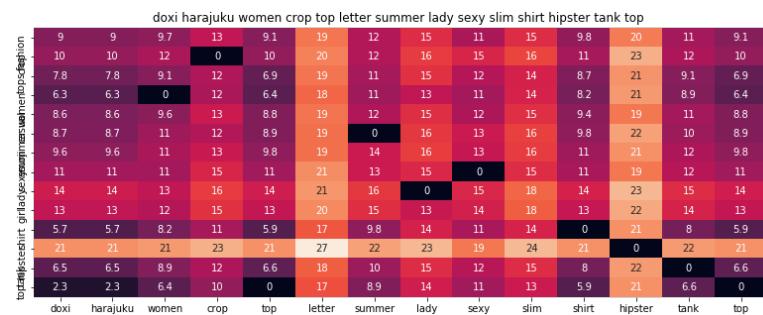
ASIN : B010V3A23U

Brand : Doxi Supermall

euclidean distance from input : 1.0621844291687013

=====

=====



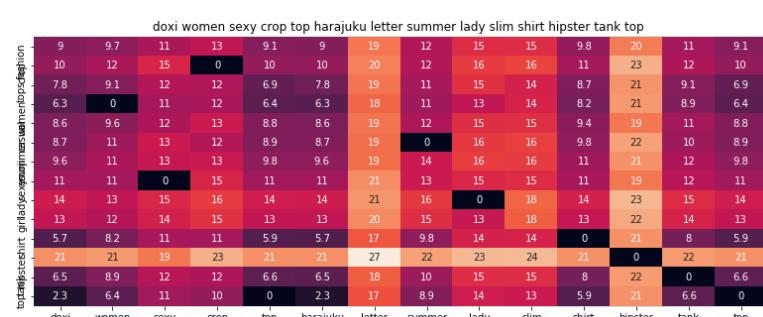
ASIN : B010V380LQ

Brand : Doxi Supermall

euclidean distance from input : 1.083746337890625

=====

=====



ASIN : B010V39146

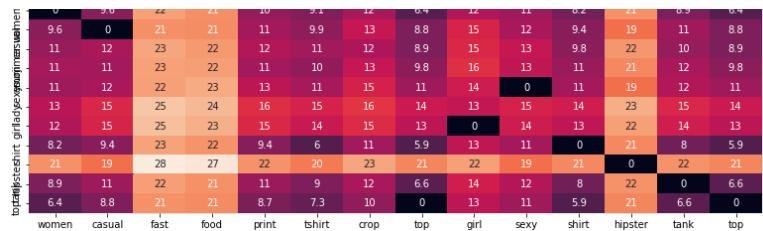
Brand : Doxi Supermall

euclidean distance from input : 1.083746337890625

=====

=====





ASIN : B010V3AB50

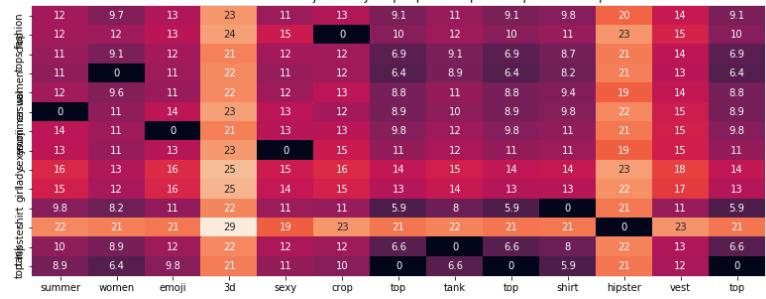
Brand : Doxi Supermall

euclidean distance from input : 1.3135896682739259

=====

=====

summer women emoji 3d sexy crop top tank top shirt hipster vest top



ASIN : B010V3E5EC

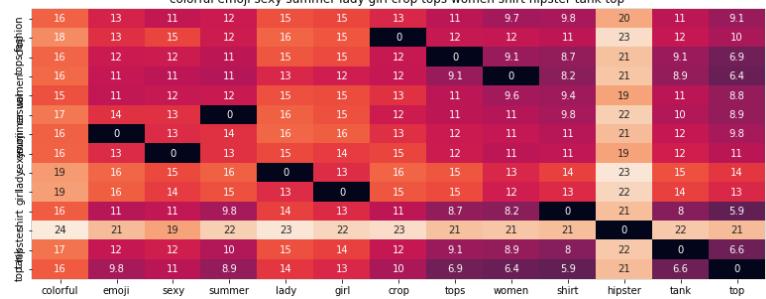
Brand : Doxi Supermall

euclidean distance from input : 1.3397350311279297

=====

=====

colorful emoji sexy summer lady girl crop tops women shirt hipster tank top



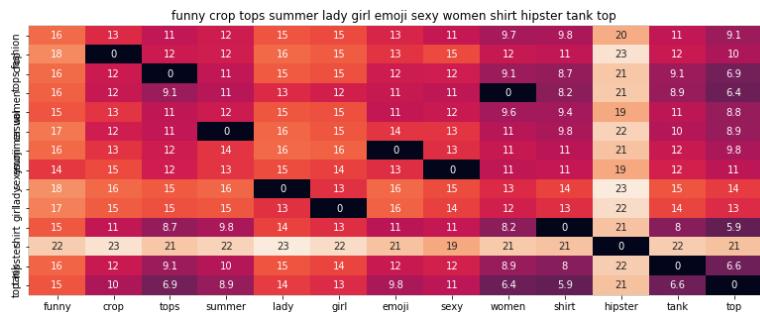
ASIN : B010V3BQZS

Brand : Doxi Supermall

euclidean distance from input : 1.3507425786871579

=====

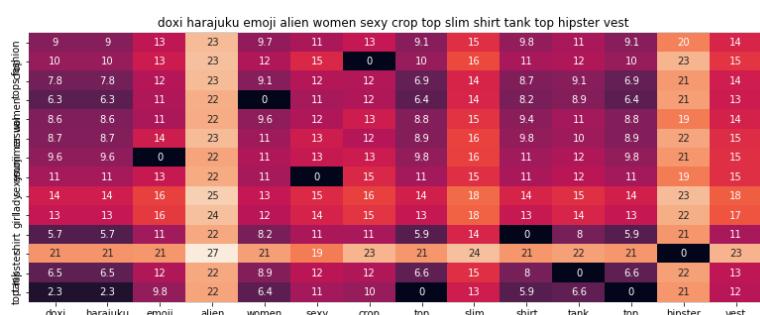
=====



ASIN : B010V3C116

Brand : Doxi Supermall

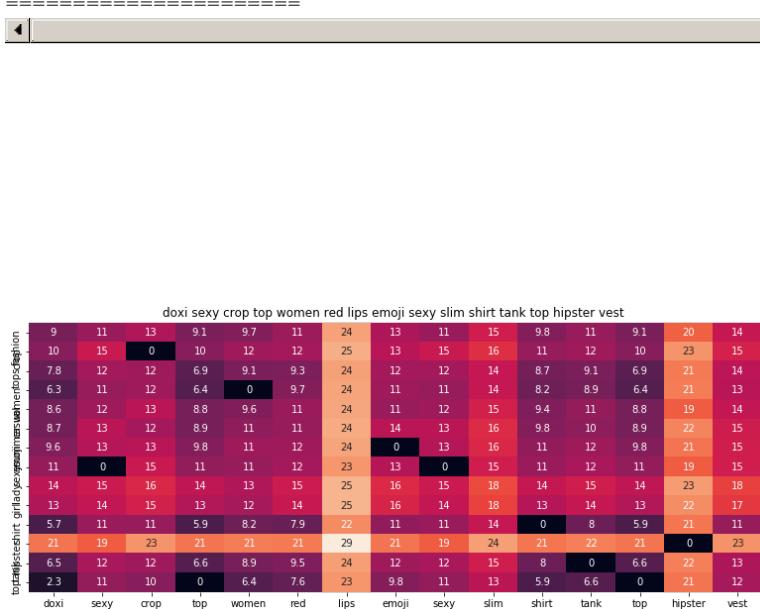
euclidean distance from input : 1.3836309911627438



ASIN : B010TKXAI4

Brand : Doxi Supermall

euclidean distance from input : 1.3876009941101075



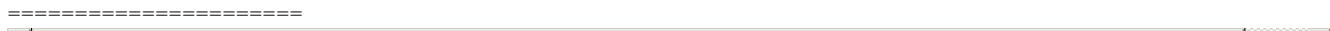
ASIN : B010TKXEHG

Brand : Doxi Supermall

=====

euclidean distance from input : 1.4269559860229493

=====



doxi women summer crop top yep letters sexy slim shirt hipster vest white tank top

doxi	women	summer	crop	top	yep	letters	sexy	slim	shirt	hipster	vest	white	tank	top
9	9.7	12	13	9.1	26	22	11	15	9.8	20	14	10	11	9.1
10	12	12	0	10	27	23	15	16	11	23	15	11	12	10
7.8	9.1	11	12	6.9	25	22	12	14	8.7	21	14	8.2	9.1	6.9
6.3	0	11	12	6.4	26	21	11	14	8.2	21	13	7.1	8.9	6.4
8.6	9.6	12	13	8.8	26	23	12	15	9.4	19	14	9.3	11	8.8
8.7	11	0	12	8.9	26	22	13	16	9.8	22	15	9.9	10	8.9
9.6	11	14	13	9.8	24	21	13	16	11	21	15	10	12	9.8
11	11	13	15	11	25	23	0	15	11	19	15	11	12	11
14	13	16	16	14	27	24	15	18	14	23	18	14	15	14
13	12	15	15	13	27	24	14	18	13	22	17	13	14	13
5.7	8.2	9.8	11	5.9	25	21	11	14	0	21	11	6.4	8	5.9
21	21	22	23	21	26	29	19	24	21	0	23	20	22	21
6.5	8.9	10	12	6.6	26	22	12	15	8	22	13	7.7	0	6.6
2.3	6.4	8.9	10	0	25	21	11	13	5.9	21	12	5.1	6.6	0



ASIN : B01OV3487Q

Brand : Doxi Supermall

euclidean distance from input : 1.4681867599487304

=====



doxi women cute emoji monkey crop top shirts sexy sleeveless tank tops girls

doxi	women	cute	emoji	monkey	crop	top	shirts	sexy	sleeveless	tank	tops	girls
9	9.7	15	13	22	13	9.1	13	11	11	11	11	14
10	12	17	13	22	0	10	15	15	14	12	12	16
7.8	9.1	15	12	21	12	6.9	12	12	12	9.1	0	14
6.3	0	15	11	21	12	6.4	12	11	11	8.9	9.1	10
8.6	9.6	14	11	22	13	8.8	13	12	12	11	11	14
8.7	11	16	14	22	12	8.9	14	13	13	10	11	15
9.6	11	14	0	20	13	9.8	15	13	14	12	12	16
11	11	11	13	21	15	11	15	0	12	12	12	14
14	13	17	16	21	16	14	18	15	16	15	15	15
13	12	15	16	20	15	13	17	14	16	14	15	11
5.7	8.2	14	11	19	11	5.9	8.4	11	9.5	8	8.7	14
21	21	20	21	27	23	21	22	19	22	22	21	23
6.5	8.9	15	12	20	12	6.6	13	12	11	0	9.1	14
2.3	6.4	14	9.8	20	10	0	12	11	11	6.6	6.9	13



ASIN : B01LF90QTO

Brand : Doxi Supermall

euclidean distance from input : 1.4761534690856934

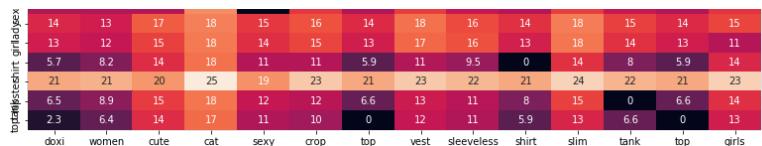
=====



doxi women cute cat sexy crop top vest sleeveless shirt slim tank top girls

doxi	women	cute	cat	sexy	crop	top	vest	sleeveless	shirt	slim	tank	top	girls
9	9.7	15	19	11	13	9.1	14	11	9.8	15	11	9.1	14
10	12	17	19	15	0	10	15	14	11	16	12	10	16
7.8	9.1	15	18	12	12	6.9	14	12	8.7	14	9.1	6.9	14
6.3	0	15	18	11	12	6.4	13	11	8.2	14	8.9	6.4	10
8.6	9.6	14	19	12	13	8.8	14	12	9.4	15	11	8.8	14
8.7	11	16	19	13	12	8.9	15	13	9.8	16	10	8.9	15
9.6	11	14	19	13	13	9.8	15	14	11	16	12	9.8	16
11	11	11	19	0	15	11	15	12	11	15	12	11	14





ASIN : B01LF90ROI

Brand : Doxi Supermall

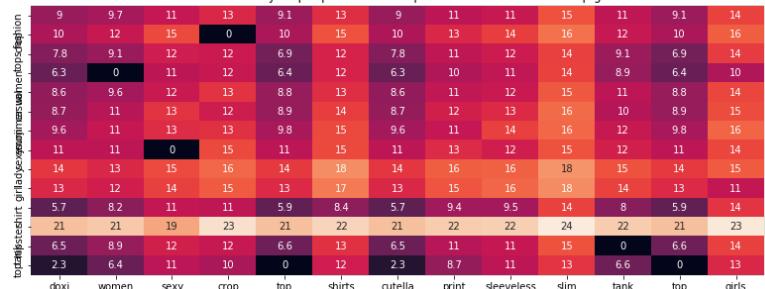
euclidean distance from input : 1.4821748733520508

=====

=====

=====

doxi women sexy crop top shirts cutella print sleeveless slim tank top girls



ASIN : B01LF90RKC

Brand : Doxi Supermall

euclidean distance from input : 1.5058938980102539

=====

=====

=====

doxi women 3cat sexy crop top sleeveless vest short shirts tank tops girls



ASIN : B01LY4GQY0

Brand : Doxi Supermall

euclidean distance from input : 1.5098180770874023

=====

=====

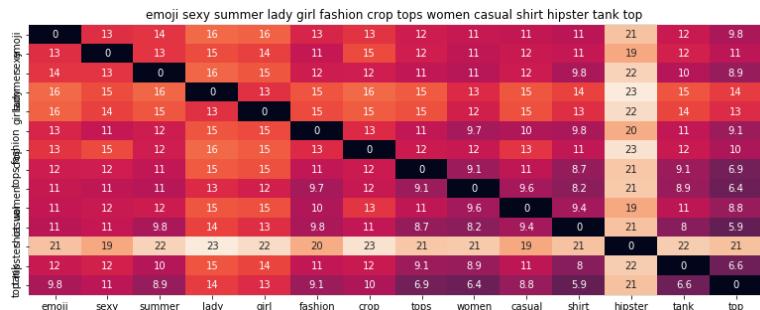
=====

=====

In [64]:

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

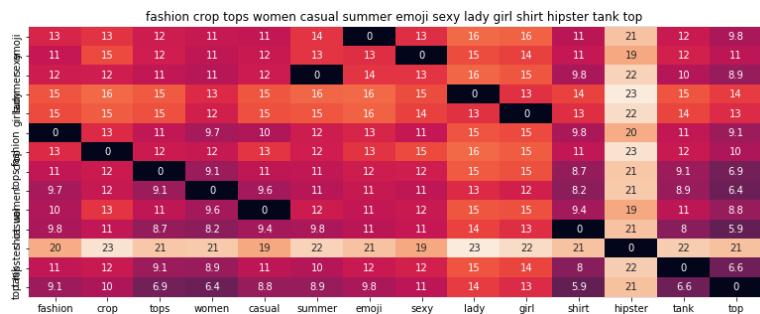
```
idf_w2v_brand(12566, 5, 50, 20)
```



ASIN : B010V3BLWQ

Brand : Doxi Supermall

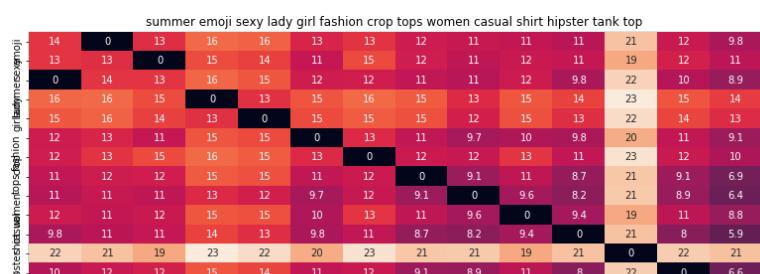
euclidean distance from input : 0.0001775568181818182

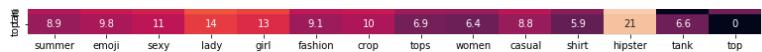


ASIN : B010V3B44G

Brand : Doxi Supermall

euclidean distance from input : 0.0001775568181818182

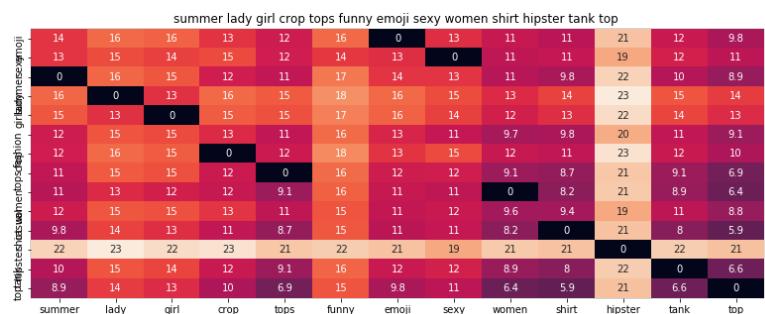




ASIN : B010V3BDII

Brand : Doxi Supermall

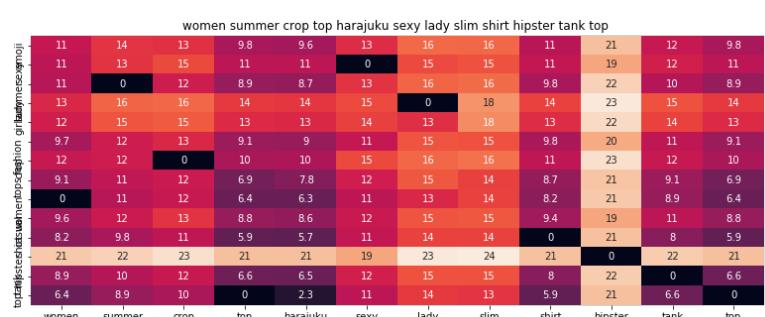
euclidean distance from input : 0.00017755681818182



ASIN : B010V3BVMQ

Brand : Doxi Supermall

euclidean distance from input : 0.12300440181385387

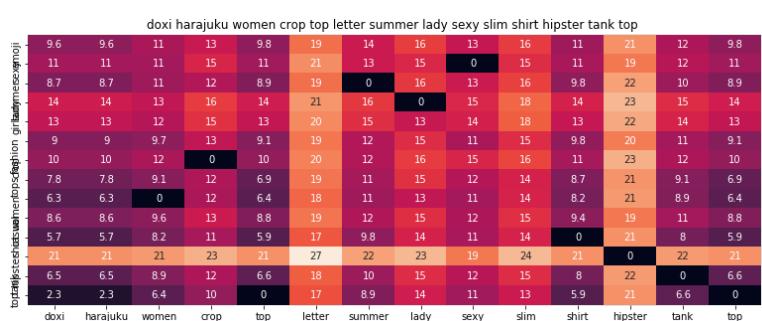
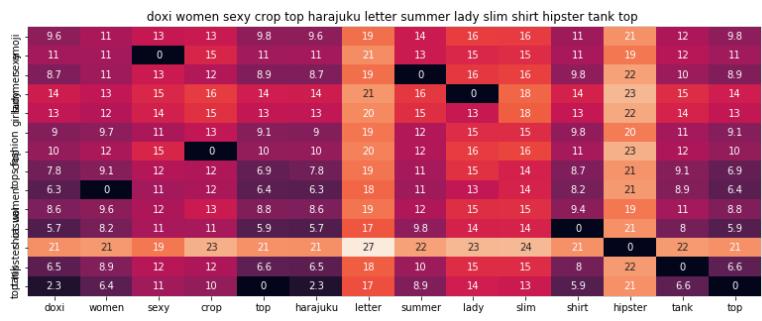
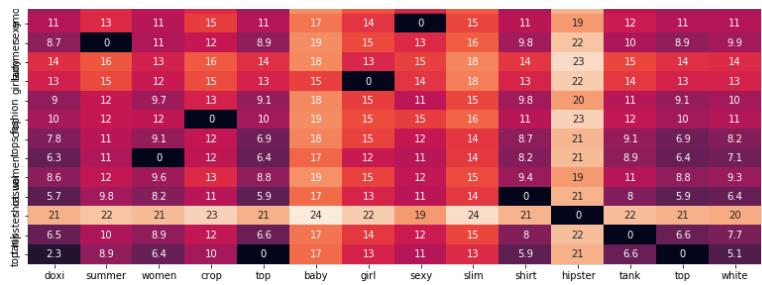


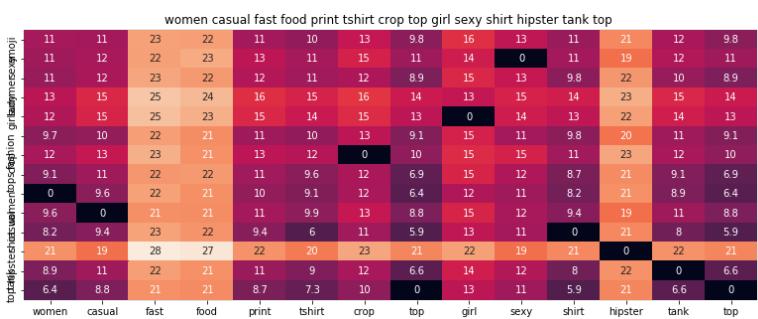
ASIN : B010V3EDEE

Brand : Doxi Supermall

euclidean distance from input : 0.15849226171320135



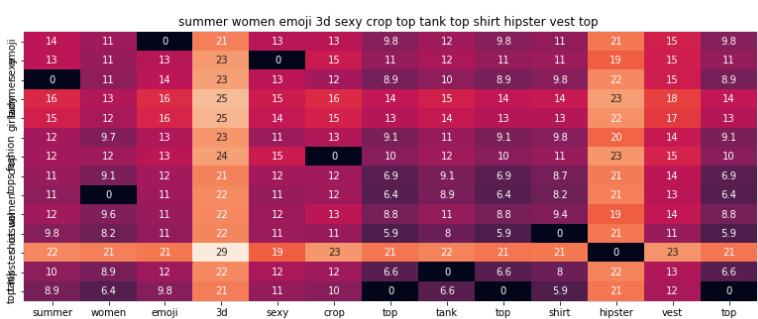




ASIN : B010V3AB50

Brand : Doxi Supermall

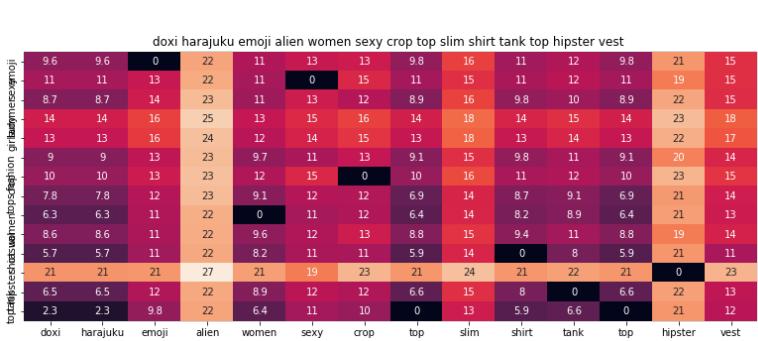
euclidean distance from input : 0.2388344851407138



ASIN : B010V3E5EC

Brand : Doxi Supermall

euclidean distance from input : 0.2435881874778054



ASIN : B010TKXAI4

Brand : Doxi Supermall

euclidean distance from input : 0.25229108983820137

=====



doxi sexy crop top women red lips emoji sexy slim shirt tank top hipster vest

doxi	sexy	crop	top	women	red	lips	emoji	sexy	slim	shirt	tank	top	hipster	vest
9.6	13	13	9.8	11	12	24	0	13	16	11	12	9.8	21	15
11	0	15	11	11	12	23	13	0	15	11	12	11	19	15
8.7	13	12	8.9	11	11	24	14	13	16	9.8	10	8.9	22	15
14	15	16	14	13	15	25	16	15	18	14	15	14	23	18
13	14	15	13	12	14	25	16	14	18	13	14	13	22	17
9	11	13	9.1	9.7	11	24	13	11	15	9.8	11	9.1	20	14
10	15	0	10	12	12	25	13	15	16	11	12	10	23	15
7.8	12	12	6.9	9.1	9.3	24	12	12	14	8.7	9.1	6.9	21	14
6.3	11	12	6.4	0	9.7	24	11	11	14	8.2	8.9	6.4	21	13
8.6	12	13	8.8	9.6	11	24	11	12	15	9.4	11	8.8	19	14
5.7	11	11	5.9	8.2	7.9	22	11	11	14	0	8	5.9	21	11
21	19	23	21	21	21	29	21	19	24	21	22	21	0	23
6.5	12	12	6.6	8.9	9.5	24	12	12	15	8	0	6.6	22	13
2.3	11	10	0	6.4	7.6	23	9.8	11	13	5.9	6.6	0	21	12



ASIN : B010TKXEHG

Brand : Doxi Supermall

euclidean distance from input : 0.25944654291326347

=====



doxi women summer crop top yep letters sexy slim shirt hipster vest white tank top

doxi	women	summer	crop	top	yep	letters	sexy	slim	shirt	hipster	vest	white	tank	top
9.6	11	14	13	9.8	24	21	13	16	11	21	15	10	12	9.8
11	11	13	15	11	25	23	0	15	11	19	15	11	12	11
8.7	11	0	12	8.9	26	22	13	16	9.8	22	15	9.9	10	8.9
14	13	16	16	14	27	24	15	18	14	23	18	14	15	14
13	12	15	15	13	27	24	14	18	13	22	17	13	14	13
9	9.7	12	13	9.1	26	22	11	15	9.8	20	14	10	11	9.1
10	12	12	0	10	27	23	15	16	11	23	15	11	12	10
7.8	9.1	11	12	6.9	25	22	12	14	8.7	21	14	8.2	9.1	6.9
6.3	0	11	12	6.4	26	21	11	14	8.2	21	13	7.1	8.9	6.4
8.6	9.6	12	13	8.8	26	23	12	15	9.4	19	14	9.3	11	8.8
5.7	8.2	9.8	11	5.9	25	21	11	14	0	21	11	6.4	8	5.9
21	21	22	23	21	26	29	19	24	21	0	23	20	22	21
6.5	8.9	10	12	6.6	26	22	12	15	8	22	13	7.7	0	6.6
2.3	6.4	8.9	10	0	25	21	11	13	5.9	21	12	5.1	6.6	0



ASIN : B010V3487Q

Brand : Doxi Supermall

euclidean distance from input : 0.26694304726340556

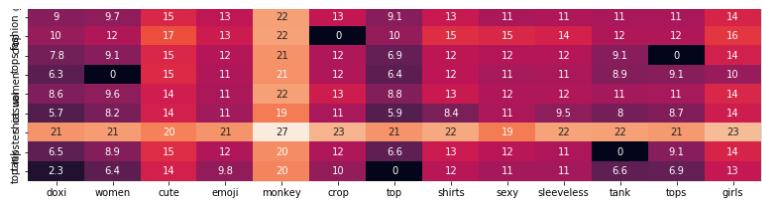
=====



doxi women cute emoji monkey crop top shirts sexy sleeveless tank tops girls

doxi	women	cute	emoji	monkey	crop	top	shirts	sexy	sleeveless	tank	tops	girls
9.6	11	14	0	20	13	9.8	15	13	14	12	12	16
11	11	11	13	21	15	11	15	0	12	12	12	14
8.7	11	16	14	22	12	8.9	14	13	13	10	11	15
14	13	17	16	21	16	14	18	15	16	15	15	15
13	12	15	16	20	15	13	17	14	16	14	15	11





ASIN : B01LF90QTO

Brand : Doxi Supermall

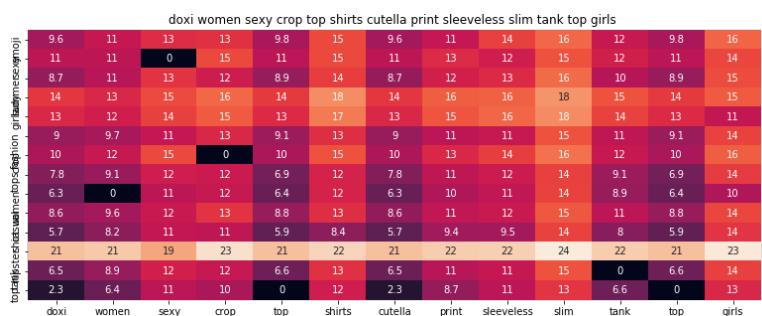
euclidean distance from input : 0.2683915398337624



ASIN : B01LF90QTO

Brand : Doxi Supermall

euclidean distance from input : 0.2694863406094638

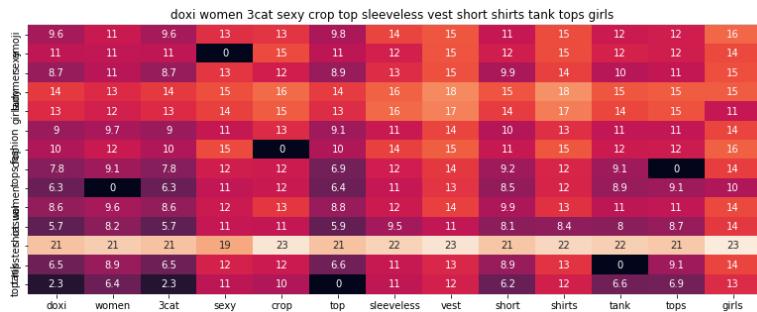


ASIN : B01LF90RKC

Brand : Doxi Supermall

euclidean distance from input : 0.2737988905473189

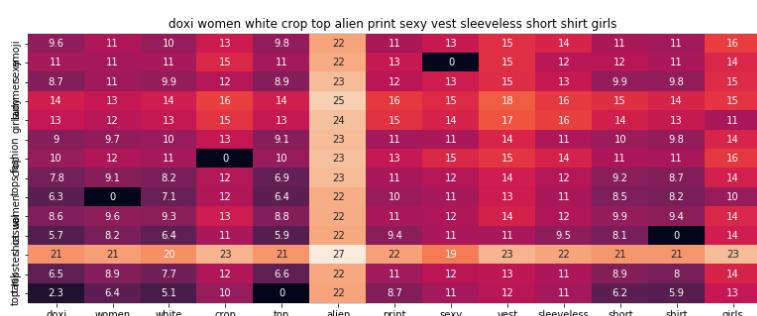




ASIN : B01LY4GQY0

Brand : Doxi Supermall

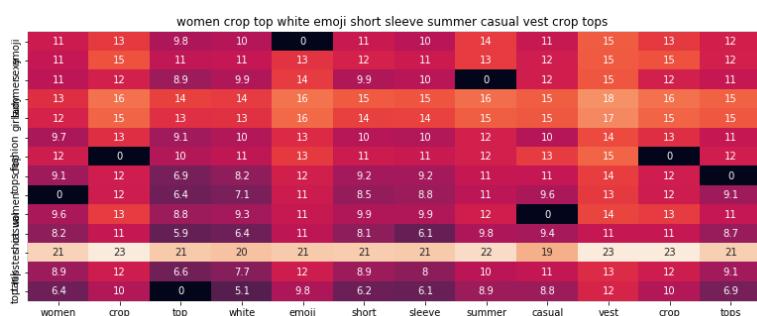
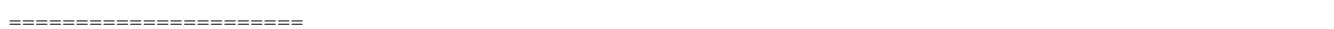
euclidean distance from input : 0.27451237765225495



ASIN : B01LF90SCY

Brand : Doxi Supermall

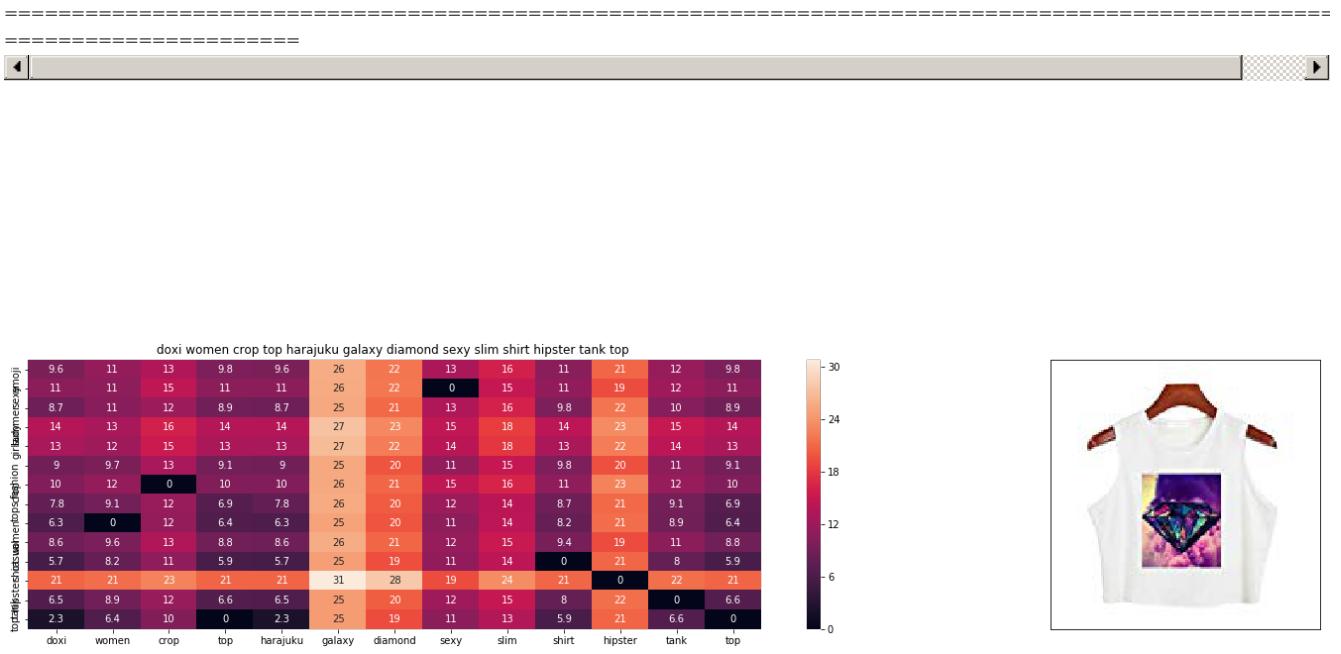
euclidean distance from input : 0.2815914500843395



ASIN : B0124E7MHS

Brand : Doxi Supermall

euclidean distance from input : 0.2871342745694247



ASIN : B01OV39EUM
 Brand : Doxi Supermall
 euclidean distance from input : 0.3081340443004261

=====

=====

[10.2] Keras and Tensorflow to extract features

In [15]:

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

Using TensorFlow backend.

In [4]:

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

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

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

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

# each image is converted into 25088 length dense-vector

'''

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

```

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

def save_bottlebeck_features():
    #Function to compute VGG-16 CNN for image feature extraction.

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

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

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

    bottleneck_features_train = model.predict_generator(generator, nb_train_samples // batch_size)
    bottleneck_features_train = bottleneck_features_train.reshape((16042,25088))

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

save_bottlebeck_features()
'''
```

Out [4]:

```

"\n# dimensions of our images.\nimg_width, img_height = 224, 224\n\ntop_model_weights_path = 'bottleneck_fc_model.h5'\ntrain_data_dir = 'images2/'\nnb_train_samples = 16042\nepochs = 50\nbatch_size = 1\n\ndef save_bottlebeck_features():\n    #Function to compute VGG-16 CNN for image feature extraction.\n    asins = []\n    datagen = ImageDataGenerator(rescale=1. / 255)\n    # build the VGG16 network\n    model = applications.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    for i in generator.filenames:\n        asins.append(i[2:-5])\n\n    bottleneck_features_train = model.predict_generator(generator, nb_train_samples // batch_size)\n    bottleneck_features_train = bottleneck_features_train.reshape((16042,25088))\n\n    np.save(open('16k_data_cnn_features.npy', 'wb'), bottleneck_features_train)\n    np.save(open('16k_data_cnn_feature_asins.npy', 'wb'), np.array(asins))\n\n\nsave_bottlebeck_features()\n"
```

[10.3] Visual features based product similarity.

In [5]:

```

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

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

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

#get similar products using CNN features (VGG-16)
def get_similar_products_cnn(doc_id, num_results):
    doc_id = asins.index(df_asins[doc_id])
    pairwise_dist = pairwise_distances(bottleneck_features_train, bottleneck_features_train[doc_id]
```

```

.reshape(1,-1))

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

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

```

get_similar_products_cnn(12566, 20)



Product Title: fashion crop tops women casual summer emoji sexy lady girl shirt hipster tank top
 Euclidean Distance from input image: 0.044194173
 Amazon Url: www.amazon.com/dp/B010V3B44G



Product Title: chic women harajuku fashion funny emoji sexy crop tops sleeveless vest shirt
 Euclidean Distance from input image: 18.58574
 Amazon Url: www.amazon.com/dp/B011RCJNL6



Product Title: women quotes boys print white sleeveless crop top
 Euclidean Distance from input image: 22.04554
 Amazon Url: www.amazon.com/dp/B0748CKWF3



Product Title: women forever young print white sleeveless crop top
 Euclidean Distance from input image: 22.321054
 Amazon Url: www.amazon.com/dp/B0748CDWPH



Product Title: kingde self confident stretch vest tshirt suspendersbqn46
Euclidean Distance from input image: 22.724638
Amazon Url: www.amazon.com/dp/B015H2R3SC



Product Title: women three wise monkeys emoji print sleeveless crop top
Euclidean Distance from input image: 24.4526
Amazon Url: www.amazon.com/dp/B074VPC98H



Product Title: women infinity books print white sleeveless crop top
Euclidean Distance from input image: 25.067322
Amazon Url: www.amazon.com/dp/B074TYKHPL



Product Title: ladies crop top emoji women never grow shirt vest style tank tops
Euclidean Distance from input image: 26.787249
Amazon Url: www.amazon.com/dp/B013SPUM7O



Product Title: women fashion giant pink donut printed white sleeveless crop top
Euclidean Distance from input image: 28.364716
Amazon Url: www.amazon.com/dp/B017Q4BOCA



Product Title: women keep swimming print sleeveless crop top
Euclidean Distance from input image: 29.022251
Amazon Url: www.amazon.com/dp/B0749CCCY4



Product Title: women summer crop top harajuku sexy lady slim shirt hipster tank top
Euclidean Distance from input image: 29.029215
Amazon Url: www.amazon.com/dp/B010V3EDEE



Product Title: women fashion wanna hug printed white sleeveless crop top
Euclidean Distance from input image: 29.049519
Amazon Url: www.amazon.com/dp/B017R0CDOQ



Product Title: summer emoji sexy lady girl fashion crop tops women casual shirt hipster tank top
Euclidean Distance from input image: 29.482334
Amazon Url: www.amazon.com/dp/B010V3BDII



Product Title: women fashion love food printed white crop top
Euclidean Distance from input image: 29.615324
Amazon Url: www.amazon.com/dp/B017K0DQZ8



Product Title: kingde slim starbucks coffee small vest tshirt sling stretchbqn53
Euclidean Distance from input image: 29.690033
Amazon Url: www.amazon.com/dp/B015H2QWY8



Product Title: women fashion cute word printed name barbie sleeveless crop top
Euclidean Distance from input image: 29.862297
Amazon Url: www.amazon.com/dp/B01AK8SC10



Product Title: women fashion best friends printed best pattern one sleeveless crop top
Euclidean Distance from input image: 29.954874
Amazon Url: www.amazon.com/dp/B01GFJOGUE



Product Title: women yabish print white sleeveless crop top
Euclidean Distance from input image: 30.084402
Amazon Url: www.amazon.com/dp/B0748JNFL9



Product Title: women fashion cool quote printed sleeveless crop top
Euclidean Distance from input image: 30.10058
Amazon Url: www.amazon.com/dp/B018RVK1FM



Product Title: women pattern 8 cute baby alien print sleeveless crop top
Euclidean Distance from input image: 30.180464
Amazon Url: www.amazon.com/dp/B074BNJM8S

As cnn features and data have different shapes, Using asins , i am making sure that both the data have same shapes

In [7]:

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

In [8]:

```
bottleneck_features_train1 = []
for i in range(len(asins)):
    if asins[i] in df['asin'].values:
        bottleneck_features_train1.append(bottleneck_features_train[i])
```

In [9]:

```
bottleneck_features_train1 = np.asarray(bottleneck_features_train1)
```

In [11]:

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

# load the original 16K dataset into data_16k_preprocessed
data = pd.read_pickle('pickels/16k_apperial_data_preprocessed')
```

Build a weighted Nearest neighbor model using Visual, Text, Brand and Color

In [12]:

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

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

In [13]:

```
def build_avg_vec(sentence, num_features, doc_id, m_name):
    # sentence: its title of the apparel
    # num_features: the length of word2vec vector, its values = 300
    # m_name: model information it will take two values
        # if m_name == 'ava'. we will append the model's 1. w2v representation of word i
```

```

    " + m_name + avg , we will append the modality, avg representation of word + 
    # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)

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

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

```

In [14]:

```

from datetime import datetime
start = datetime.now()

# *****Data Modifications*****

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

# Replace spaces with hyphen
brands = [x.replace(" ", "-") for x in df['brand'].values]
colors = [x.replace(" ", "-") for x in df['color'].values]

# *****End of Data Modifications*****


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

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

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

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


brand_vectorizer = CountVectorizer()
color_vectorizer = CountVectorizer()
vocab = model.keys()
doc_id = 0
title_features = []

# for every title we build a weighted vector representation
for i in df['title']:
    title_features.append(build_avg_vec(i, 300, doc_id, 'weighted'))
    doc_id += 1

# Creating Title Vector

title_features = np.array(title_features)

# Creating Brand Vector

brand_features = brand_vectorizer.fit_transform(brands)

```

```

# Creating Color Vector

color_features = color_vectorizer.fit_transform(colors)

# Creating Image Vector
# Load the features using CNN features (VGG-16)

image_features = bottleneck_features_train1

print("\nTime Taken: ", datetime.now() - start)

```

Time Taken: 0:03:16.347713

In [36]:

```

# Utility functions

def get_word_vec(sentence, doc_id, m_name):
    # sentence : title of the apparel
    # doc_id: document id in our corpus
    # m_name: model information it will take two values
        # if m_name == 'avg', we will append the model[i], w2v representation of word i
        # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)
    vec = []
    for i in sentence.split():
        if i in vocab:
            if m_name == 'weighted' and i in idf_title_vectorizer.vocabulary_:
                vec.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[i]] * model[i])
            elif m_name == 'avg':
                vec.append(model[i])
        else:
            # if the word in our corpus is not there in the google word2vec corpus, we are just
            # ignoring it
            vec.append(np.zeros(shape=(300,)))
    # we will return a numpy array of shape (#number of words in title * 300 ) 300 =
    len(w2v_model[word])
    # each row represents the word2vec representation of each word (weighted/avg) in given
    sentance
    return np.array(vec)

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

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

```

In [45]:

```
weighted_final(12566,20,99,1,1,1)
```





ASIN : B06Y3B8FC4

TITLE : lovely souls clothing womens bottle caps raglan chocolate ivory rayon shirt medium

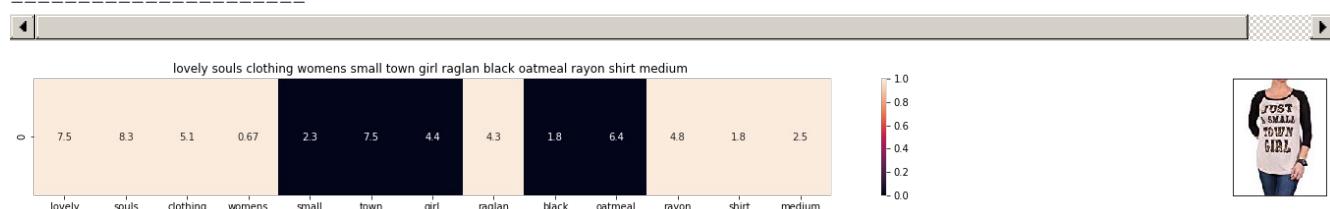
BRAND : Lovely Souls Clothing

COLOR : Chocolate Ivory

Euclidean Distance from input is : 0.0

=====

=====



ASIN : B06Y3CPXHK

TITLE : lovely souls clothing womens small town girl raglan black oatmeal rayon shirt medium

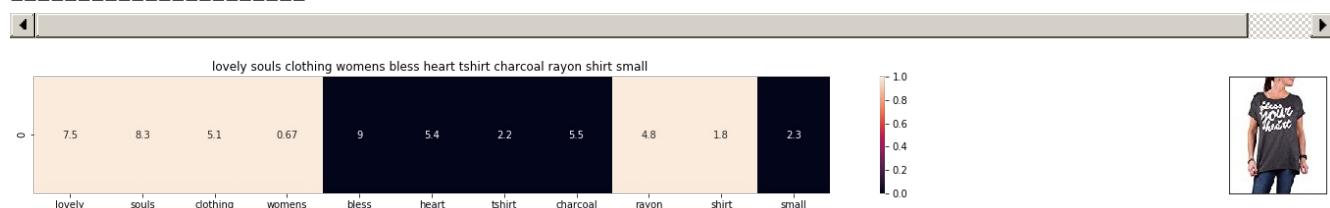
BRAND : Lovely Souls Clothing

COLOR : Black Oatmeal

Euclidean Distance from input is : 4.893377416274127

=====

=====



ASIN : B06XZN371Z

TITLE : lovely souls clothing womens bless heart tshirt charcoal rayon shirt small

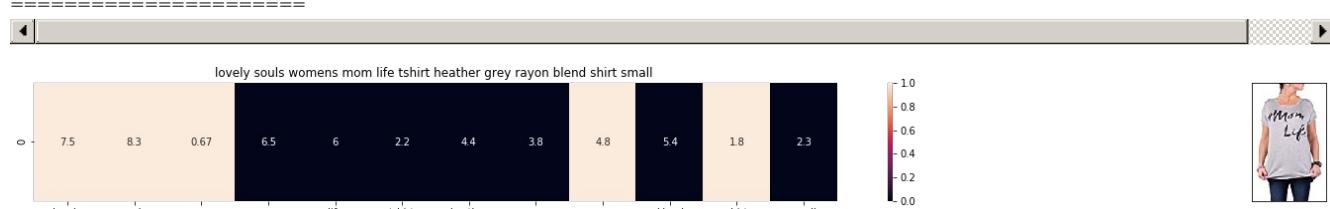
BRAND : Lovely Souls Clothing

COLOR : Charcoal

Euclidean Distance from input is : 5.538014981396968

=====

=====





ASIN : B06Y3CDCYN

TITLE : lovely souls womens mom life tshirt heather grey rayon blend shirt small

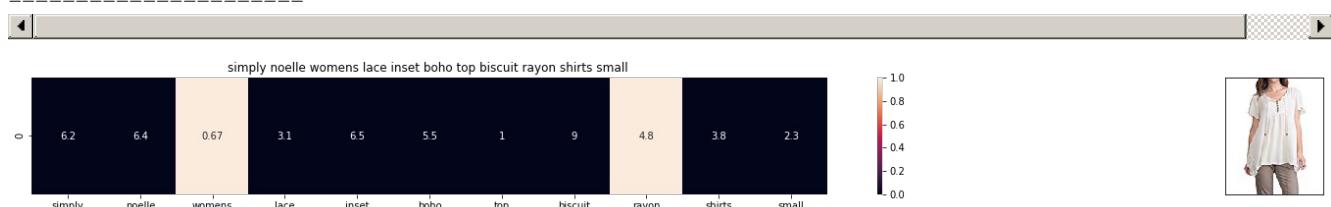
BRAND : Lovely Souls

COLOR : Heather Grey

Euclidean Distance from input is : 5.540706858915441

=====

=====



ASIN : B06XBRHYQP

TITLE : simply noelle womens lace inset boho top biscuit rayon shirts small

BRAND : Simply Noelle

COLOR : Biscuit

Euclidean Distance from input is : 5.692389377174865

=====

=====



ASIN : B07522R7DH

TITLE : sacred threads beautiful rayon brown cap sleeve top tunic small

BRAND : Sacred Threads

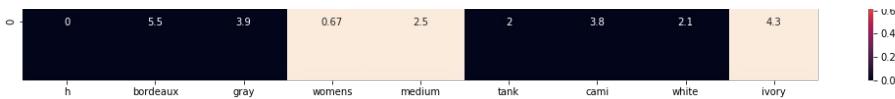
COLOR : Colors: Brown-tan , Gold, Maroon

Euclidean Distance from input is : 5.710554595482824

=====

=====





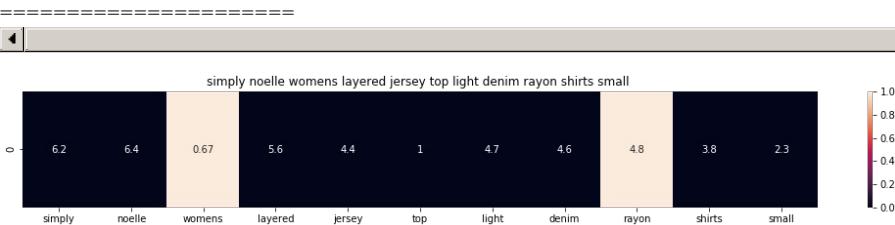
ASIN : B074MJH44S

TITLE : h bordeaux gray womens medium tank cami white ivory

BRAND : H By Bordeaux

COLOR : White Ivory

Euclidean Distance from input is : 5.876439090952722



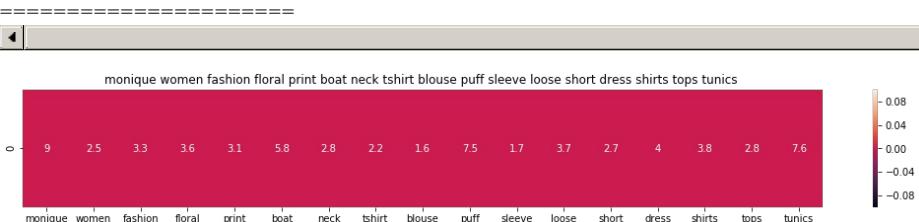
ASIN : B06XYVFMMD

TITLE : simply noelle womens layered jersey top light denim rayon shirts small

BRAND : Simply Noelle

COLOR : Light Denim

Euclidean Distance from input is : 5.921418986746609



ASIN : B0749LD5NL

TITLE : monique women fashion floral print boat neck tshirt blouse puff sleeve loose short dress shirts tops tunics

BRAND : Monique

COLOR : Flower

Euclidean Distance from input is : 5.928301894502053



ASIN : B06XFWSQS1

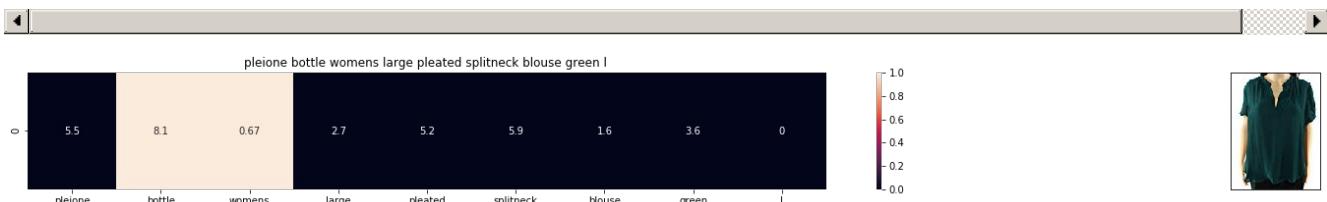
TITLE : tosangnclothes sexy lady women cashmere gold lace sexy tank top camis blouse vest l

BRAND : Tosangn_Clothes

COLOR : Black

Euclidean Distance from input is : 5.950395162503835

=====



ASIN : B0751JYXLG

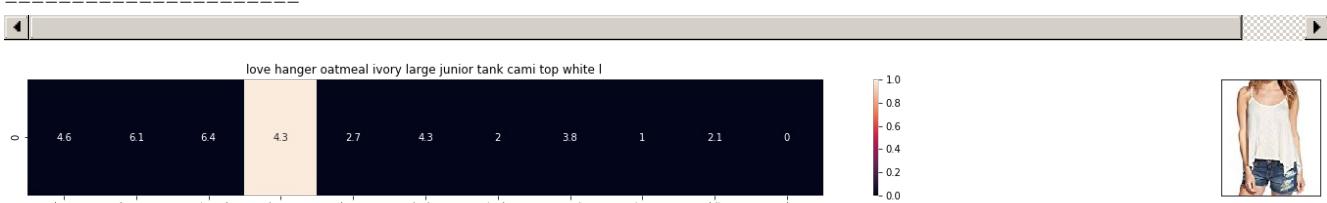
TITLE : pleione bottle womens large pleated splitneck blouse green l

BRAND : Pleione

COLOR : Green

Euclidean Distance from input is : 5.966483087236716

=====



ASIN : B01LOT4A6Y

TITLE : love hanger oatmeal ivory large junior tank cami top white l

=====

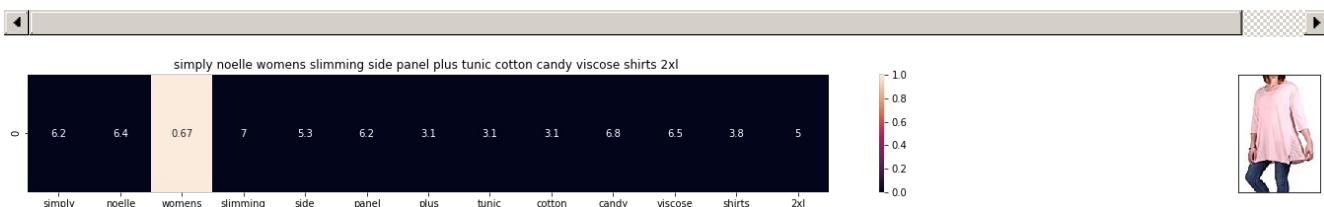
BRAND : Love on a Hanger

COLOR : White

Euclidean Distance from input is : 5.971851901111658

=====

=====



ASIN : B06Y3NVFKY

TITLE : simply noelle womens slimming side panel plus tunic cotton candy viscose shirts 2xl

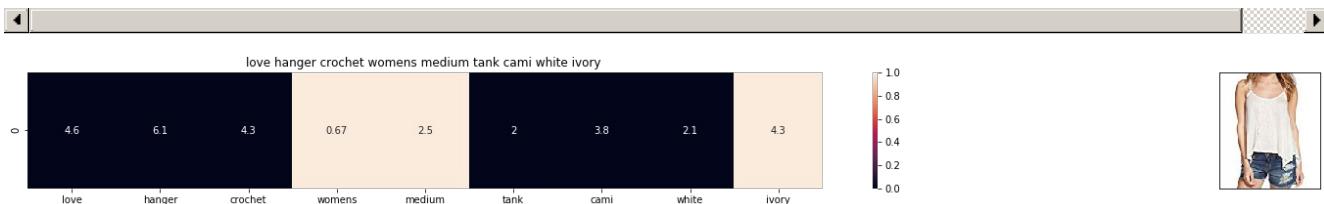
BRAND : Simply Noelle

COLOR : Cotton Candy

Euclidean Distance from input is : 5.973417237371322

=====

=====



ASIN : B07175LXDS

TITLE : love hanger crochet womens medium tank cami white ivory

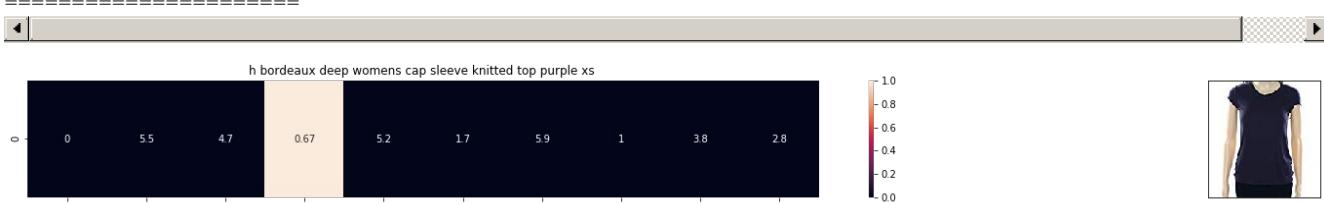
BRAND : Love on a Hanger

COLOR : White Ivory

Euclidean Distance from input is : 5.981221540833525

=====

=====



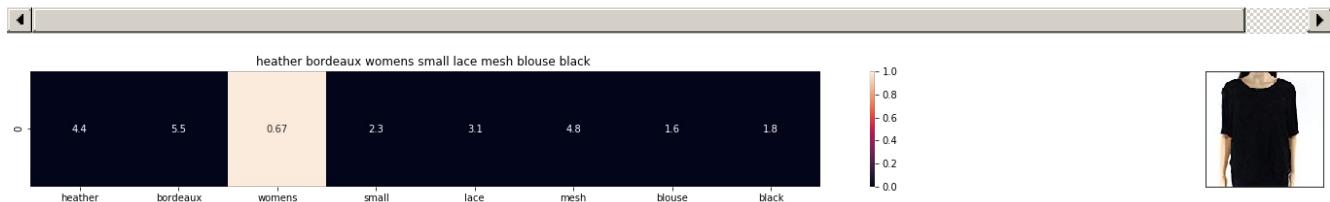
ASIN : B0722ZGM5C

TITLE : h bordeaux deep womens cap sleeve knitted top purple xs

BRAND : H By Bordeaux

COLOR : Purple

Euclidean Distance from input is : 6.020671191151677



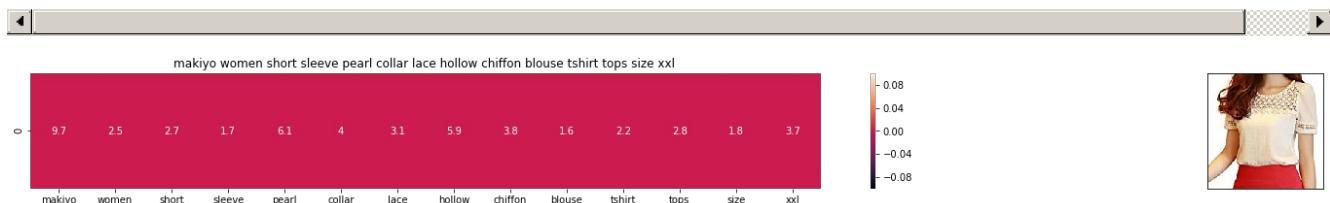
ASIN : B074N9NJ4Z

TITLE : heather bordeaux womens small lace mesh blouse black

BRAND : Heather by Bordeaux

COLOR : Black

Euclidean Distance from input is : 6.035862072440932



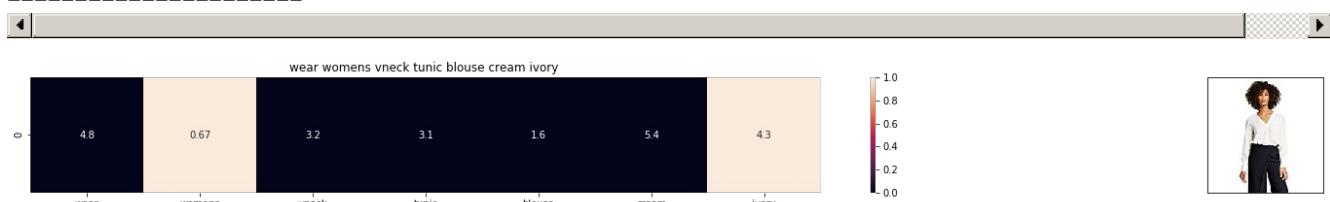
ASIN : B012SQWPYC

TITLE : makiyo women short sleeve pearl collar lace hollow chiffon blouse tshirt tops size xxl

BRAND : MAKIYO

COLOR : White

Euclidean Distance from input is : 6.039750332061892





ASIN : B01N0R90TU

TITLE : wear womens vneck tunic blouse cream ivory

BRAND : Who What Wear

COLOR : Cream

Euclidean Distance from input is : 6.043846645206806

=====

=====



ASIN : B06XHPTGJ1

TITLE : women lightweight cotton pullover hoodie ettellut hair band one size turquoise sofra tube bra

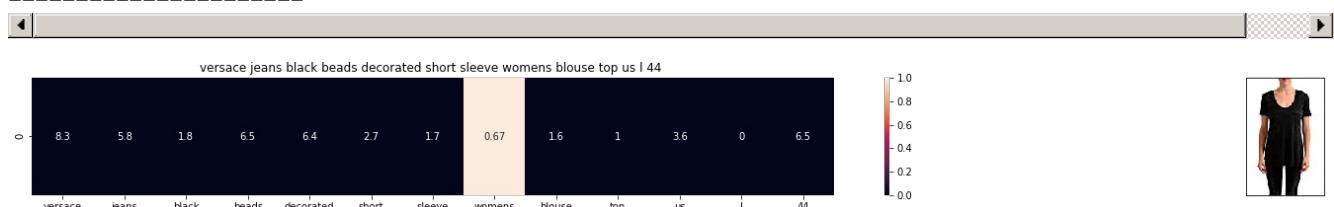
BRAND : Sofra

COLOR : Turquoise With Sofra Tube Bra

Euclidean Distance from input is : 6.045453906776866

=====

=====



ASIN : B01LI142F8

TITLE : versace jeans black beads decorated short sleeve womens blouse top us l 44

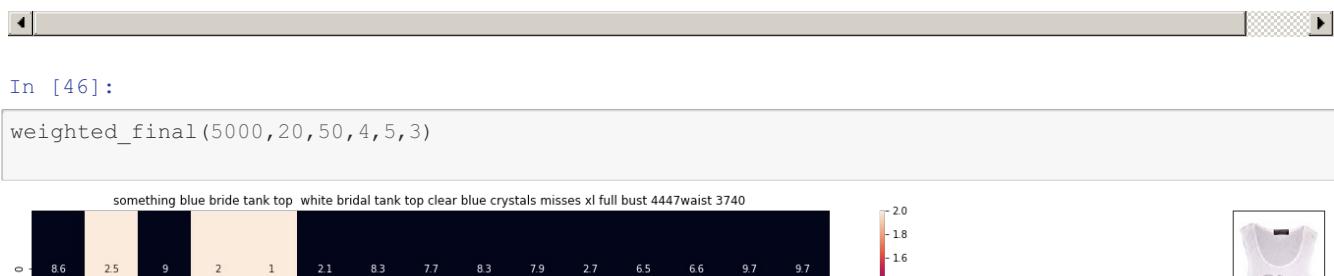
BRAND : Versace Jeans

COLOR : Black

Euclidean Distance from input is : 6.048664860848559

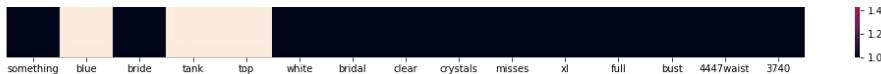
=====

=====



In [46]:

```
weighted_final(5000,20,50,4,5,3)
```



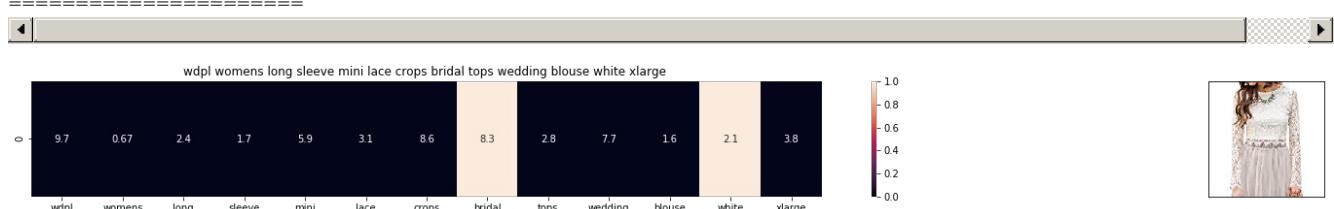
ASIN : B00I55DAEA

TITLE : something blue bride tank top white bridal tank top clear blue crystals misses xl full bust 4447waist 3740

BRAND : Girl ExtraOrdinaire

COLOR : White

Euclidean Distance from input is : 0.00354618122500758



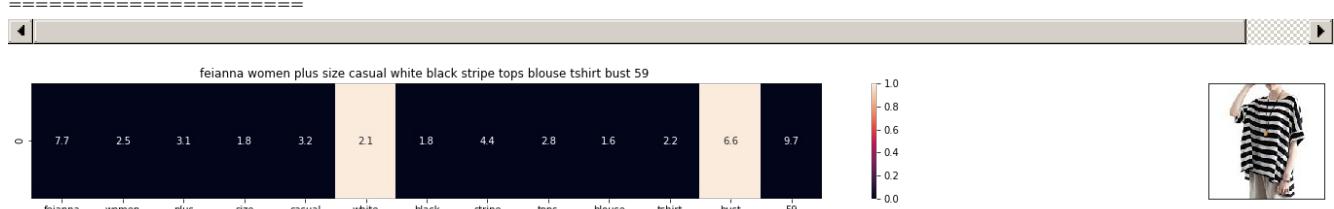
ASIN : B01MFERHO4

TITLE : wdpl womens long sleeve mini lace crops bridal tops wedding blouse white xlarge

BRAND : WDPL

COLOR : White

Euclidean Distance from input is : 5.485937728118732



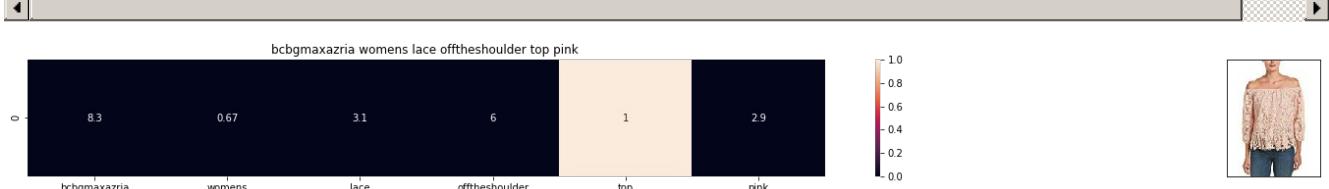
ASIN : B072WQ86QJ

TITLE : feianna women plus size casual white black stripe tops blouse tshirt bust 59

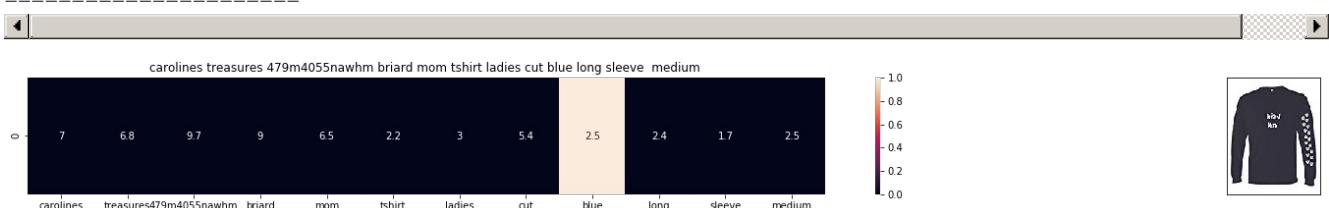
BRAND : FEIANNAA

COLOR : White Black Stripe

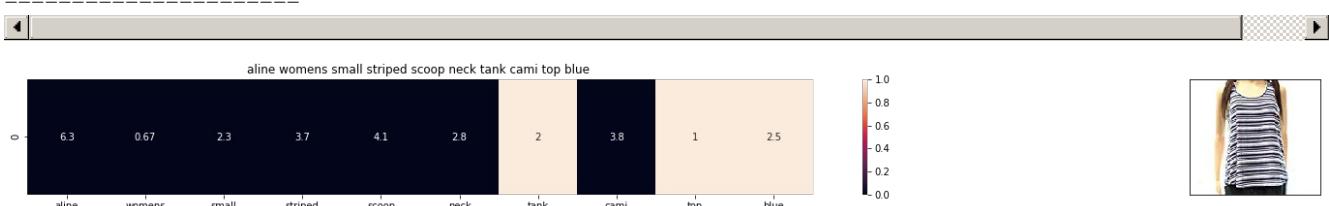
Euclidean Distance from input is : 5.486600993253093



ASIN : B074TH83H1
 TITLE : bcbgmaxazria womens lace offtheshoulder top pink
 BRAND : BCBGMAXAZRIA
 COLOR : Bare Pink
 Euclidean Distance from input is : 5.6703732731784715



ASIN : B00J6ZNVDS
 TITLE : carolines treasures 479m4055nawhm briard mom tshirt ladies cut blue long sleeve medium
 BRAND : Caroline's Treasures
 COLOR : Blue
 Euclidean Distance from input is : 5.695101091967778

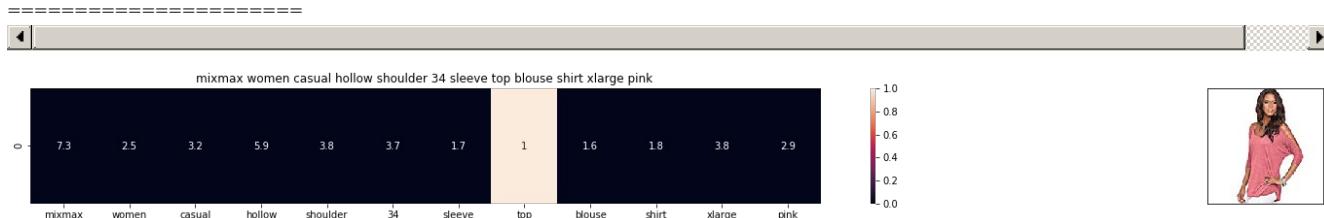


ASIN : B073V44QB8
 TITLE : aline womens small striped scoop neck tank cami top blue
 BRAND : a_line
 COLOR : Blue

Euclidean Distance from input is : 5.695101091967778

Euclidean distance from input is : 5.699453840444688

=====



ASIN : B01CA1MOUK

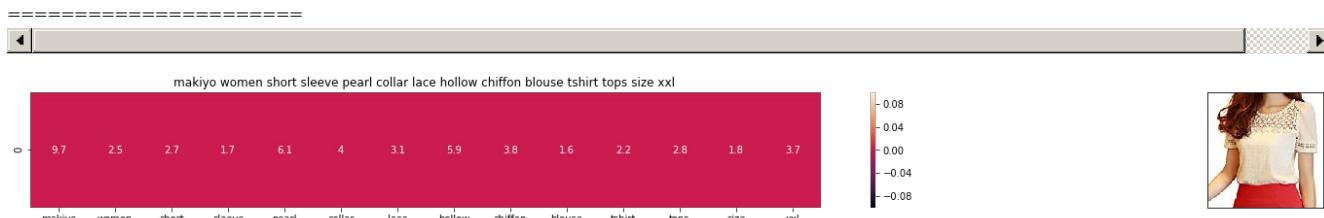
TITLE : mixmax women casual hollow shoulder 34 sleeve top blouse shirt xlarge pink

BRAND : Mixmax

COLOR : Pink

Euclidean Distance from input is : 5.705556125522183

=====



ASIN : B012SQWPYC

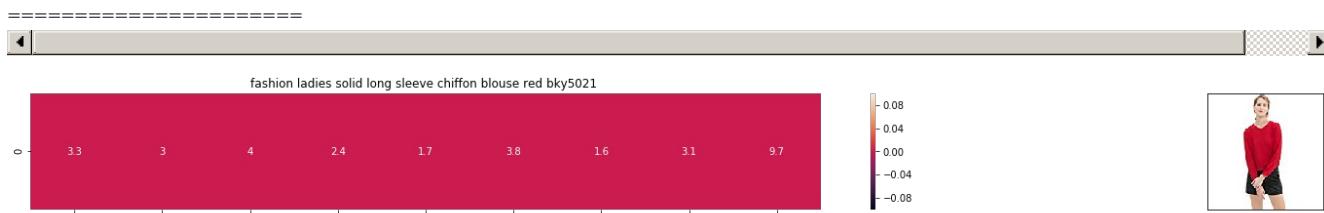
TITLE : makiyo women short sleeve pearl collar lace hollow chiffon blouse tshirt tops size xxl

BRAND : MAKIYO

COLOR : White

Euclidean Distance from input is : 5.7067252686775785

=====

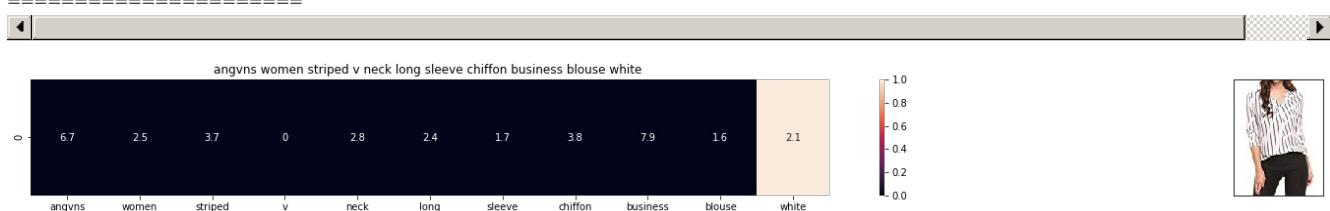


ASIN : B00R060H4M

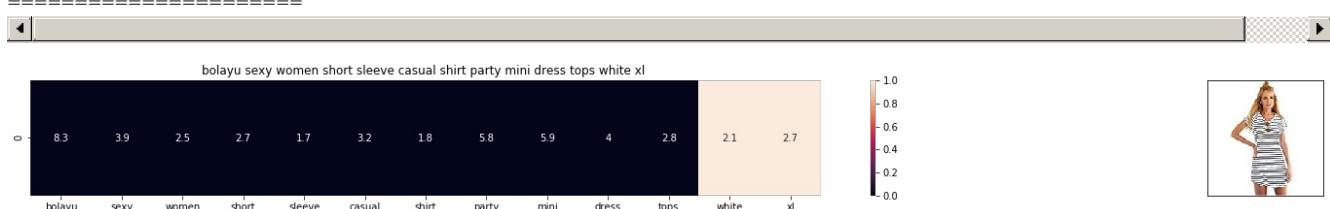
TITLE : fashion ladies solid long sleeve chiffon blouse red bky5021
BRAND : bankhunyabangyai store
COLOR : Red
Euclidean Distance from input is : 5.728879867090635



ASIN : B00PGY4T8E
TITLE : pandada womens lace tshirt see tee tops basic shirt blouse long sleeve
BRAND : PanDaDa
COLOR : Grey
Euclidean Distance from input is : 5.744865719461195



ASIN : B06XH9MZ2F
TITLE : angvns women striped v neck long sleeve chiffon business blouse white
BRAND : Angvns
COLOR : White
Euclidean Distance from input is : 5.7641514752047955





ASIN : B01DU2OULE

TITLE : bolayu sexy women short sleeve casual shirt party mini dress tops white xl

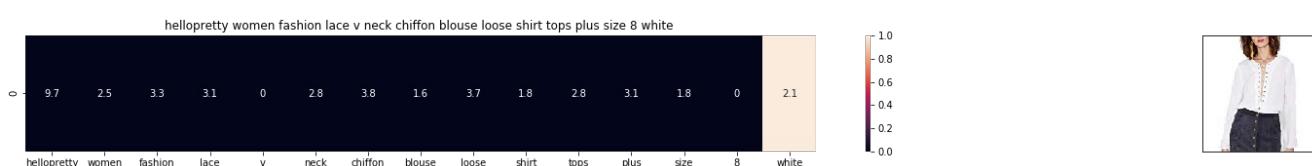
BRAND : Bolayu

COLOR : White

Euclidean Distance from input is : 5.767112910830426

=====
=====

=====



ASIN : B016CW52N4

TITLE : hellopretty women fashion lace v neck chiffon blouse loose shirt tops plus size 8 white

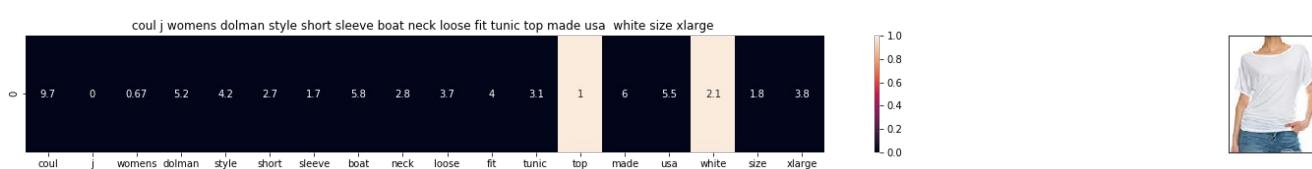
BRAND : HelloPretty

COLOR : White

Euclidean Dist

Euclidean Distance from Input is : 5.771020020772402

1



ASIN : B06XKY1HKZ

TITLE : coul j womens dolman style short sleeve boat neck loose fit tunic top made usa white size xlarge

BRAND : coul J

COLOR : White

Euclidean Dist

Euclidean Distance from Input is : 5.770905157458772

1





ASIN : B074TP53WS

TITLE : tahari pink womens petite floral printed blouse blue pxs

BRAND : T Tahari

COLOR : Blue

Euclidean Distance from input is : 5.787750300011851

=====

=====



ASIN : B0732TLJ16

TITLE : women sexy cross deep vneck blouse shirt floral print long sleeve loose chiffon top

BRAND : rbwinner

COLOR : White

Euclidean Distance from input is : 5.793082385376797

=====

=====



ASIN : B0746Q5X93

TITLE : xscape lace sequin womens zipback knit crop top black 10

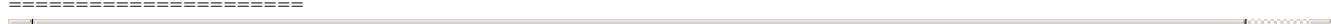
BRAND : Xscape

COLOR : Black

Euclidean Distance from input is : 5.80496935124988

=====

=====





ASIN : B019EGEV9Q

TITLE : trendyloosefit women plus size loose fit top shirt blouse bust 52

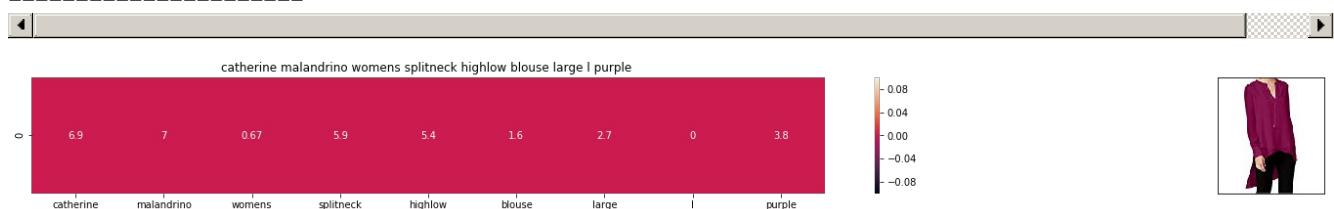
BRAND : Trendyloosefit

COLOR : Brown

Euclidean Distance from input is : 5.817467776149074

=====

=====



ASIN : B01N3T2WXO

TITLE : catherine malandrino womens splitneck highlow blouse large l purple

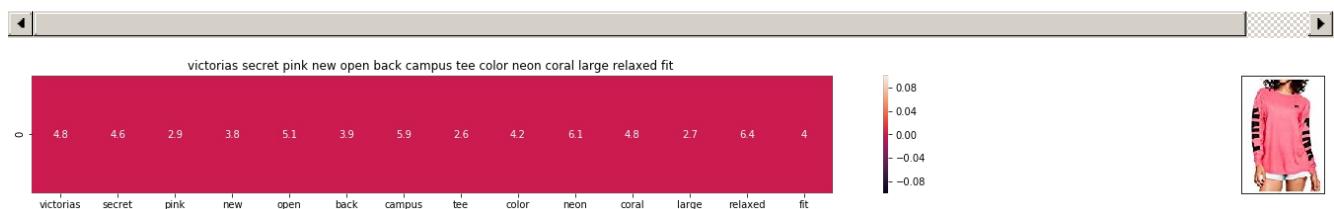
BRAND : Catherine Malandrino

COLOR : Purple

Euclidean Distance from input is : 5.819634591408617

=====

=====



ASIN : B0744X5DNR

TITLE : victorias secret pink new open back campus tee color neon coral large relaxed fit

BRAND : VS Pink

COLOR : Neon Coral

Euclidean Distance from input is : 5.8292259483643

=====

=====

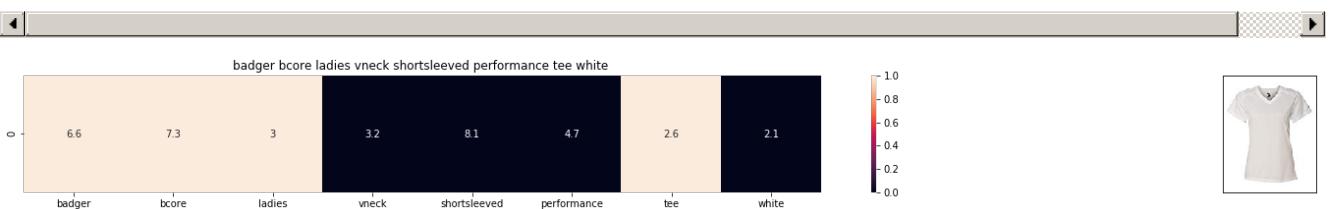


In [47]:

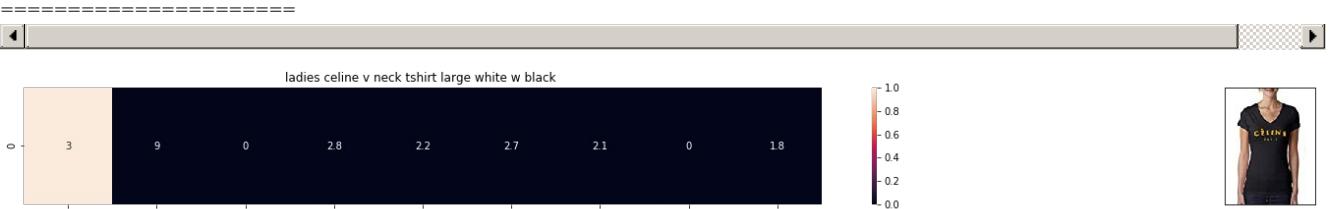
```
weighted_final(600,20,30,1,2,3)
```



ASIN : B010GMG3R0
 TITLE : badger bd4160 bcore ladies tee silver small
 BRAND : BADGER
 COLOR : Silver
 Euclidean Distance from input is : 0.007450698150528802



ASIN : B00DOL9HN8
 TITLE : badger bcore ladies vneck shortsleeved performance tee white
 BRAND : BADGER
 COLOR : White
 Euclidean Distance from input is : 6.525759718173288

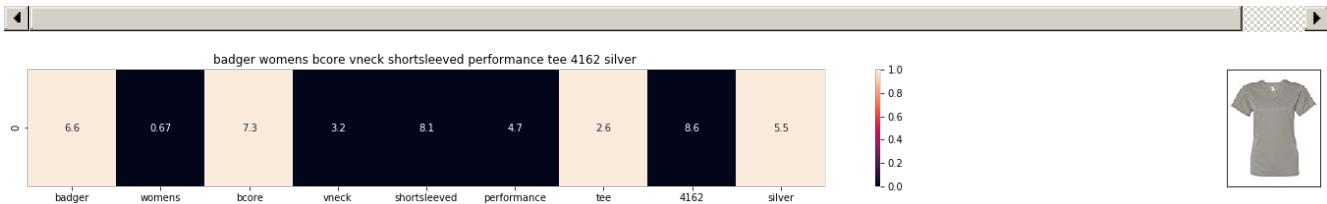


ASIN : B015S1TSNU
 TITLE : ladies celine v neck tshirt large white w black
 BRAND : Fashion Place 4 Less

COLOR : White w Black

Euclidean Distance from input is : 6.6734175633111175

=====



ASIN : B00DOL8VXK

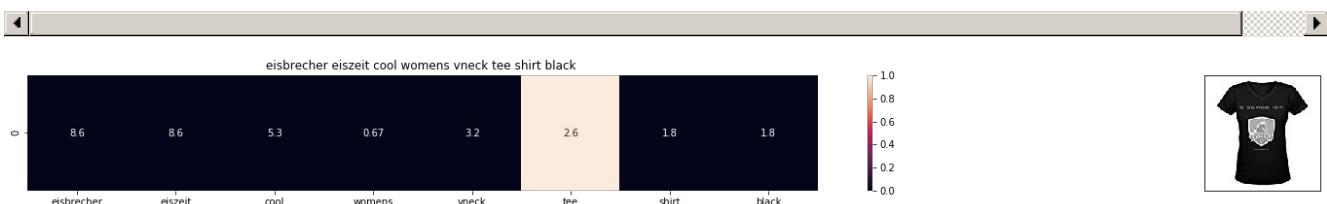
TITLE : badger womens bcore vneck shortsleeved performance tee 4162 silver

BRAND : BADGER

COLOR : Silver

Euclidean Distance from input is : 6.690134896172418

=====



ASIN : B01IT9KGS4

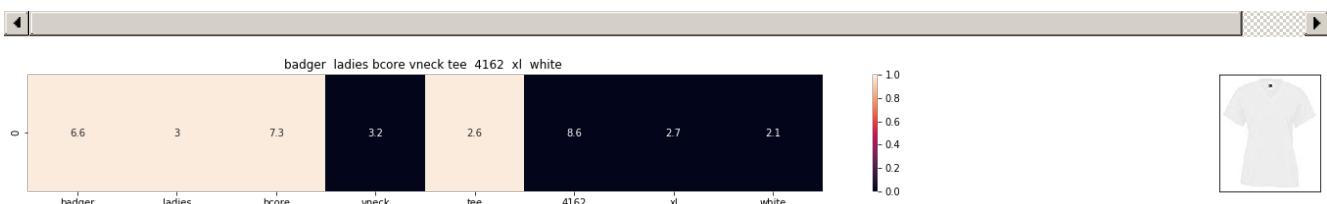
TITLE : eisbrecher eiszeit cool womens vneck tee shirt black

BRAND : Bunny Angle

COLOR : Black

Euclidean Distance from input is : 6.825387946985965

=====



ASIN : B00DOL9FXU

TITLE : badger ladies bcore vneck tee 4162 xl white

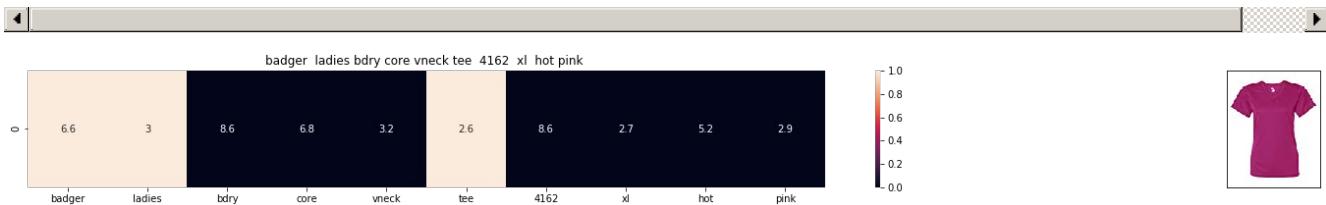
BRAND : BADGER

COLOR : White

Euclidean Distance from input is : 7.000765821688913

=====

=====



ASIN : B00DOL8XYM

TITLE : badger ladies bdry core vneck tee 4162 xl hot pink

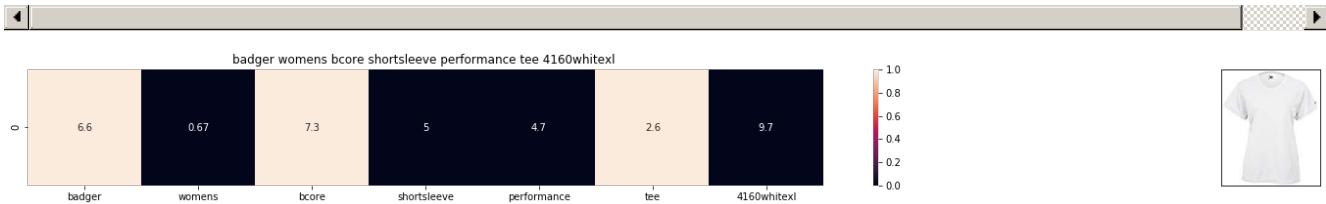
BRAND : BADGER

COLOR : Hot Pink

Euclidean Distance from input is : 7.008091974660944

=====

=====



ASIN : B00C22HYDK

TITLE : badger womens bcore shortsleeve performance tee 4160whitexl

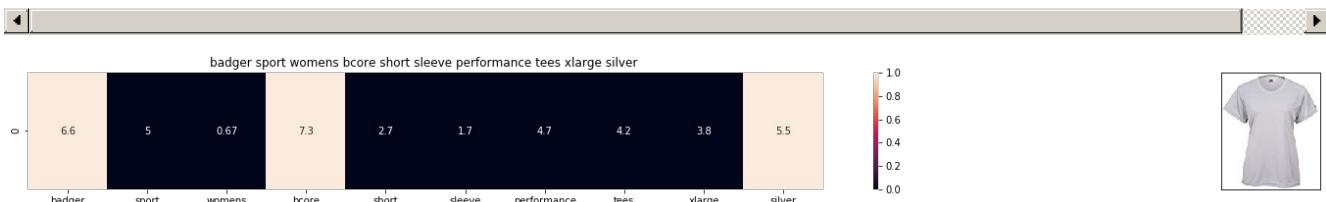
BRAND : Badger Sportswear

COLOR : White

Euclidean Distance from input is : 7.107162178907338

=====

=====





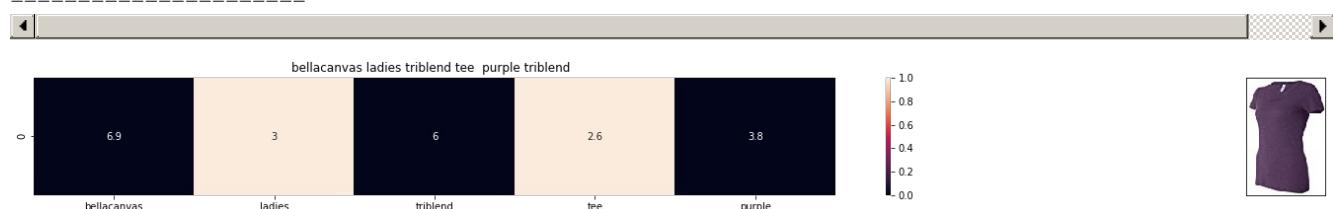
ASIN : B003WYE192

TITLE :: badger sport womens bcore short sleeve performance tees xlarge silver

BRAND : Badger Sportswear

COLOR : Silver

Euclidean Distance from input is : 7.2230574289957685



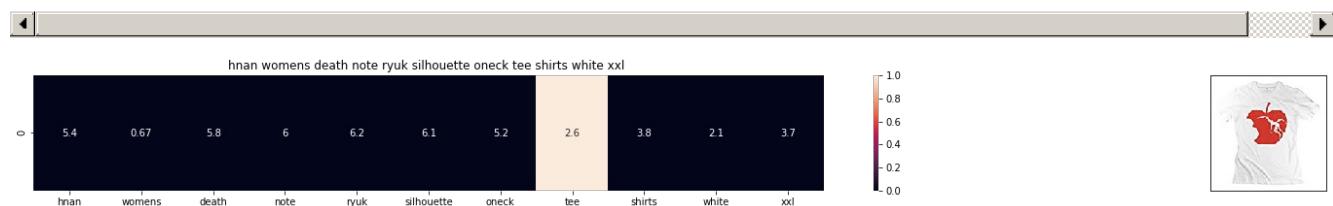
ASIN : B007P88J6W

TITLE : bellacanvas ladies triblend tee purple triblend

BRAND : Bodek And Rhodes

COLOR : Purple

Euclidean Distance from input is : 7.282802708963762



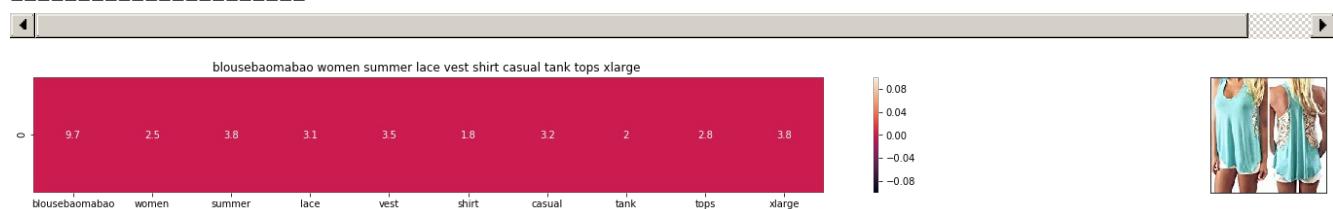
ASIN : B011I80UI0

TITLE : hnan womens death note ryuk silhouette oneck tee shirts white xxl

BRAND : H'nan

COLOR : White

Euclidean Distance from input is : 7.373106087608783





ASIN : B01EMNLU90

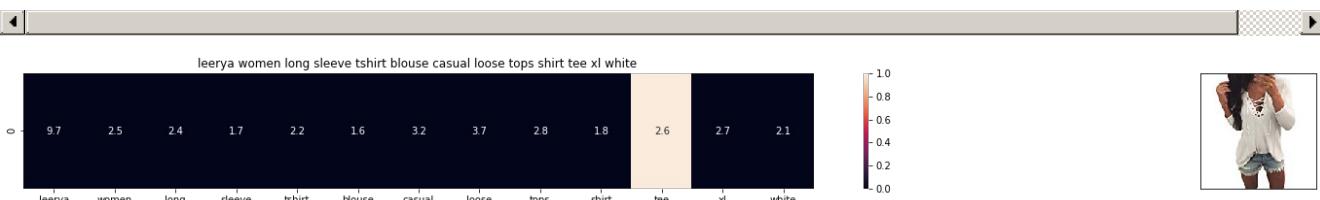
TITLE : blousebaomabao women summer lace vest shirt casual tank tops xlarge

BRAND : Baomabao_Short

COLOR : Blue

Euclidean Distance from input is : 7.392922592193202

=====



ASIN : B06XKNW8TM

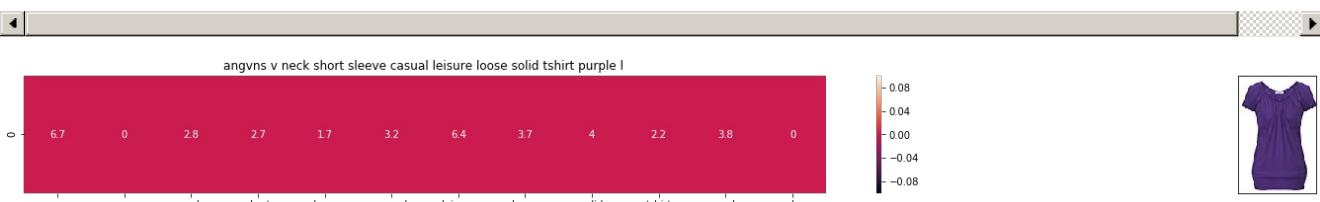
TITLE : leerya women long sleeve tshirt blouse casual loose tops shirt tee xl white

BRAND : LEERYA

COLOR : White

Euclidean Distance from input is : 7.397647518611929

=====



ASIN : B015STV2M2

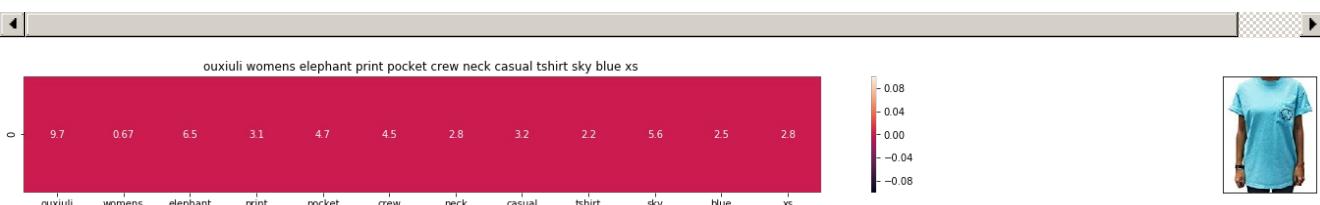
TITLE : angvns v neck short sleeve casual leisure loose solid tshirt purple l

BRAND : ANGVNS

COLOR : Purple

Euclidean Distance from input is : 7.438758934898822

=====





ASIN : B01I53HU6K

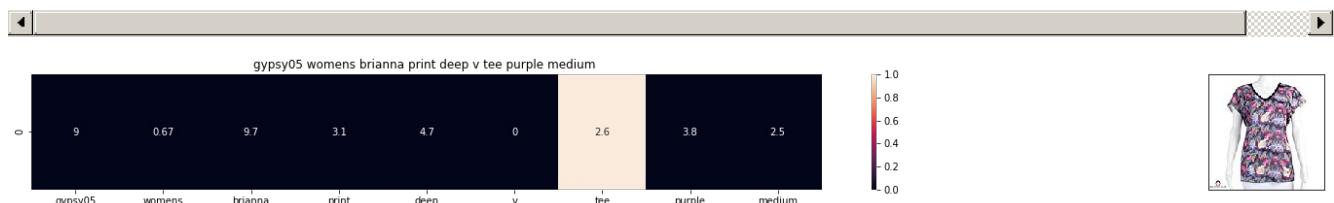
TITLE : ouxiuli womens elephant print pocket crew neck casual tshirt sky blue xs

BRAND : ouxiuli

COLOR : Sky Blue

Euclidean Distance from input is : 7.442796183045008

=====



ASIN : B074G31DTS

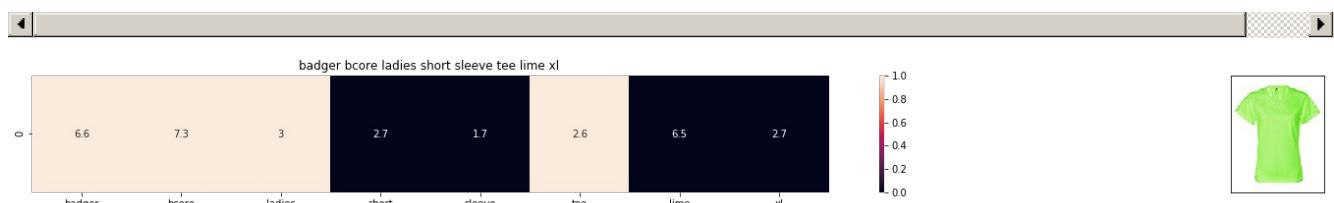
TITLE : gypsy05 womens brianna print deep v tee purple medium

BRAND : Gypsy05

COLOR : Purple

Euclidean Distance from input is : 7.443582831518829

=====



ASIN : B00DNMUTHQ

TITLE : badger bcore ladies short sleeve tee lime xl

BRAND : BADGER

COLOR : Lime

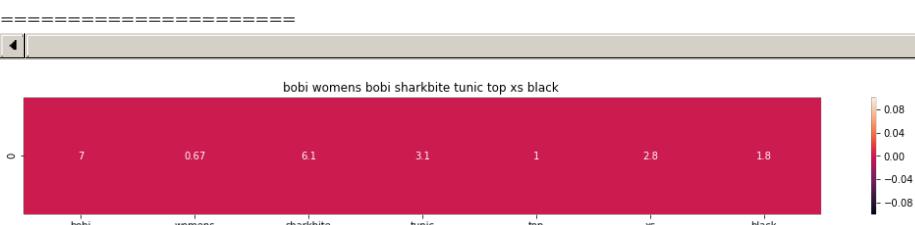
Euclidean Distance from input is : 7.460568025397139

=====

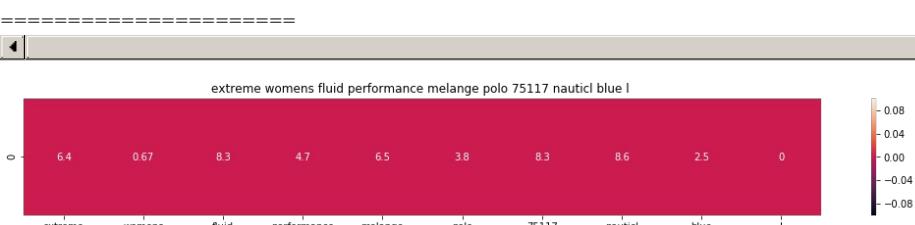




ASIN : B01N7V7SIG
 TITLE : xcvi womens top xs brown
 BRAND : XCVI
 COLOR : Brown
 Euclidean Distance from input is : 7.498636118601078



ASIN : B01M4NNXXZ
 TITLE : bobi womens bobi sharkbite tunic top xs black
 BRAND : Bobi
 COLOR : Black
 Euclidean Distance from input is : 7.504126951459694



ASIN : B00JV3MPOU
 TITLE : extreme womens fluid performance melange polo 75117 nauticl blue l
 BRAND : Ash City
 COLOR : Nautical Blue
 Euclidean Distance from input is : 7.504187285179351

In [48]:

```
weighted_final(1000,10,15,1,1,1)
```



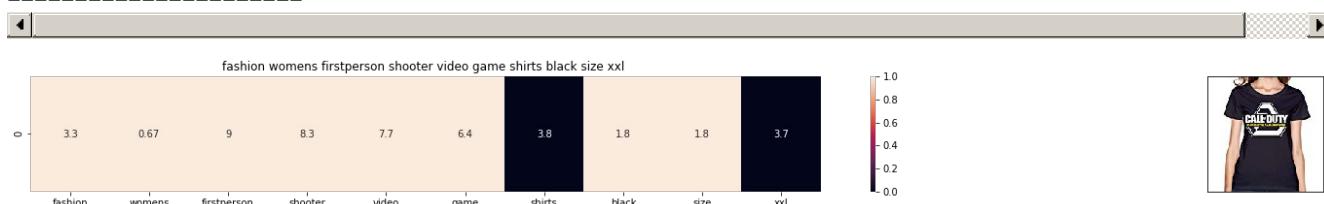
ASIN : B01INF7UVA

TITLE : fashion womens firstperson shooter video game tshirts black size l

BRAND : Aip Yep Novelty Fashion

COLOR : Black

Euclidean Distance from input is : 0.003363715277777778



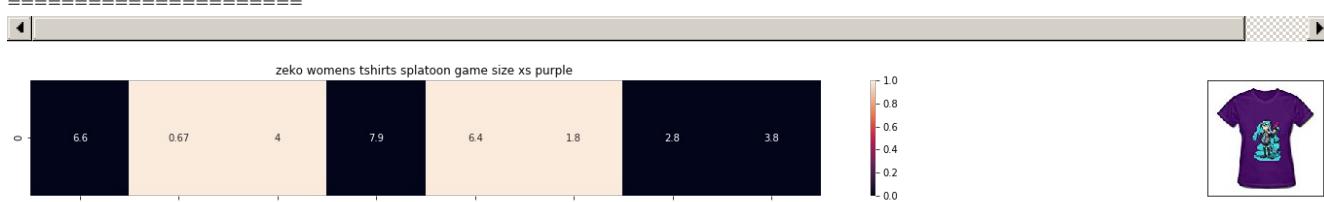
ASIN : B01INF84YM

TITLE : fashion womens firstperson shooter video game shirts black size xxl

BRAND : Aip Yep Novelty Fashion

COLOR : Black

Euclidean Distance from input is : 3.6886625819736056



ASIN : B0167UURXQ

TITLE : zeko womens tshirts splatoon game size xs purple

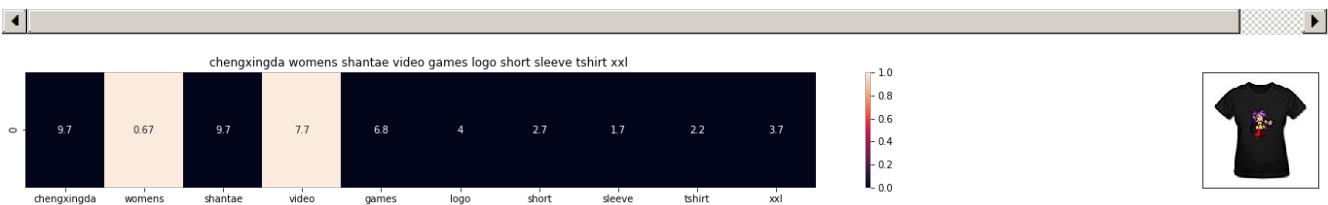
BRAND : ZEKO

COLOR : Purple

Euclidean Distance from input is : 6.434026167973081

=====

=====



ASIN : B01IMP76XI

TITLE : chengxingda womens shantae video games logo short sleeve tshirt xxl

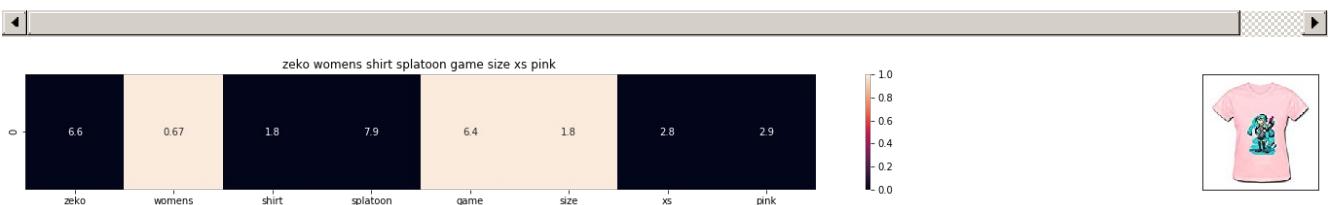
BRAND : CHENGXINGDA

COLOR : Black

Euclidean Distance from input is : 6.613532554603938

=====

=====



ASIN : B0167UTZ4I

TITLE : zeko womens shirt splatoon game size xs pink

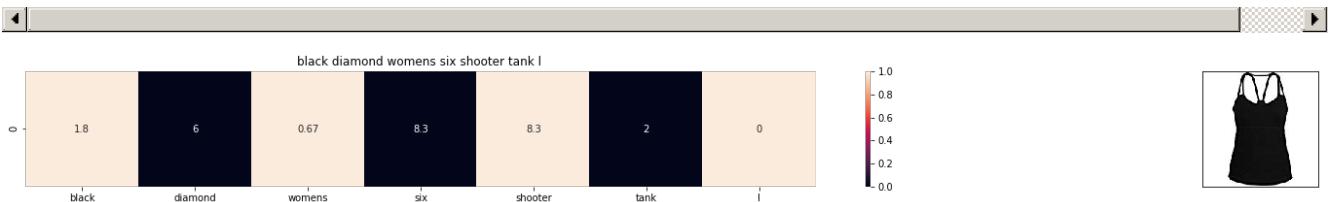
BRAND : ZEKO

COLOR : Pink

Euclidean Distance from input is : 6.653145133969717

=====

=====





ASIN : B00LTQPEQ0

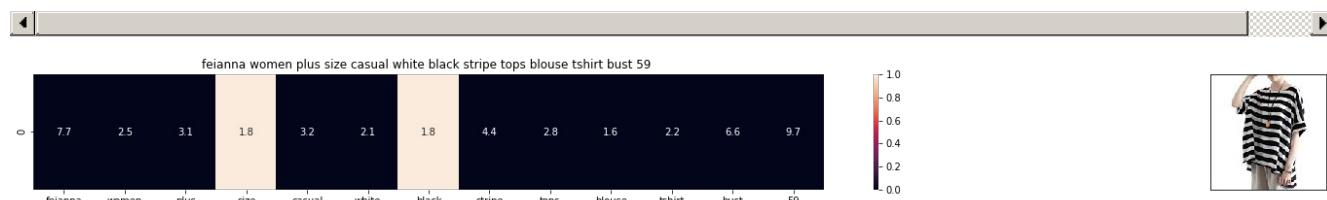
TITLE : black diamond womens six shooter tank l

BRAND : Black Diamond

COLOR : Black

Euclidean Distance from input is : 6.73443159296929

=====



ASIN : B072WQ86QJ

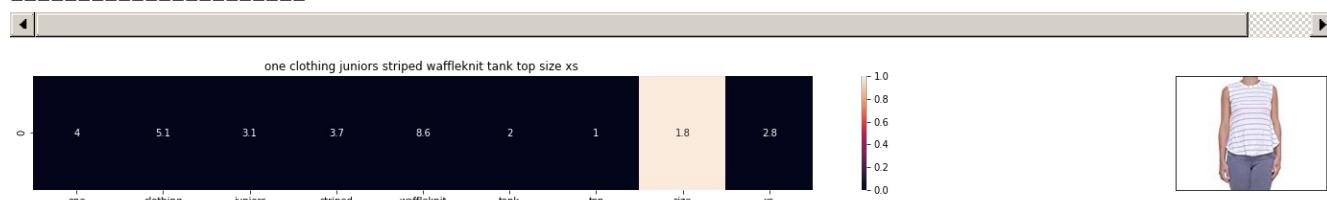
TITLE : feianna women plus size casual white black stripe tops blouse tshirt bust 59

BRAND : FEIANNA

COLOR : White Black Stripe

Euclidean Distance from input is : 6.8337576980566315

=====



ASIN : B06XG9JX79

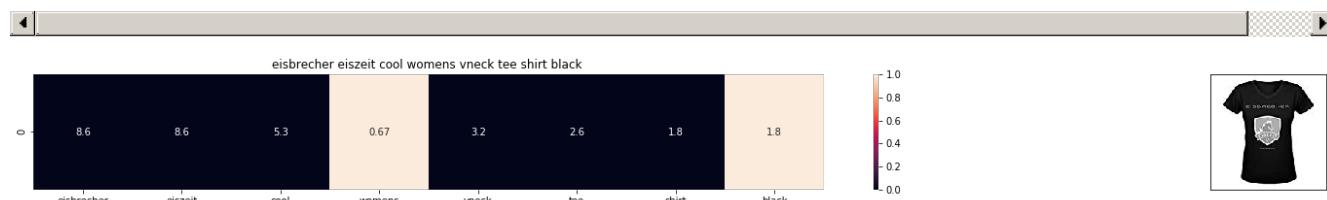
TITLE : one clothing juniors striped waffleknit tank top size xs

BRAND : One Clothing

COLOR : Cream

Euclidean Distance from input is : 6.864807688231338

=====





ASIN : B01IT9KGS4

TITLE : eisbrecher eiszeit cool womens vneck tee shirt black

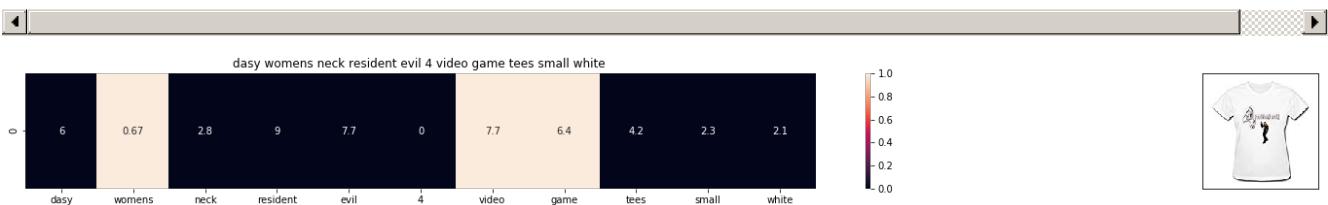
BRAND : Bunny Angle

COLOR : Black

Euclidean Distance from input is : 6.873681182677407

=====

=====



ASIN : B016BENJ38

TITLE : dasy womens neck resident evil 4 video game tees small white

BRAND : Dasy

COLOR : White

Euclidean Distance from input is : 6.887650893314878

=====

=====



Conclusion

1)Importing the required Packages 2)loading the data 3)Performing EDA on each features like colour , brand etc 4)Taking only 28k data
5)Removing Duplicates so that we dont have color different items of same product which worsen the user experience are removed 6)We do BOW and find similar items 7)We do TF-IDF and find similar items 8)We do IDF and find similar items 9)We do W2V and find similar items 10)We do TF-IDF weighted W2V and find similar items 11)We do IDF Weighted W2V and find similar items 12)We will get weighted model using text, brand and color vectors attached to get similar Items 13)We will get Image features using VGG16 and using that we will get similar items 14)Now as we have unbalanced shapes for both cnn features and original data , using asins.. We will make sure both have same shape.. 15)Then we will get a vector with image features , color , brand , Text features and get the similar items...