Amazon Apparel Recommendation

Import Modules and Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nltk
import math
import time
import re
import os
import seaborn as sns
import requests
from PIL import Image
from io import BytesIO
import warnings
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
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]:

```
data = pd.read_json('C:/Users/bhave/Downloads/tops_fashion.json')
```

In [3]:

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

```
Number of datapoints:  183138
 Number of features/variables:  19
```

In [4]:

```
data.head()
```

Out[4]:

	asin	author	availability	availability_type	brand	color	editorial_reivew	editorial_review	formatted_price
0	B016I2TS4W	None	None	None	FNC7C	None	NaN	Minions Como Superheroes Ironman Women's O Nec...	None
1	B01N49AI08	N	N	N	FIG	N	N N	Sizing runs on the small side.	N

	asin	author	availability	availability_type	brand	color	editorial_reivew	editorial_review	formatted_price
1	B01N49AI08	None	None	None	Clothing	None	NaN	FIG® recommends...	None
2	B01JDPCOHO	None	None	None	FIG Clothing	None	NaN	Sizing runs on the small side. FIG® recommends...	None
3	B01N19U5H5	None	None	None	Focal18	None	NaN	100% Brand New & Fashion Quantity: 1 Piece...	None
4	B004GSI2OS	None	Usually ships in 6-10 business days	now	FeatherLite	Onyx Black/ Stone	NaN		$26.26

In [5]:

```
# Print the name of the features/ columns
data.columns
```

Out[5]:

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

In [6]:

```
# Considering only 7 features out of 19
data = data[['asin', 'brand', 'color', 'medium_image_url', 'product_type_name', 'title', 'formatted
_price']]
```

In [7]:

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

data.head()
```

```
Number of datapoints:  183138
 Number of features/ variables:  7
```

Out[7]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
0	B016I2TS4W	FNC7C	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Minions Como Superheroes Ironman Long Sleeve R...	None
1	B01N49AI08	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG Clothing Womens Izo Tunic	None
2	B01JDPCOHO	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG Clothing Womens Won Top	None
3	B01N19U5H5	Focal18	None	https://images-na.ssl-images-	SHIRT	Focal18 Sailor Collar Bubble Sleeve Blouse	None

	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

Basic Stats for feature: product_type_name

In [8]:

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

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

In [9]:

```
# list of all the 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 [12]:

```
# list of top 10 frequent products

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

Out[12]:

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

Basic Stats for feature: brand

In [13]:

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

```
count     182987
```

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

In [14]:

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

Out[14]:

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

Basic Stats for feature: color

In [15]:

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

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

In [16]:

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

Out[16]:

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

Basic Stats for feature: formatted_price

In [17]:

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

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

In [18]:

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

```
formatted_price_count.most_common(10)
```

Out[18]:

```
[(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 feature: title

In [19]:

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

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

In [20]:

```
title_count = Counter(list(data['title']))
title_count.most_common(10)
```

Out[20]:

```
[('Nakoda Cotton Self Print Straight Kurti For Women', 77),
 ("Q-rious Women's Racerback Cotton Lycra Camsioles", 56),
 ('FINEJO Casual Women Long Sleeve Lace Irregular Hem Blouse Tops', 47),
 ('Girlzwalk Women Cami Sleeveless Printed Swing Vest Top Plus Sizes', 44),
 ("ELINA FASHION Women's Indo-Western Tunic Top Cotton Kurti", 43),
 ('Victoria Scoop Neck Front Lace Floral High-Low Top in 4 Sizes', 40),
 ("Cenizas Women's Indian Tunic Top Cotton Kurti", 39),
 ('Indistar Womens Premium Cotton Half Sleeves Printed T-Shirts/Tops (Pack of 3)',
  37),
 ("Rajnandini Women's Cotton Printed Kurti", 35),
 ('Long Sleeve Mock Neck Top', 32)]
```

Handling the missing values

In [21]:

```
data = data.loc[~data['formatted_price'].isnull()]
print('Number of datapoints after eliminating null values: ', data.shape[0])
```

```
Number of datapoints after eliminating null values:  28395
```

In [22]:

```
data = data.loc[~data['color'].isnull()]
print('Number of datapoints after eliminating color= Null: ', data.shape[0])
```

```
Number of datapoints after eliminating color= Null:  28385
```

Number of data points has been reduced from 183k to 28k.

Handling Near Duplicate Items

In [23]:

```
# Products having the same or duplicate titles
print(sum(data.duplicated('title')))
```

```
2325
```

We have 2325 products having same title but different color and/or size

In [24]:

```
# Remove the rows having very short title or description as they are not useful
data_sorted = data[data['title'].apply(lambda x: len(x.split())>4)]
print("Number of datapoints after removal of short description:  27949
```

```
Number of datapoints after removal of short description:  27949
```

In [25]:

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

Out[25]:

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

In [26]:

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

In [27]:

```
import itertools
stage1_dedupe_asins = []
i = 0
```

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

    previous_i = i

    # store the list of words of ith string in variable a
    a = data['title'].loc[indices[i]].split()

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

        # store the list of words for jth string in variable b
        b = data['title'].loc[indices[j]].split()

        # maximum length of the 2 strings
        length = max(len(a), len(b))

        # count represents the total number of words matched in both the strings
        count = 0

        for k in itertools.zip_longest(a,b):
            if (k[0] == k[1]):
                count += 1

        # number of words by which both the strings differ
        if (length - count) > 2:
            stage1_dedupe_asins.append(data_sorted['asin'].loc[indices[i]])

            if j == num_data_points-1:
                stage1_dedupe_asins.append(data_sorted['asin'].loc[indices[j]])

            i = j
            break
        else:
            j += 1
    if previous_i == i:
        break
```

In [28]:

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

Removed the duplicates which differ only at the end

In [29]:

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

```
Number of datapoints:  17593
```

In [ ]:

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

stage2_dedupe_asins = []
while len(indices)!= 0:
    i = indices.pop()
    stage2_dedupe_asins.append(data['asin'].loc[i])

    a= data['title'].loc[i].split()

    for j in indices:
        b = data['title'].loc[j].split()
        length = max(len(a), len(b))
        count = 0

        for k in itertools.zip_longest(a,b):
            if(k[0] == k[1]):
                count += 1
```

```
                count += 1

        if(length - count) < 3:
            indices.remove(j)
```

In [ ]:

```
data = data.loc[data['asin'].isin(stage2_dedupe_asins)]
print("Number of datapoints after stage 2 of deduplication: ", data.shape[0])
```

Text Pre-Processing

In [30]:

```
import nltk


stop_words = set(stopwords.words('english'))
print('List of stopwords: ', stop_words)

def nlp_preprocess(text, index, column):
    if type(text) is not int:
        string = ""
        for words in text.split():
            word = ("".join(e for e in words if e.isalnum()))
            word = word.lower()

            if not word in stop_words:
                string += word + " "
        data[column][index] = string
```

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

In [31]:

```
start_time = time.clock()

for index, row in data.iterrows():
    nlp_preprocess(row['title'], index, 'title')

print(time.clock() - start_time, "seconds")
```

```
4.616456190125411 seconds
```

Bag of Words on Title

In [32]:

```
def display_img(url, ax, fig):
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    plt.imshow(img)

def plot_heatmap(keys, values, labels, url, text):
    # Divide the plot into 2 parts: heat map and image of apparel
```

```
    # divide the plot into 2 parts: heat-map and image of apparel
    gs = gridspec.GridSpec(2, 2, width_ratios=[4,1], height_ratios= [4,1])
    fig = plt.figure(figsize=(25,3))

    # Plotting the heat-map
    ax = plt.subplot(gs[0])
    ax = sns.heatmap(np.array([values]), annot=np.array([labels]))
    ax.set_xticklabels(keys)
    ax.set_title(text)

    # Plotting the image of apparel
    ax = plt.subplot(gs[1])
    ax.grid(False)
    ax.set_xticks([])
    ax.set_yticks([])

    display_img(url, ax, fig)

    plt.show()

def plot_heatmap_image(doc_id, vec1, vec2, url, text, model):
    intersection = set(vec1.keys()) & set(vec2.keys())

    for i in vec2:
        if i not in intersection:
            vec2[i]=0

    keys = list(vec2.keys())
    values = [vec2[x] for x in vec2.keys()]

    if model == 'bag_of_words':
        labels = values
    elif model == 'tfidf':
        labels = []
        for x in vec2.keys():
            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():
            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)

def text_to_vector(text):
    word = re.compile(r'\w+')
    words = word.findall(text)
    return Counter(words)

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

    vector1 = text_to_vector(text1)

    vector2 = text_to_vector(text2)

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

In [33]:

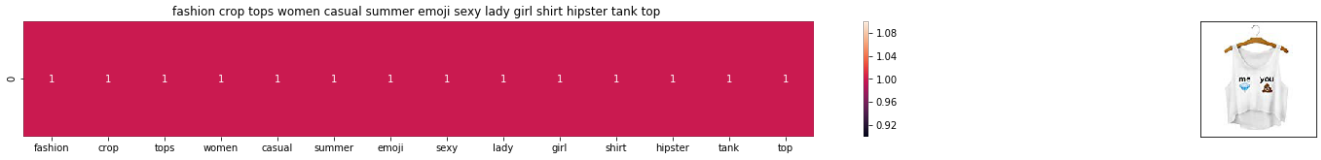
```
from sklearn.feature_extraction.text import CountVectorizer
title_vectorizer = CountVectorizer()
title_features = title_vectorizer.fit_transform(data['title'])
title_features.get_shape()
```

Out[33]:

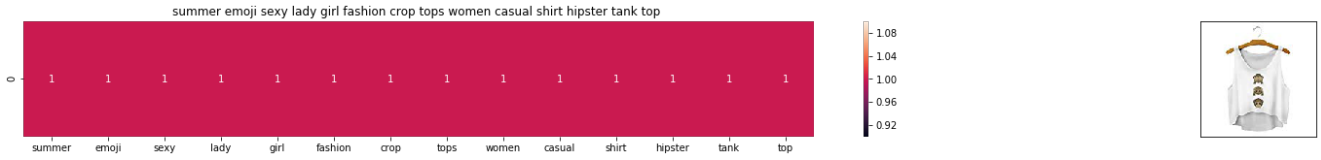
```
(16435, 12684)
```

In [34]:

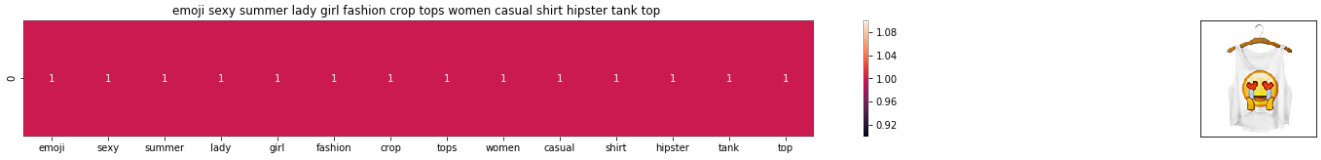
```
def bag_of_words_model(doc_id, num_results):

    pairwise_dist = pairwise_distances(title_features, title_features[doc_id])
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    pdist = np.sort(pairwise_dist.flatten())[0:num_results]

    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]], '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: ', pdist[i])
        print('='*60)

bag_of_words_model(12566, 15)
```



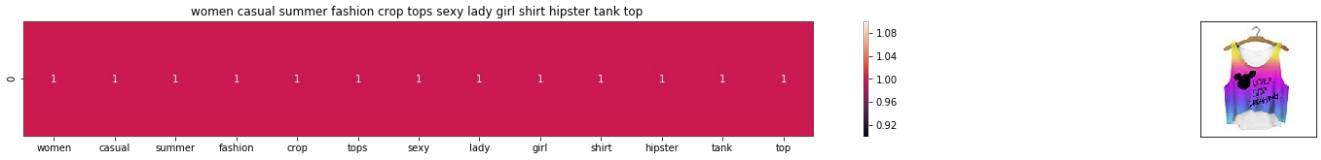
ASIN: B010V3B44G
Brand: Doxi Supermall
Title: fashion crop tops women casual summer emoji sexy lady girl shirt hipster tank top
Euclidean similarity with the query image: 0.0
==



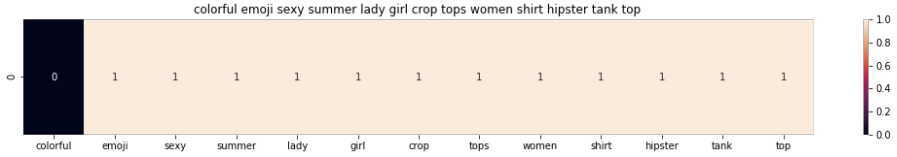
ASIN: B010V3BDII
Brand: Doxi Supermall
Title: summer emoji sexy lady girl fashion crop tops women casual shirt hipster tank top
Euclidean similarity with the query image: 0.0
==



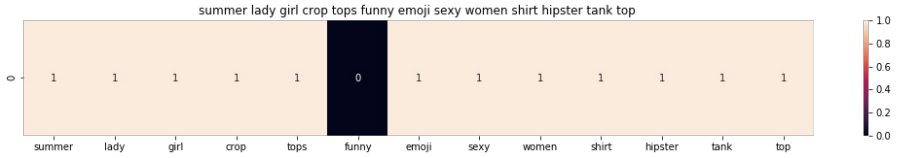
ASIN: B010V3BLWQ
Brand: Doxi Supermall
Title: emoji sexy summer lady girl fashion crop tops women casual shirt hipster tank top
Euclidean similarity with the query image: 0.0
==



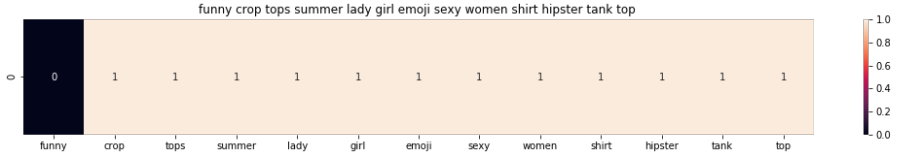
ASIN: B010V3AYSS
Brand: Doxi Supermall
Title: women casual summer fashion crop tops sexy lady girl shirt hipster tank top
Euclidean similarity with the query image: 1.0
==



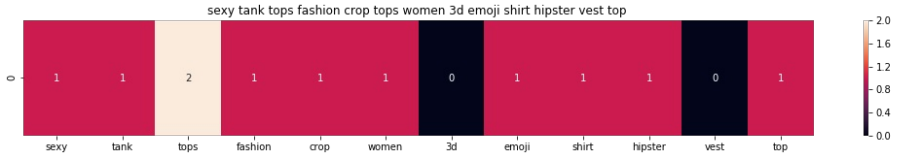
ASIN: B010V3BQZS
Brand: Doxi Supermall
Title: colorful emoji sexy summer lady girl crop tops women shirt hipster tank top
Euclidean similarity with the query image: 1.7320508075688772
==========



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
==========

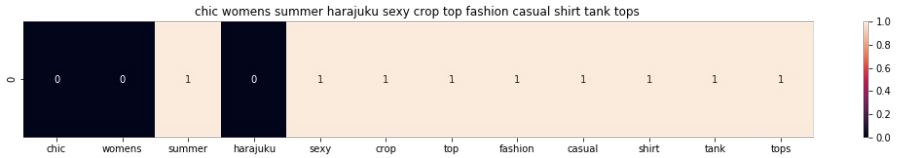


ASIN: B010V3C116
Brand: Doxi Supermall
Title: funny crop tops summer lady girl emoji sexy women shirt hipster tank top
Euclidean similarity with the query image: 1.7320508075688772
==========

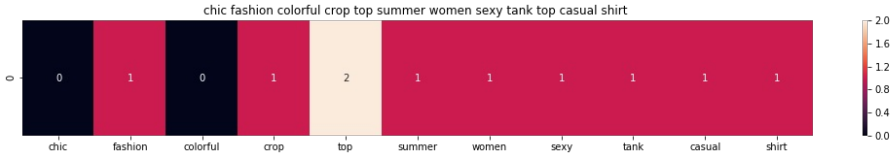


ASIN: B010V3DB9C
Brand: Doxi Supermall
Title: sexy tank tops fashion crop tops women 3d emoji shirt hipster vest top
Euclidean similarity with the query image: 2.6457513110645907
==========

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
==========



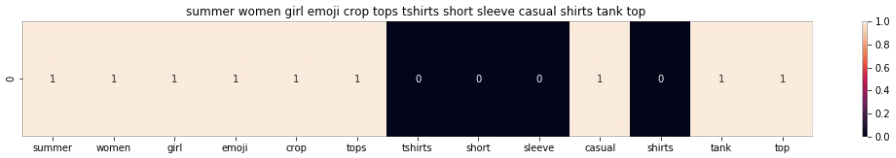
ASIN: B011RCJEMO
Brand: Chiclook Cool
Title: chic womens summer harajuku sexy crop top fashion casual shirt tank tops
Euclidean similarity with the query image: 2.8284271247461903
===



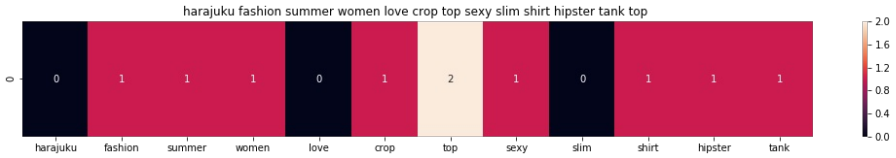
ASIN: B011RCJ6UE
Brand: Chiclook Cool
Title: chic fashion colorful crop top summer women sexy tank top casual shirt
Euclidean similarity with the query image: 2.8284271247461903
===

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: B010V35OBU
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
===

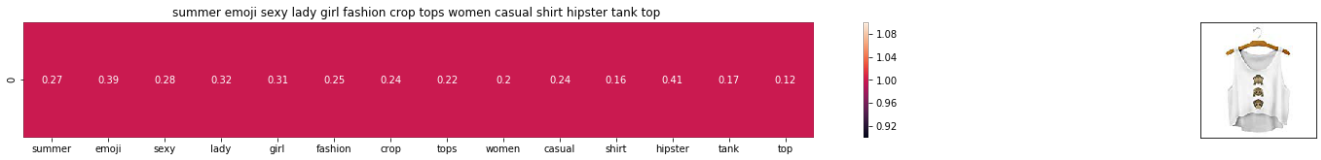
TF-IDF based Product Similarity

In [52]:

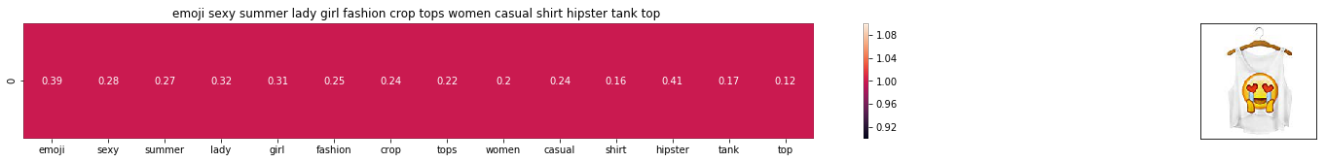
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_title_vectorizer = TfidfVectorizer(min_df = 0)
tfidf_title_features = tfidf_title_vectorizer.fit_transform(data['title'])
```

In [53]:

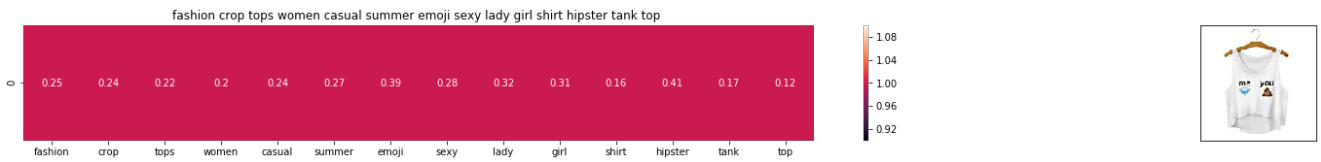
```
def tfidf_model(doc_id, num_results):
    # doc_id is apparel's id in a given corpus

    pairwise_dist = pairwise_distances(tfidf_title_features, tfidf_title_features[doc_id])
    # pairwise_dist stores the distance between the query or given input apparel to the remaining
apparels

    indices = np.argsort(pairwise_dist.flatten())[0: num_results]
    # argsort returns the indices of 9 smallest distances between the apparels

    pdists = np.sort(pairwise_dist.flatten())[0: num_results]
    # pdists stores the 9 smallest distances

    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]], 'tfidf')
        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("="*100)

# consider the example for any index number
tfidf_model(12566, 20)
```



Asin: B010V3BDII
Brand: Doxi Supermall
Euclidean distance from the given image: 0.0
==========

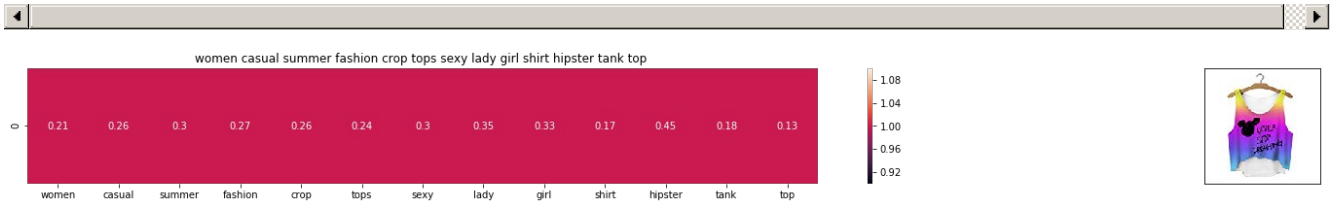


Asin: B010V3BLWQ
Brand: Doxi Supermall
Euclidean distance from the given image: 0.0
==========

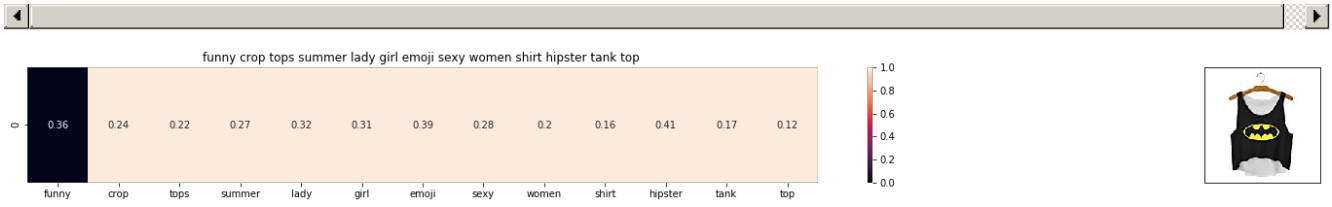


Asin: B010V3B44G
Brand: Doxi Supermall

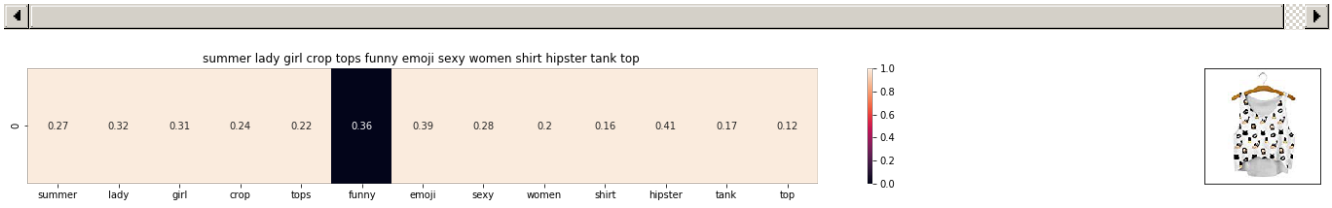
Euclidean distance from the given image: 0.0
==========



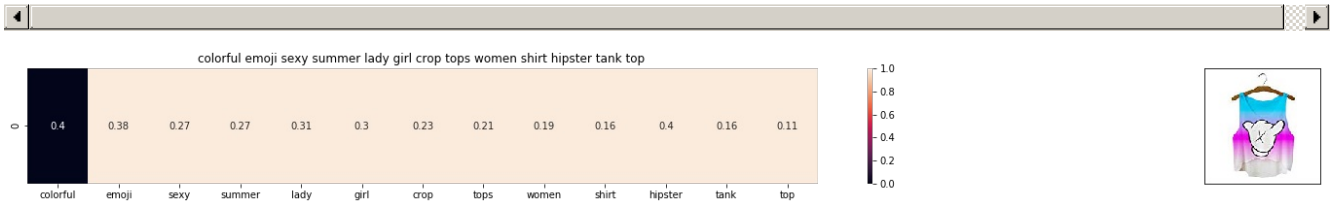
Asin: B010V3AYSS
Brand: Doxi Supermall
Euclidean distance from the given image: 0.40138594750234946
==========



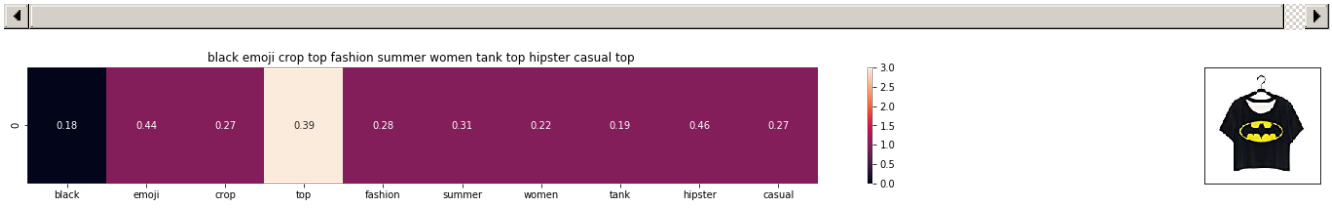
Asin: B010V3C116
Brand: Doxi Supermall
Euclidean distance from the given image: 0.4954421553895553
==========



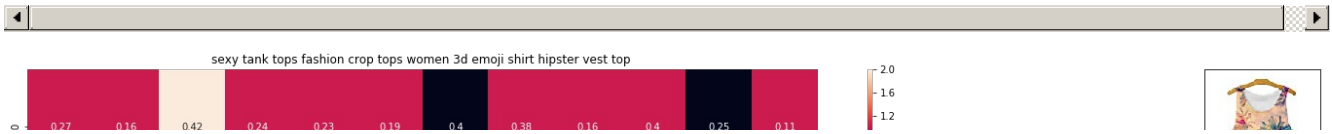
Asin: B010V3BVMQ
Brand: Doxi Supermall
Euclidean distance from the given image: 0.4954421553895553
==========



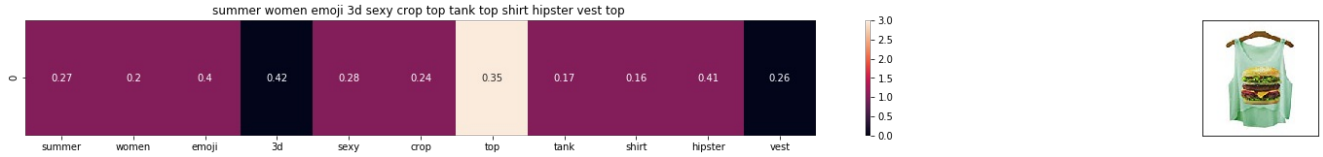
Asin: B010V3BQZS
Brand: Doxi Supermall
Euclidean distance from the given image: 0.5241689140292635
==========



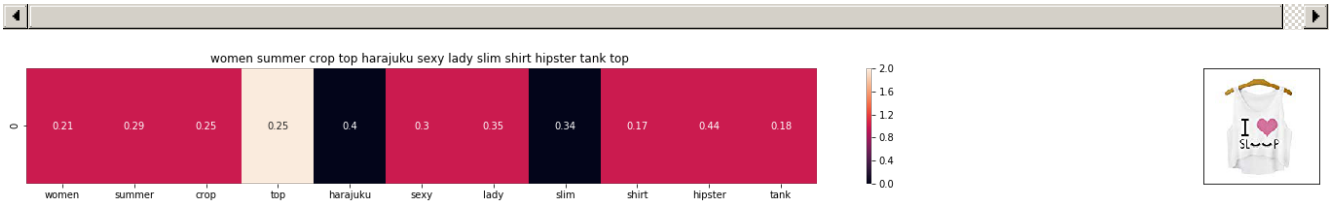
Asin: B0124E80M4
Brand: Doxi Supermall
Euclidean distance from the given image: 0.6841581626642753
==========



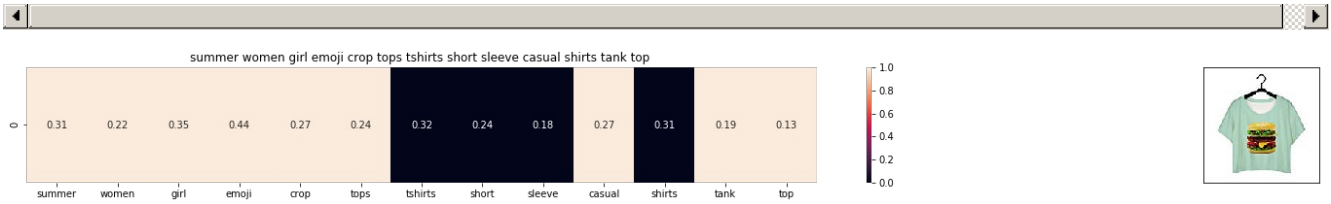
Asin: B010V3DB9C
Brand: Doxi Supermall
Euclidean distance from the given image: 0.7731343455110793
==========



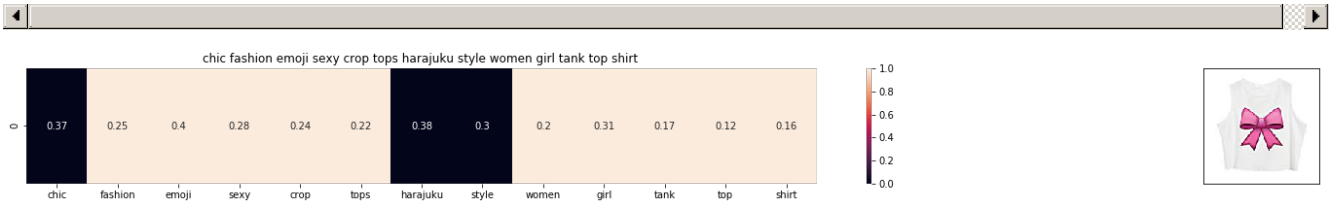
Asin: B010V3E5EC
Brand: Doxi Supermall
Euclidean distance from the given image: 0.8128754313990525
==========



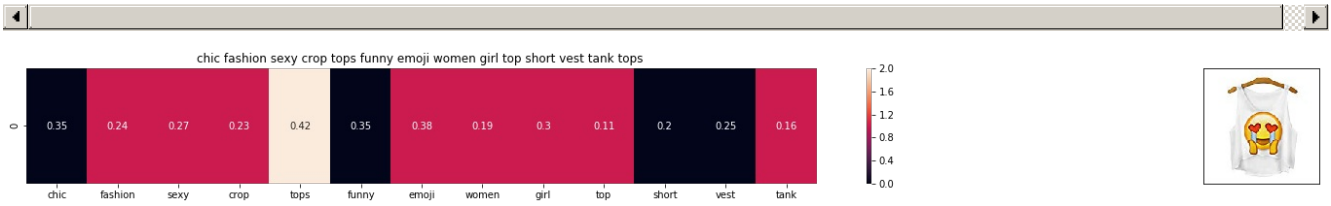
Asin: B010V3EDEE
Brand: Doxi Supermall
Euclidean distance from the given image: 0.8437056912864757
==========



Asin: B0124ECIU4
Brand: Doxi Supermall
Euclidean distance from the given image: 0.8553483954246196
==========

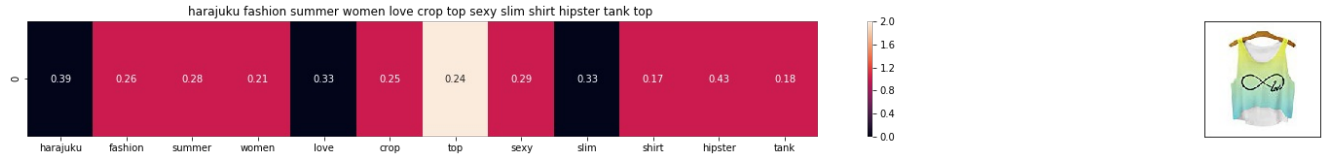


Asin: B011RCJPR8
Brand: Chiclook Cool
Euclidean distance from the given image: 0.8826321316686155
==========



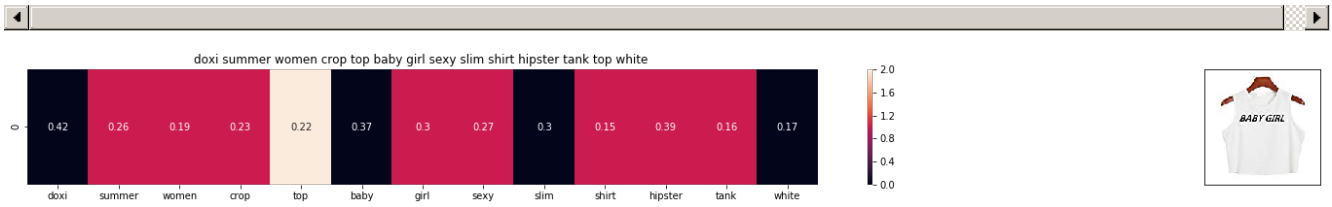
Asin: B011RCJH58
Brand: Chiclook Cool
Euclidean distance from the given image: 0.900401746801386

==========



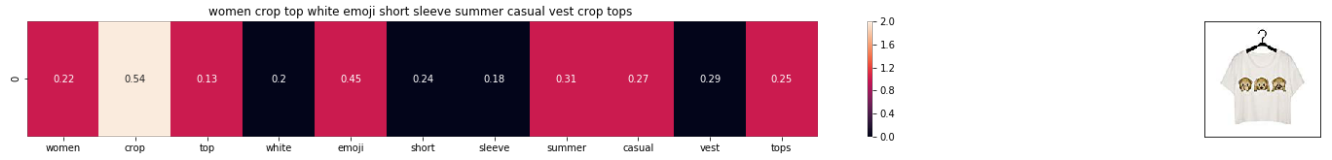
Asin: B010V35OBU
Brand: Doxi Supermall
Euclidean distance from the given image: 0.9172816572673459

==========



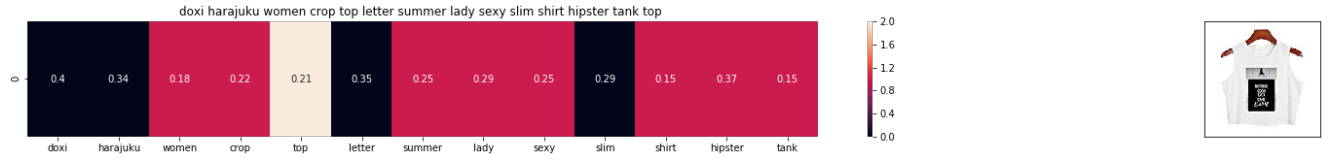
Asin: B010V3A23U
Brand: Doxi Supermall
Euclidean distance from the given image: 0.9303848929561549

==========



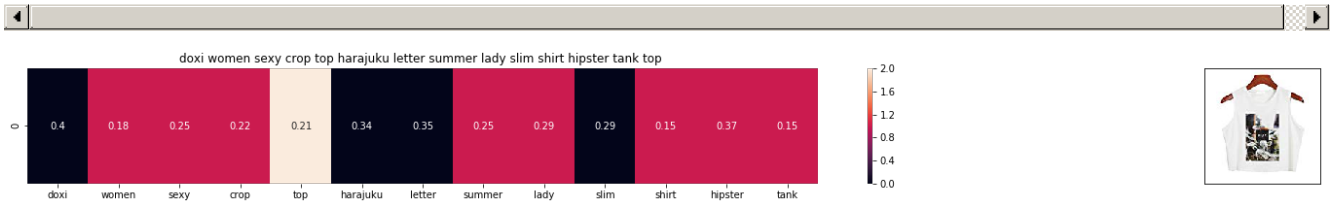
Asin: B0124E7MHS
Brand: Doxi Supermall
Euclidean distance from the given image: 0.9334421080980748

==========



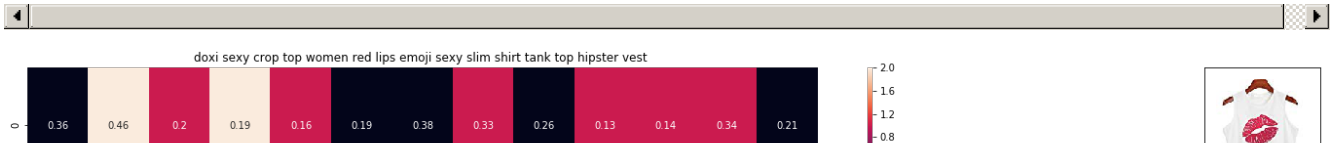
Asin: B010V380LQ
Brand: Doxi Supermall
Euclidean distance from the given image: 0.9525344637195033

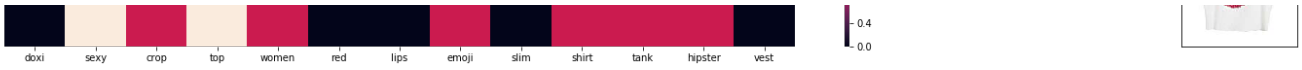
==========



Asin: B010V39146
Brand: Doxi Supermall
Euclidean distance from the given image: 0.9525344637195033

==========






```
Asin:  B010TKXEHG
Brand:  Doxi Supermall
Euclidean distance from the given image:  0.9560708174154958
```

IDF Based Product Smilarity

In [55]:

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

In [42]:

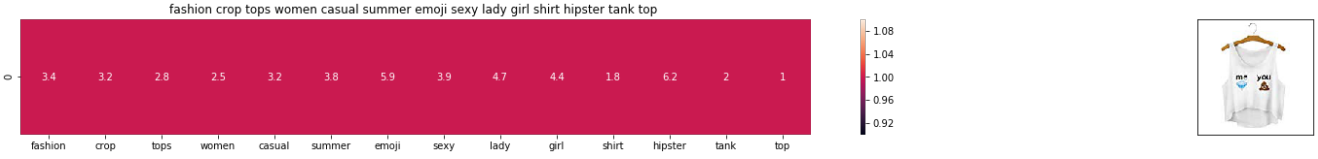
```
def n_containing(word):
    return sum(1 for blob in data['title'] if word in blob.split())

def idf(word):
    return math.log(data.shape[0] / (n_containing(word)))
```

In [56]:

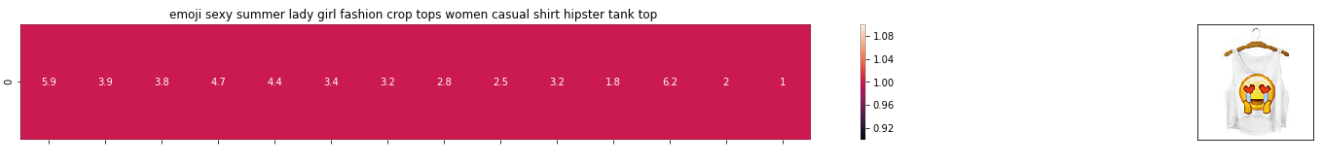
```
idf_title_features = idf_title_features.astype(np.float)

for i in idf_title_vectorizer.vocabulary_.keys():
    idf_val = idf(i)

    for j in idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0]:
        idf_title_features[j, idf_title_vectorizer.vocabulary_[i]] = idf_val
```

In [57]:

```
def idf_model(doc_id, num_results):
    pairwise_dist = pairwise_distances(idf_title_features, idf_title_features[doc_id])
    indices = np.argsort(pairwise_dist.flatten())[0: num_results]
    pdists = np.sort(pairwise_dist.flatten())[0: num_results]
    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 th given image: ', pdists[i])
        print('='*125)
idf_model(12566, 20)
```

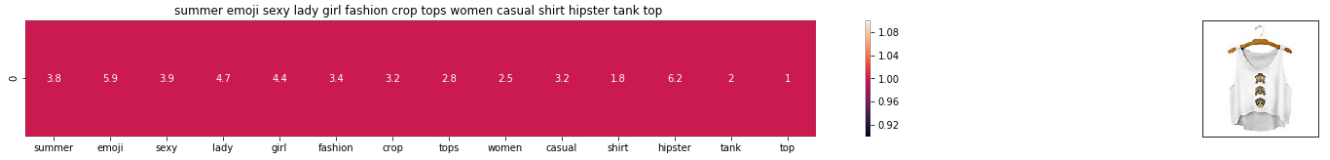



```
ASIN:  B010V3B44G
Brand:  Doxi Supermall
Euclidean distance from th given image:  0.0
```

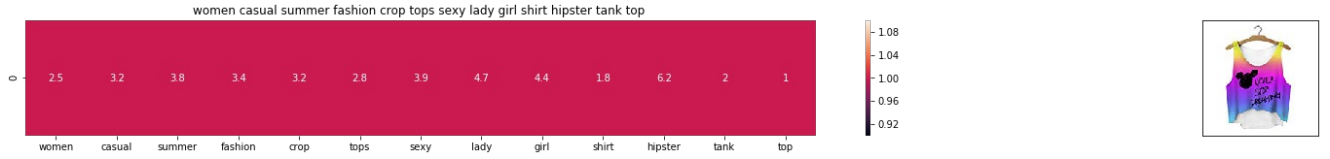


emoji sexy summer lady girl fashion crop tops women casual shirt hipster tank top

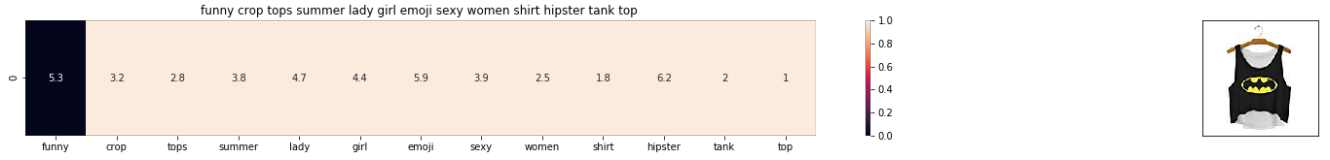
ASIN: B010V3BLWQ
Brand: Doxi Supermall
Euclidean distance from th given image: 0.0
==========



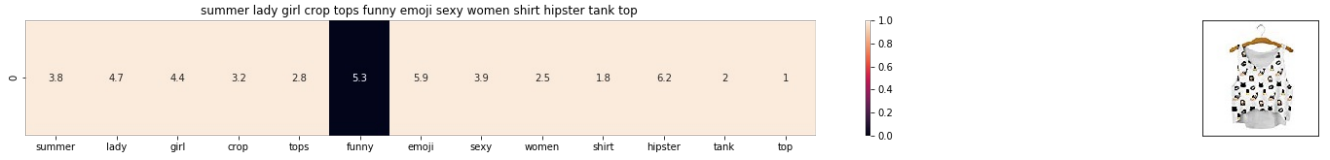
ASIN: B010V3BDII
Brand: Doxi Supermall
Euclidean distance from th given image: 0.0
==========



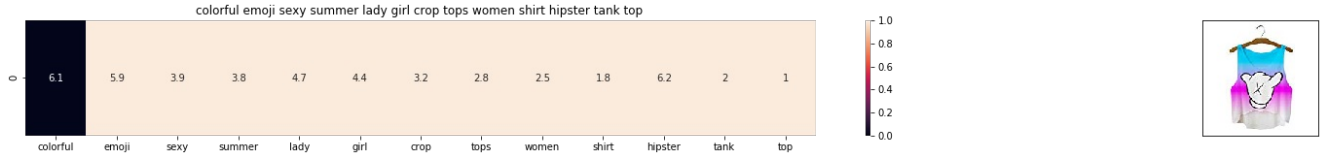
ASIN: B010V3AYSS
Brand: Doxi Supermall
Euclidean distance from th given image: 5.922978852180059
==========



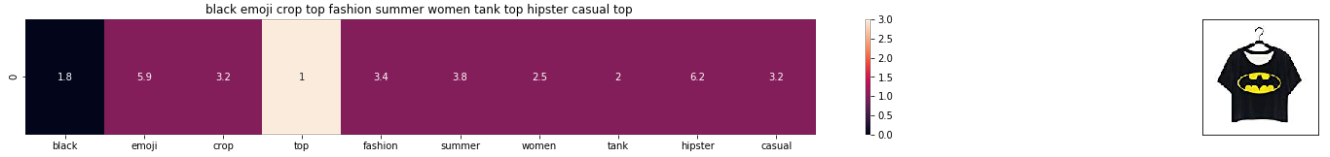
ASIN: B010V3C116
Brand: Doxi Supermall
Euclidean distance from th given image: 7.037272185298332
==========



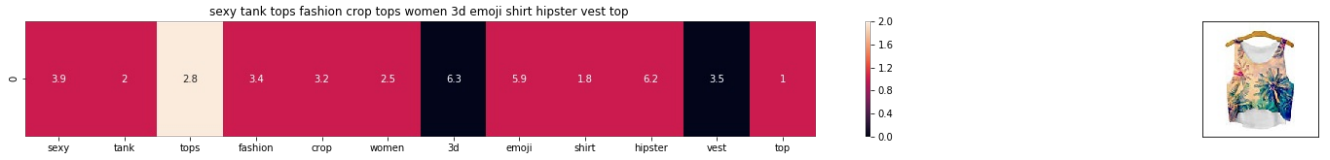
ASIN: B010V3BVMQ
Brand: Doxi Supermall
Euclidean distance from th given image: 7.037272185298332
==========



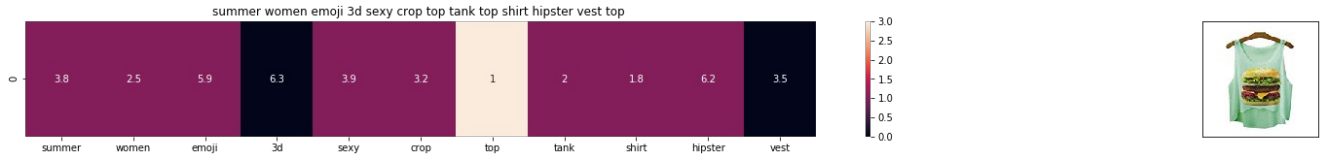
ASIN: B010V3BQZS
Brand: Doxi Supermall
Euclidean distance from th given image: 7.667939547088641
==========



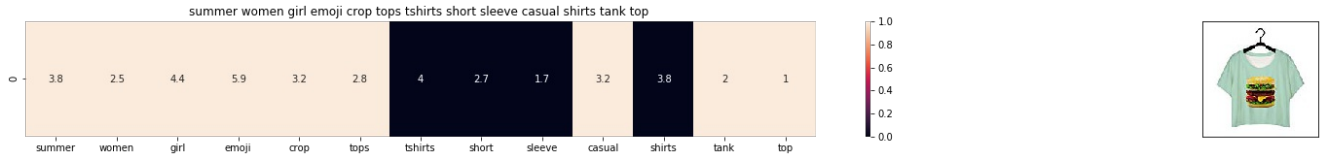
ASIN: B0124E80M4
Brand: Doxi Supermall
Euclidean distance from th given image: 8.39863603811248



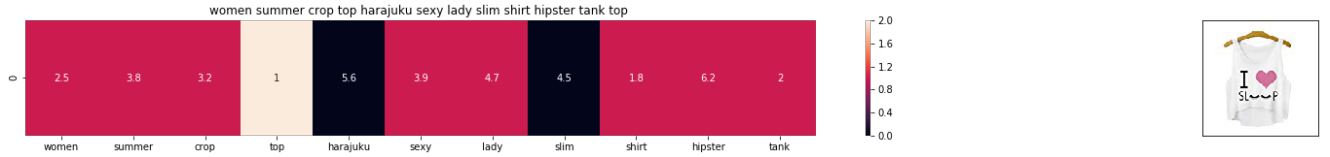
ASIN: B010V3DB9C
Brand: Doxi Supermall
Euclidean distance from th given image: 10.835090137343855



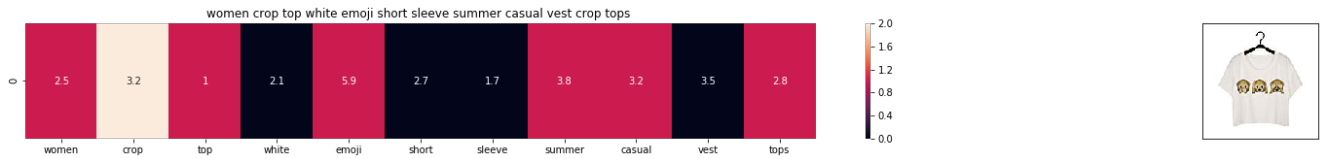
ASIN: B010V3E5EC
Brand: Doxi Supermall
Euclidean distance from th given image: 11.060530175472811



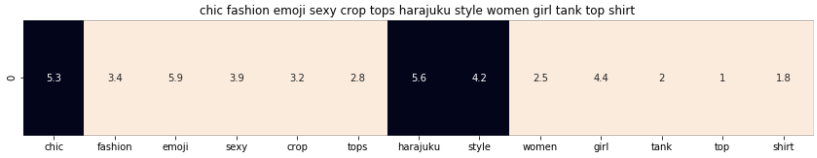
ASIN: B0124ECIU4
Brand: Doxi Supermall
Euclidean distance from th given image: 11.456017724459604



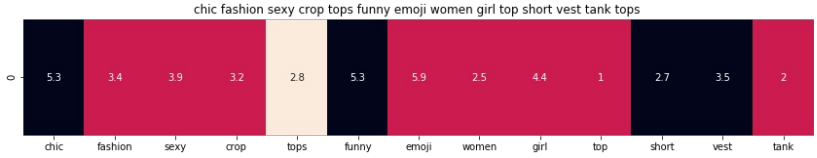
ASIN: B010V3EDEE
Brand: Doxi Supermall
Euclidean distance from th given image: 11.635265990125815



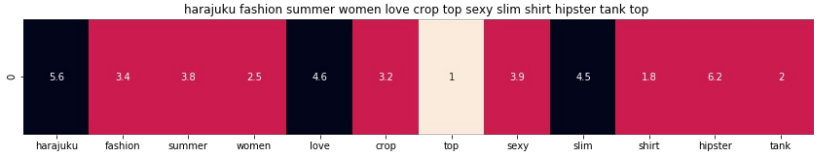
ASIN: B0124E7MHS
Brand: Doxi Supermall
Euclidean distance from th given image: 11.844834283822053
==========



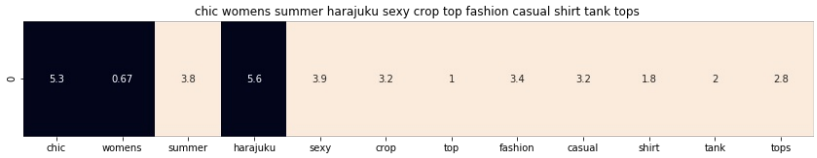
ASIN: B011RCJPR8
Brand: Chiclook Cool
Euclidean distance from th given image: 12.760692810178545
==========



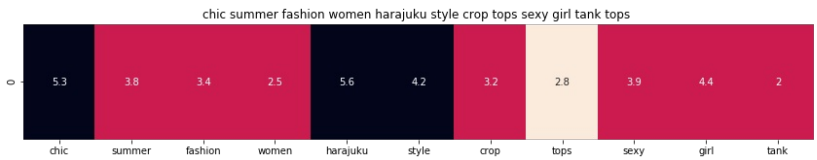
ASIN: B011RCJH58
Brand: Chiclook Cool
Euclidean distance from th given image: 12.829128504563066
==========



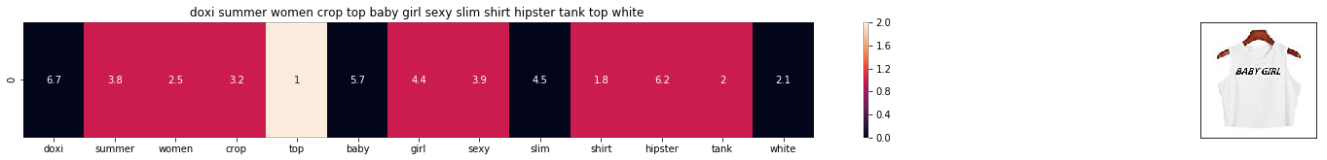
ASIN: B010V35OBU
Brand: Doxi Supermall
Euclidean distance from th given image: 12.92180694297083
==========



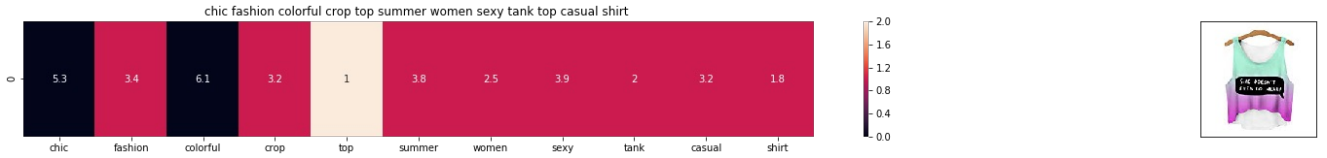
ASIN: B011RCJEMO
Brand: Chiclook Cool
Euclidean distance from th given image: 13.474555736106787
==========



ASIN: B011OU51US
Brand: Chiclook Cool
Euclidean distance from th given image: 13.713807644827545
==========



ASIN: B010V3A23U
Brand: Doxi Supermall
Euclidean distance from th given image: 13.725325741546714
==========



ASIN: B011RCJ6UE
Brand: Chiclook Cool
Euclidean distance from th given image: 13.746787336587824
==========

Text Semantics Based Product Similarity

In [66]:

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

with
open('C:/Users/bhave/Downloads/Applied_AI_Workshop_Code_Data/Applied_AI_Workshop_Code_Data/word2vec
el', 'rb') as handle:
    model = pickle.load(handle)
```

In [67]:

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


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 = get_word_vec(sentence1, doc_id1, model)
    s2_vec = get_word_vec(sentence2, doc_id2, model)
    s1_s2_dist = get_distance(s1_vec, s2_vec)


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

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

    ax = plt.subplot(gs[1])
    ax.grid(False)
    ax.set_xticks([])
    ax.set_yticks([])
    display_img(url, ax, fig)

    plt.show()
```

In [69]:

```
vocab = model.keys()

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

    featureVec = np.zeros((num_features,), dtype="float32")
    # intialize a vector of size 300 with all zeros
    # 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.voc
abulary_[word]] * model[word])
            elif m_name == 'avg':
                featureVec = np.add(featureVec, model[word])
    if(nwords>0):
        featureVec = np.divide(featureVec, nwords)
```

```
    # return the avg vector of given sentance, its of shape (1, 300)
    return featureVec
```

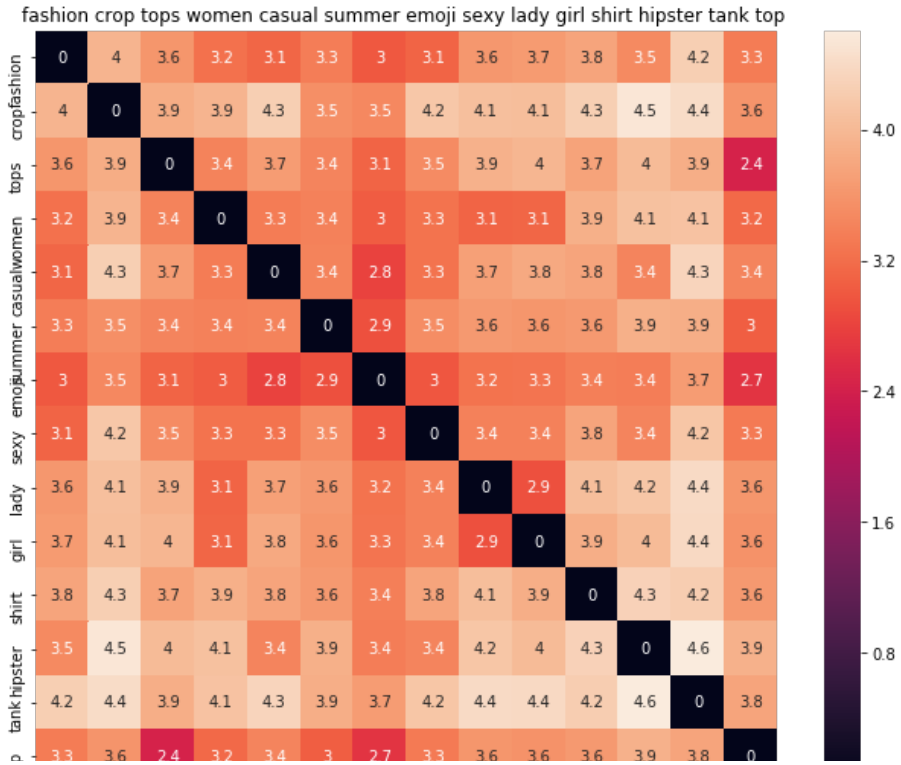
Average Word2Vec Produt Similarity

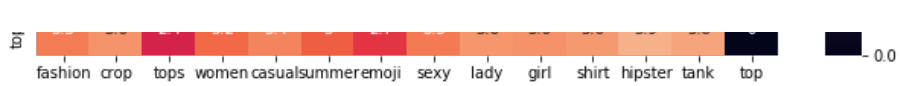
In [70]:

```
doc_id = 0
w2v_title = []

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 is a doc
w2v_title = np.array(w2v_title)
```

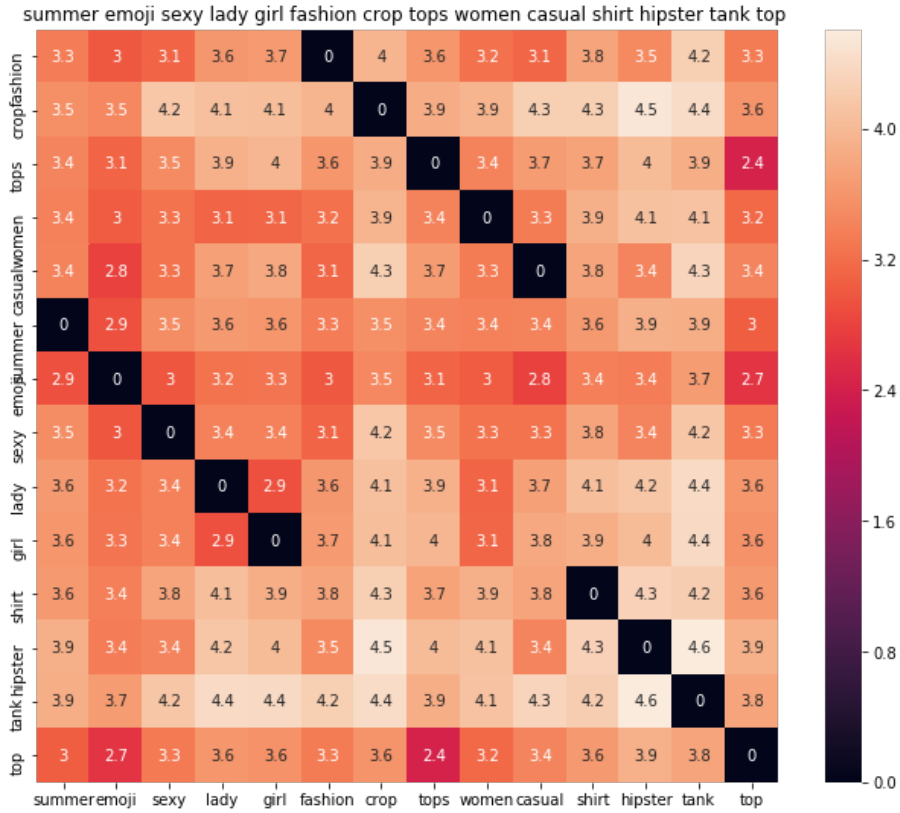
In [71]:

```
def avg_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in 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 returns indices of 9 smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists stores 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['mediu
m_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
```

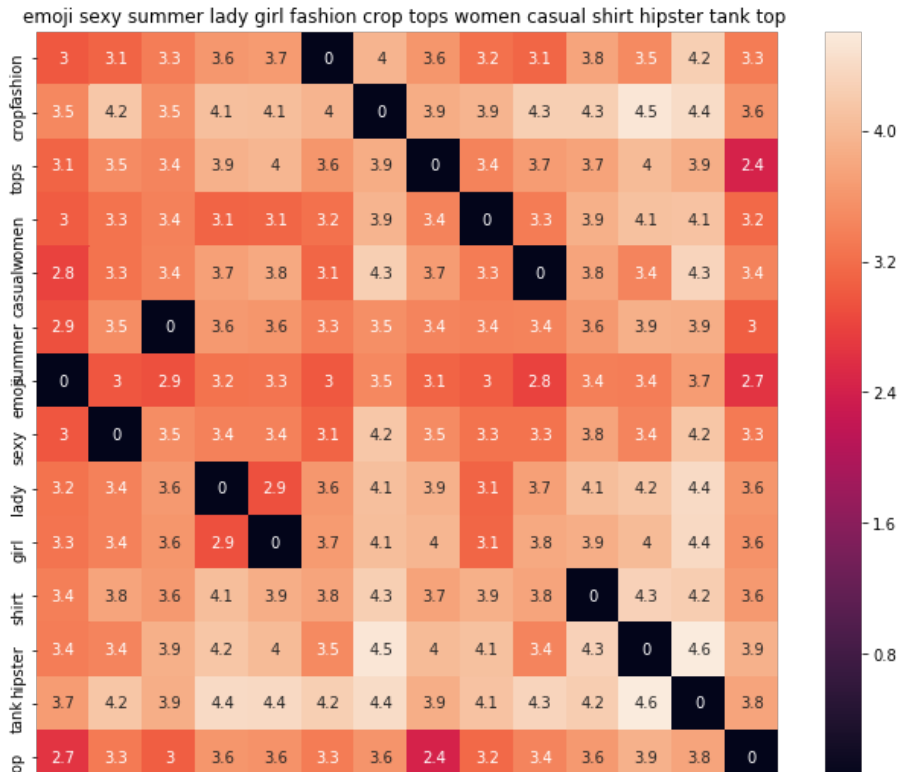




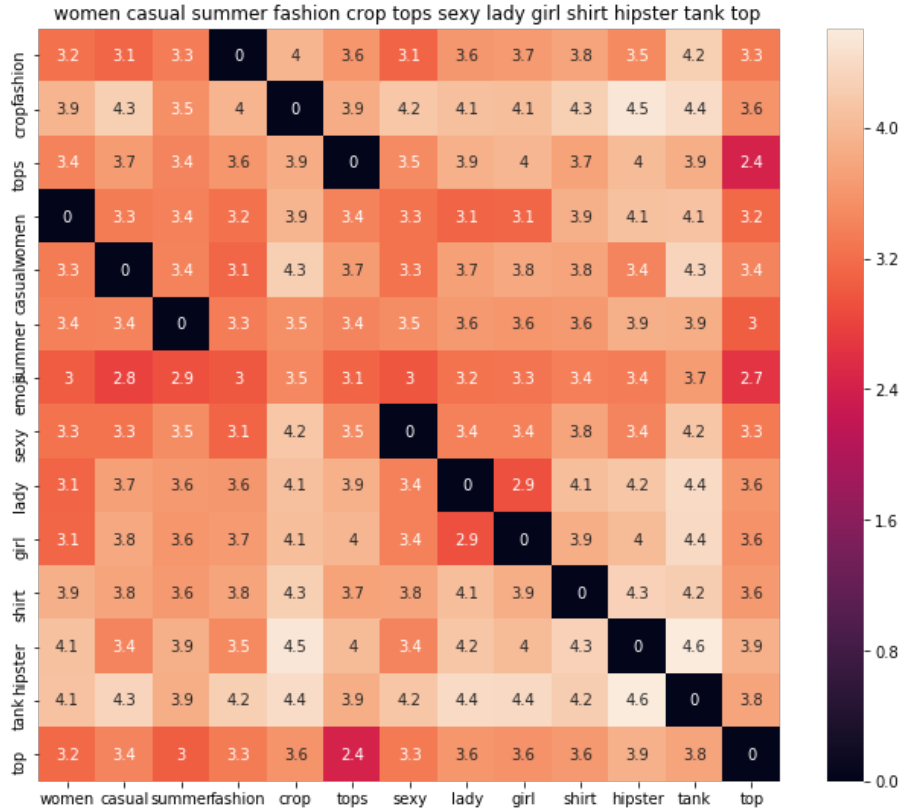
ASIN : B010V3B44G
BRAND : Doxi Supermall
euclidean distance from given input image : 0.0
==========



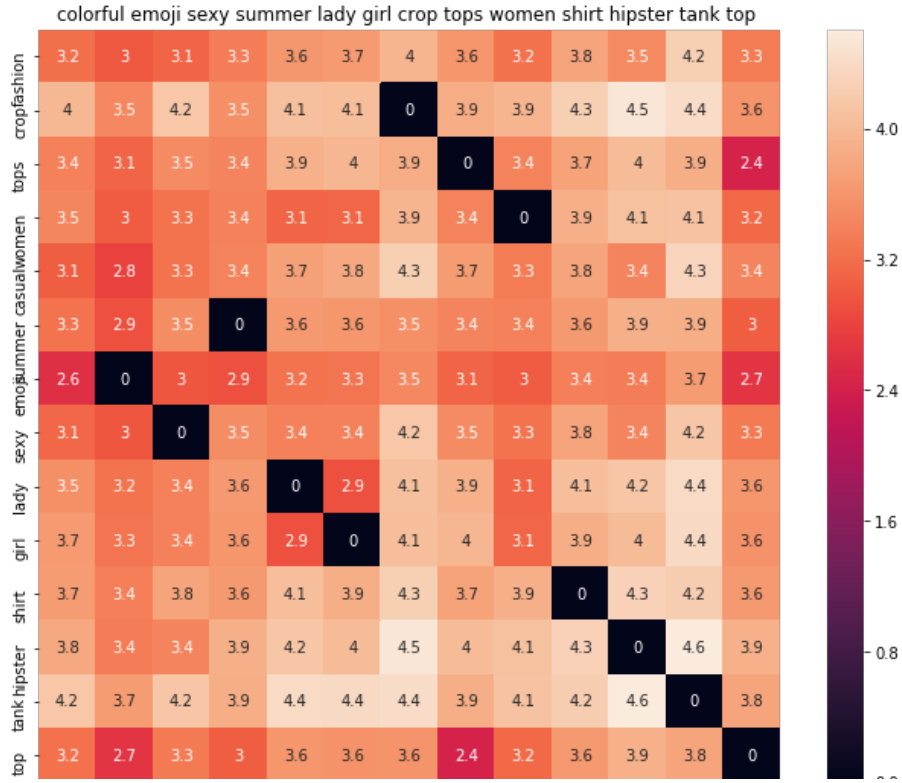
ASIN : B010V3BDII
BRAND : Doxi Supermall
euclidean distance from given input image : 0.0
==========



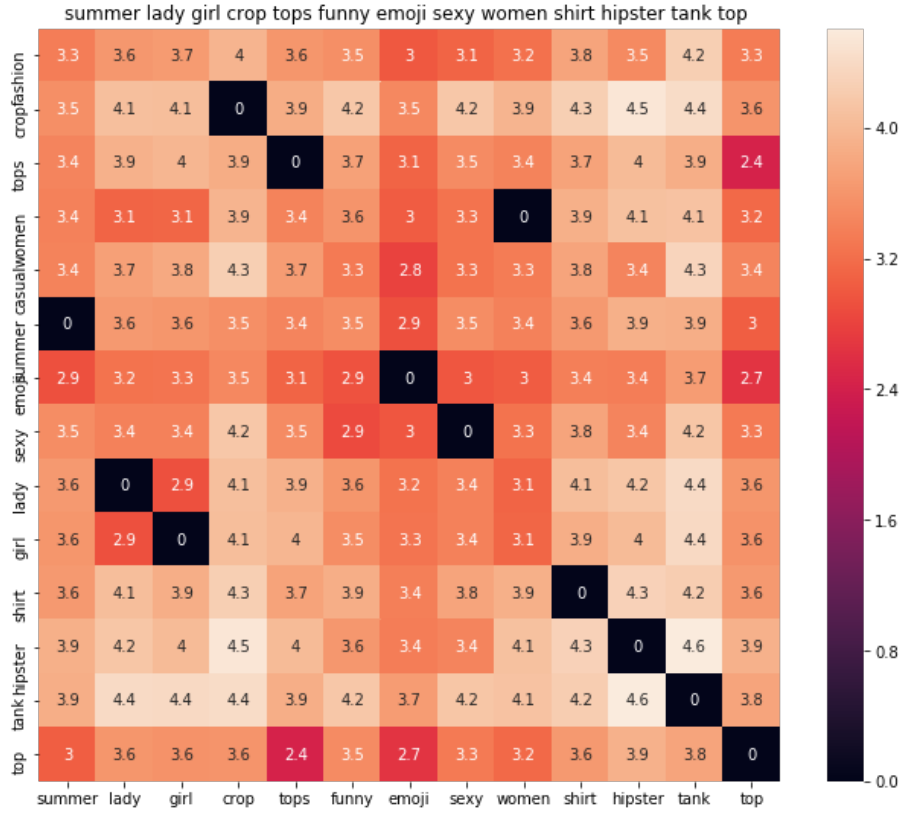
ASIN : B010V3BLWQ
BRAND : Doxi Supermall
euclidean distance from given input image : 0.0
==========



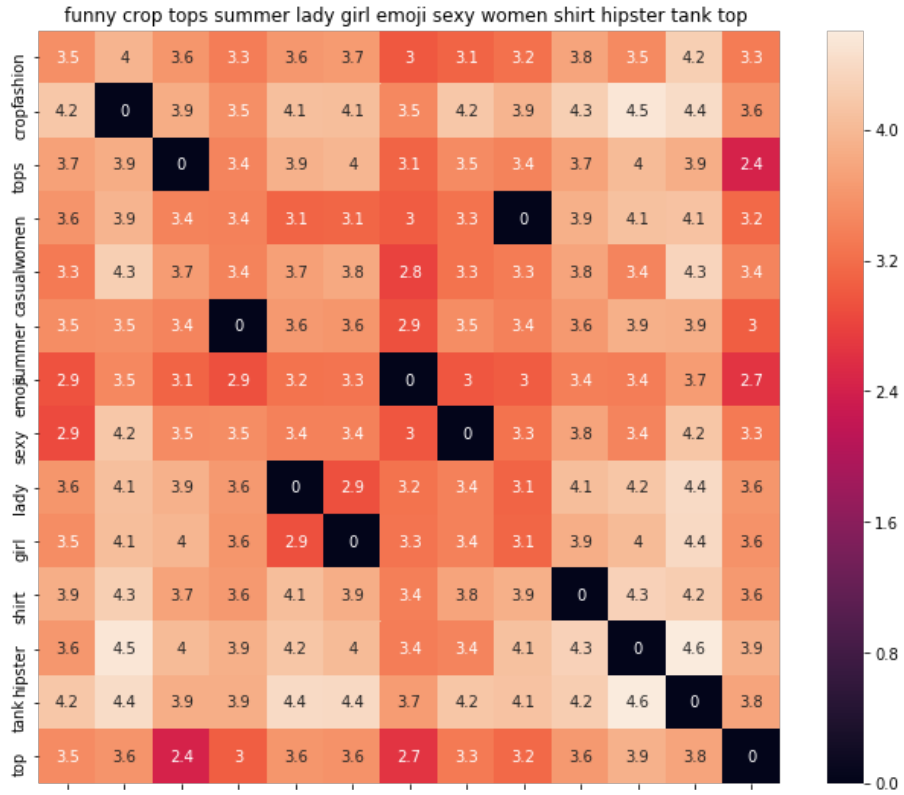
ASIN : B010V3AYSS
BRAND : Doxi Supermall
euclidean distance from given input image : 0.1336821
==========



ASIN : B010V3BQZS
BRAND : Doxi Supermall
euclidean distance from given input image : 0.298377

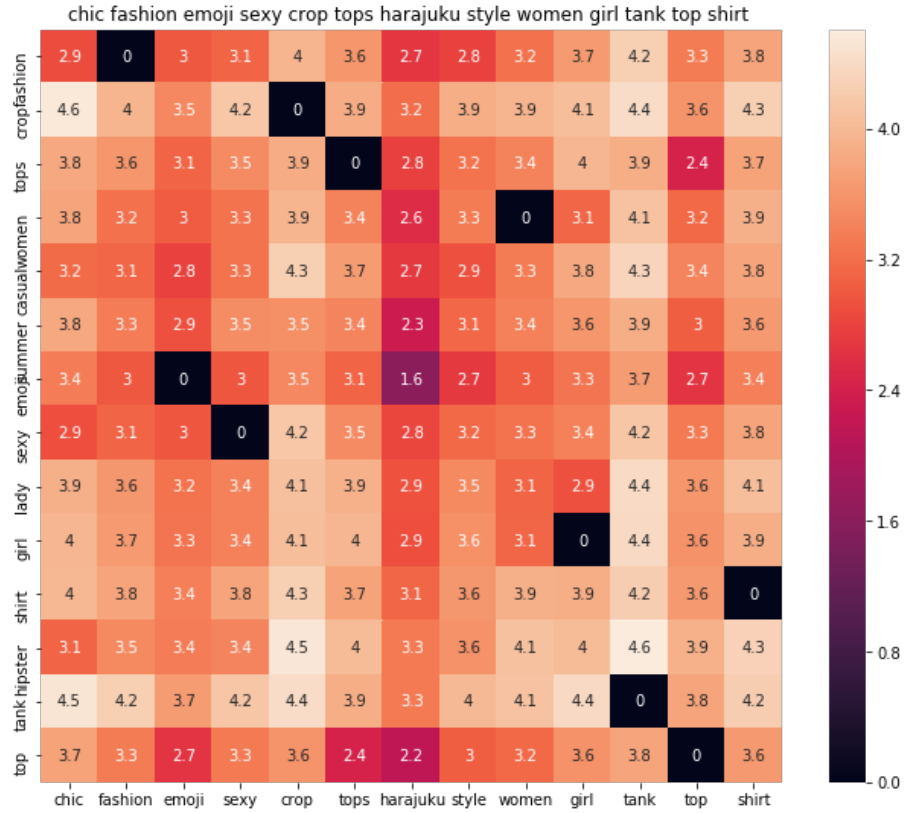


ASIN : B010V3BVMQ
BRAND : Doxi Supermall
euclidean distance from given input image : 0.3274633

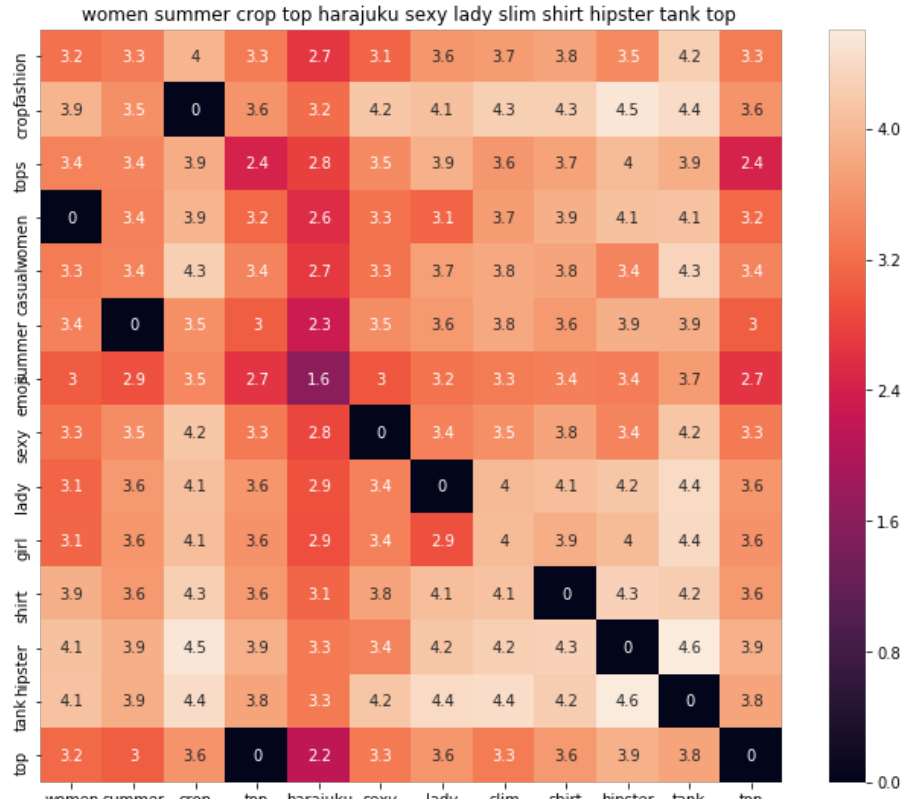


funny crop tops summer lady girl emoji sexy women shirt hipster tank top

ASIN : B010V3C116
BRAND : Doxi Supermall
euclidean distance from given input image : 0.3274633

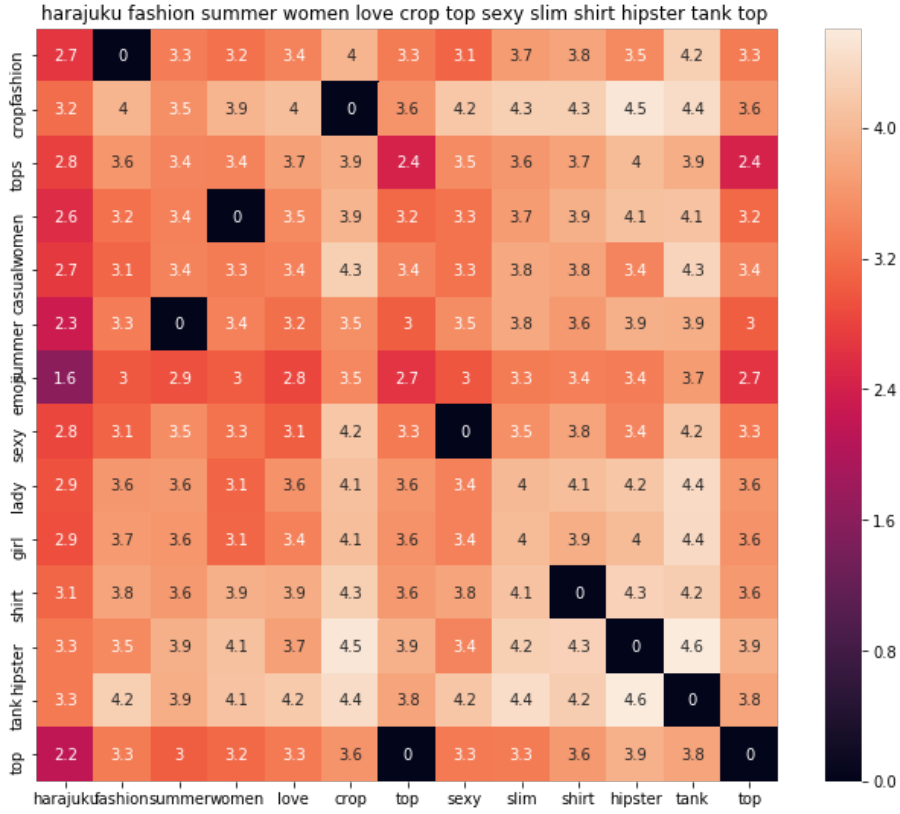


ASIN : B011RCJPR8
BRAND : Chiclook Cool
euclidean distance from given input image : 0.4097586

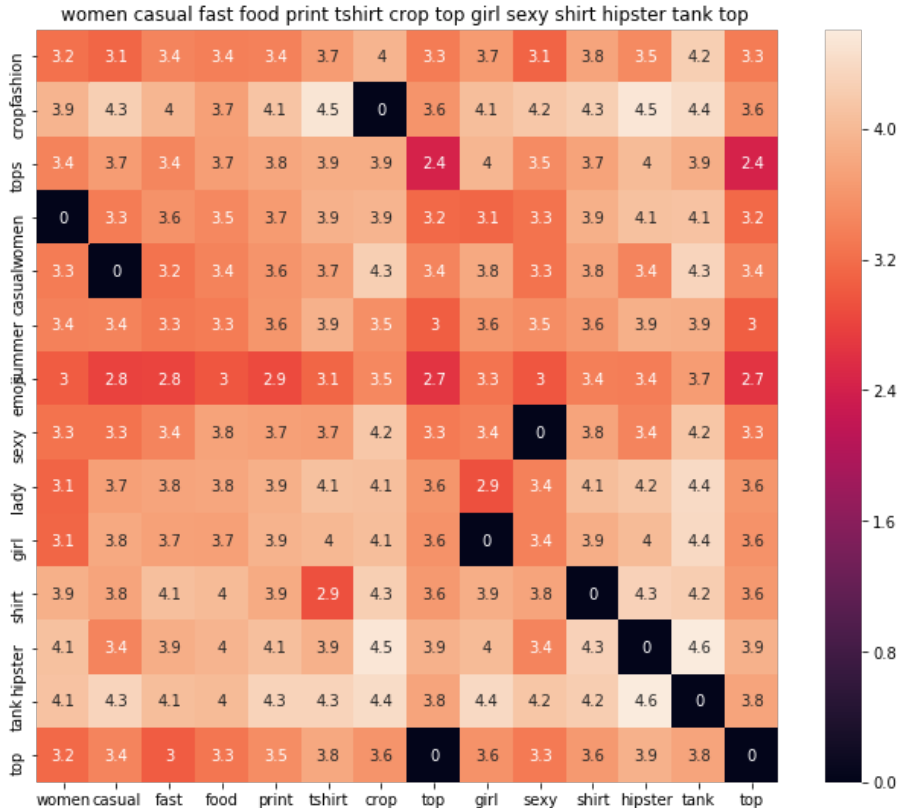


women summer crop top harajuku sexy lady slim shirt hipster tank top

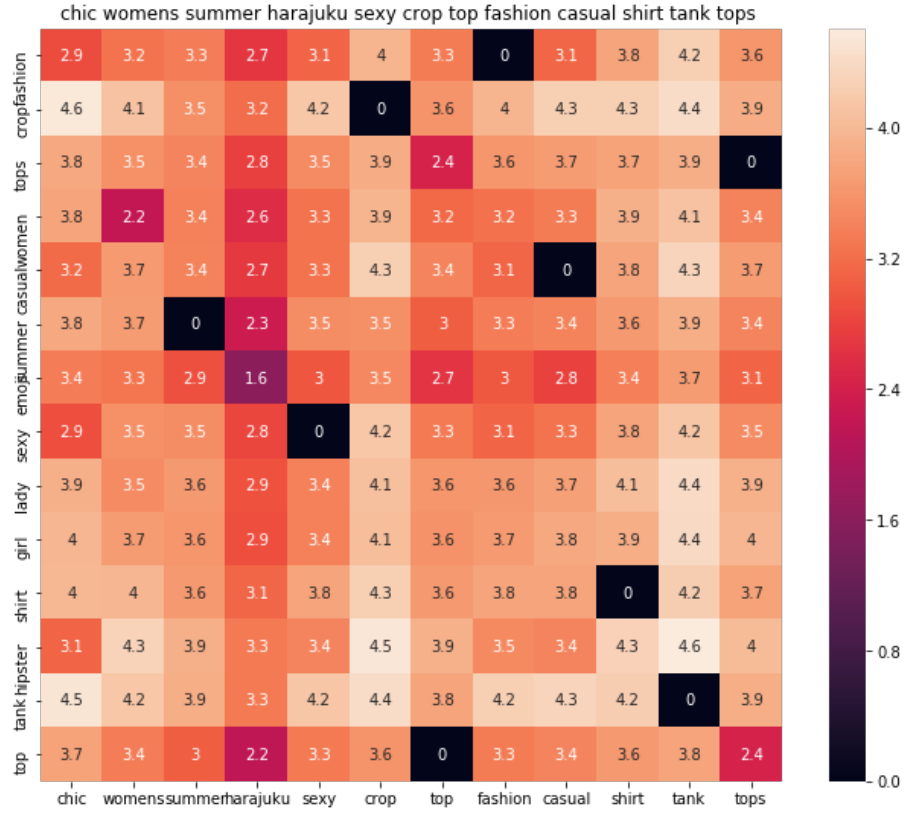
ASIN : B010V3EDEE
BRAND : Doxi Supermall
euclidean distance from given input image : 0.46806592
==========



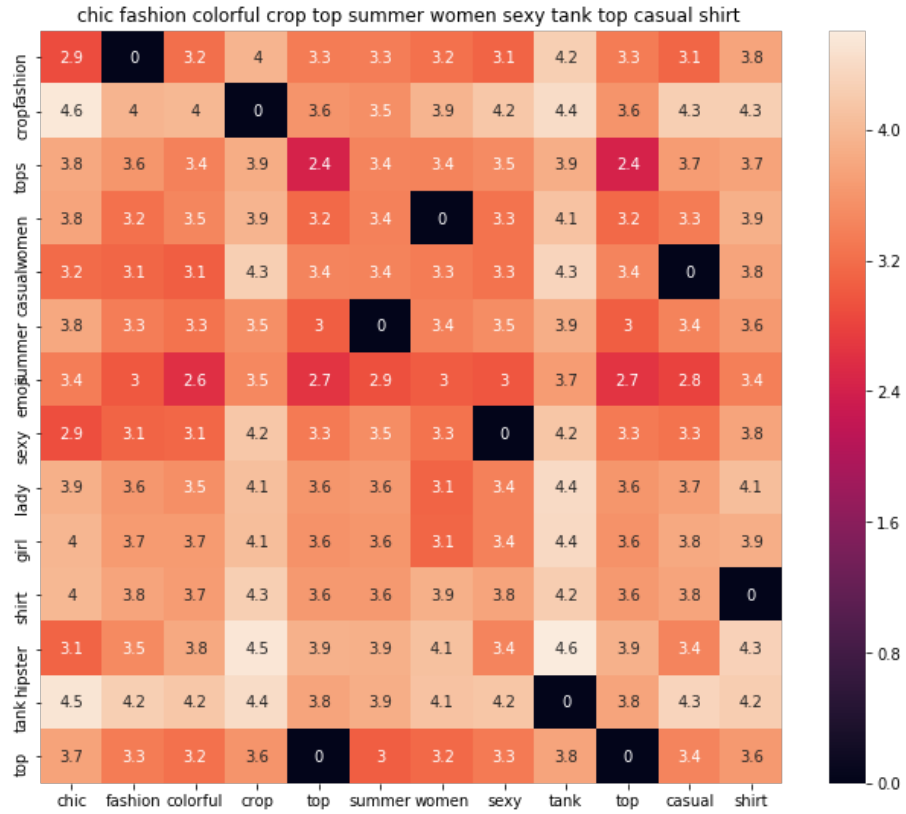
ASIN : B010V35OBU
BRAND : Doxi Supermall
euclidean distance from given input image : 0.4957315
==========



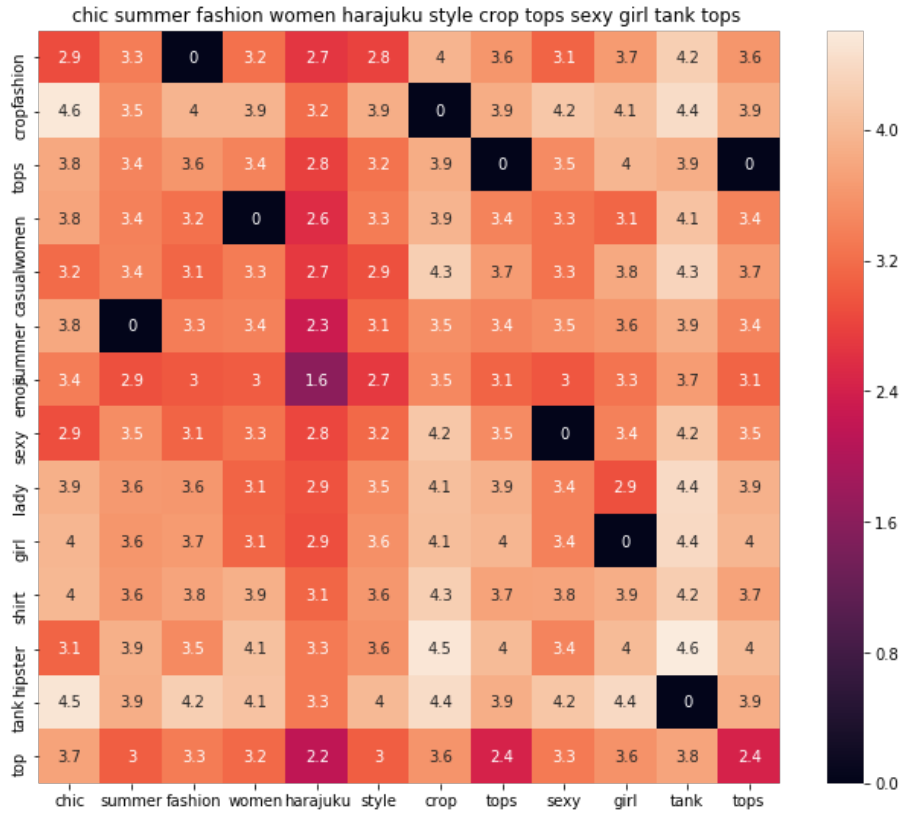
ASIN : B010V3AB5O
BRAND : Doxi Supermall
euclidean distance from given input image : 0.50107557
==========
==========



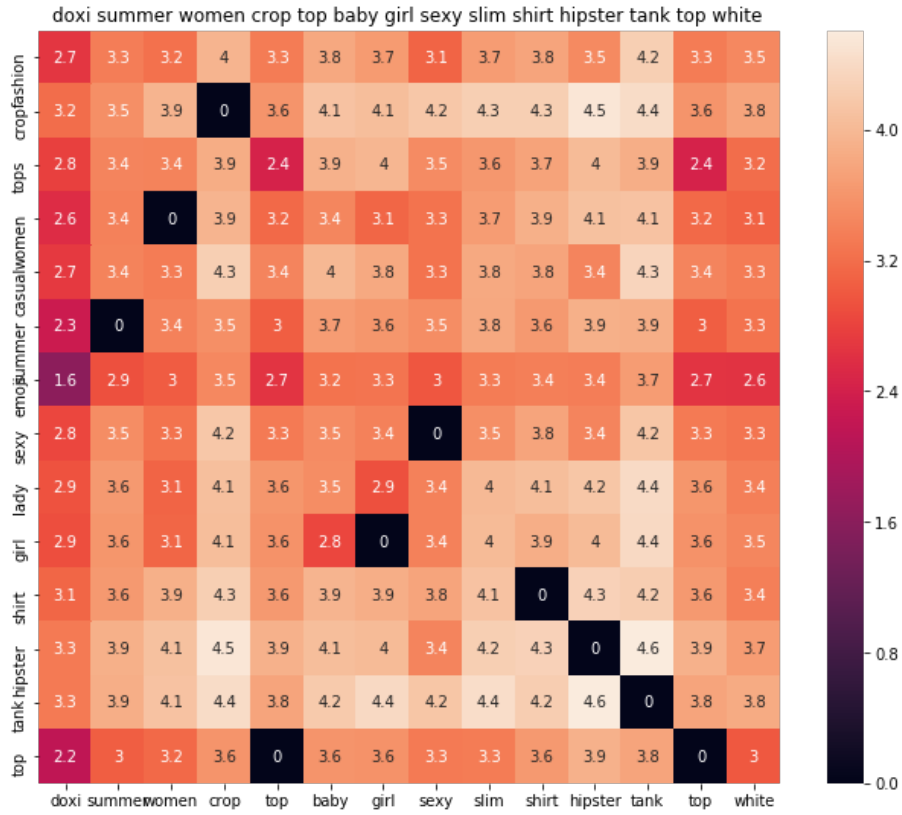
ASIN : B011RCJEMO
BRAND : Chiclook Cool
euclidean distance from given input image : 0.50272816
==========
==========



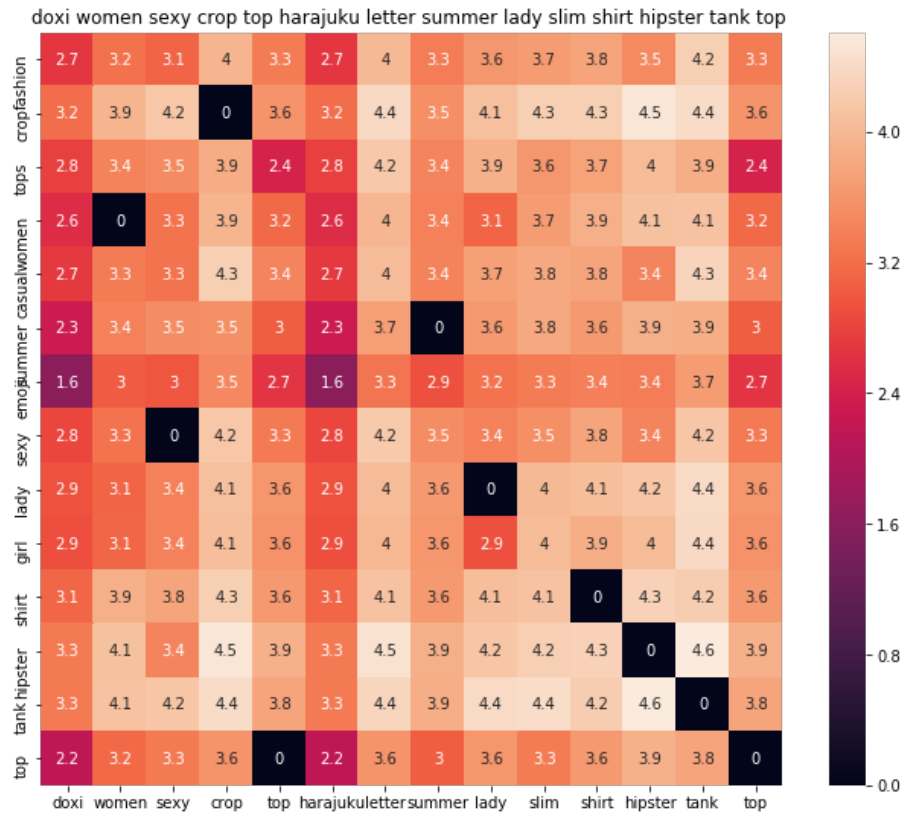
ASIN : B011RCJ6UE
BRAND : Chiclook Cool
euclidean distance from given input image : 0.50736403



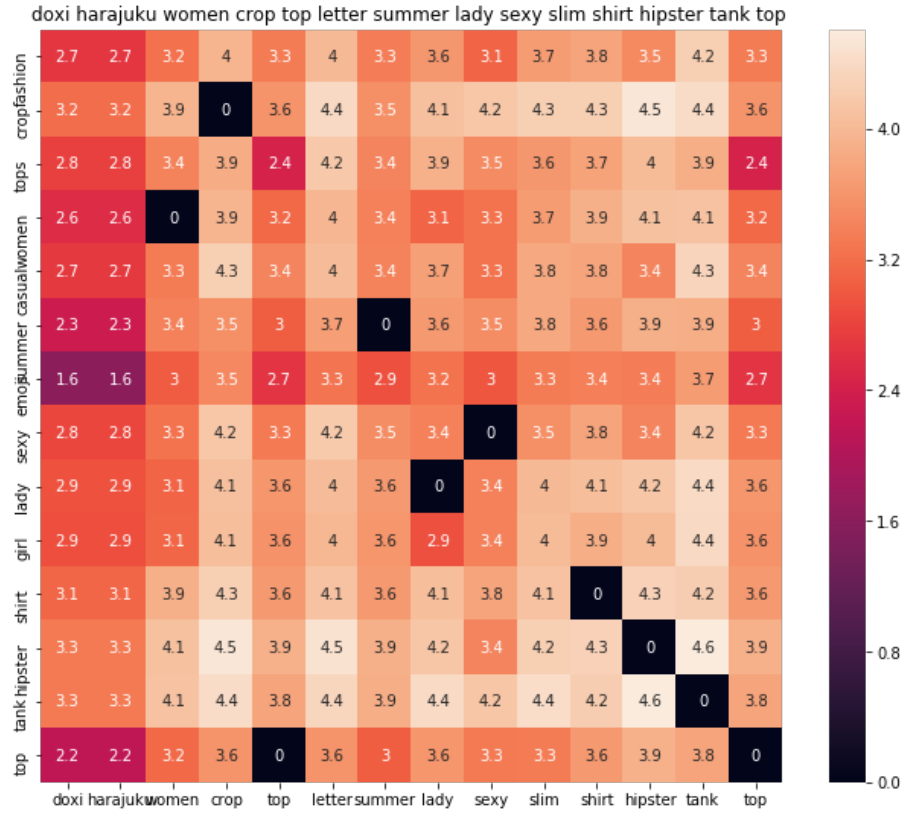
ASIN : B011OU51US
BRAND : Chiclook Cool
euclidean distance from given input image : 0.5160888



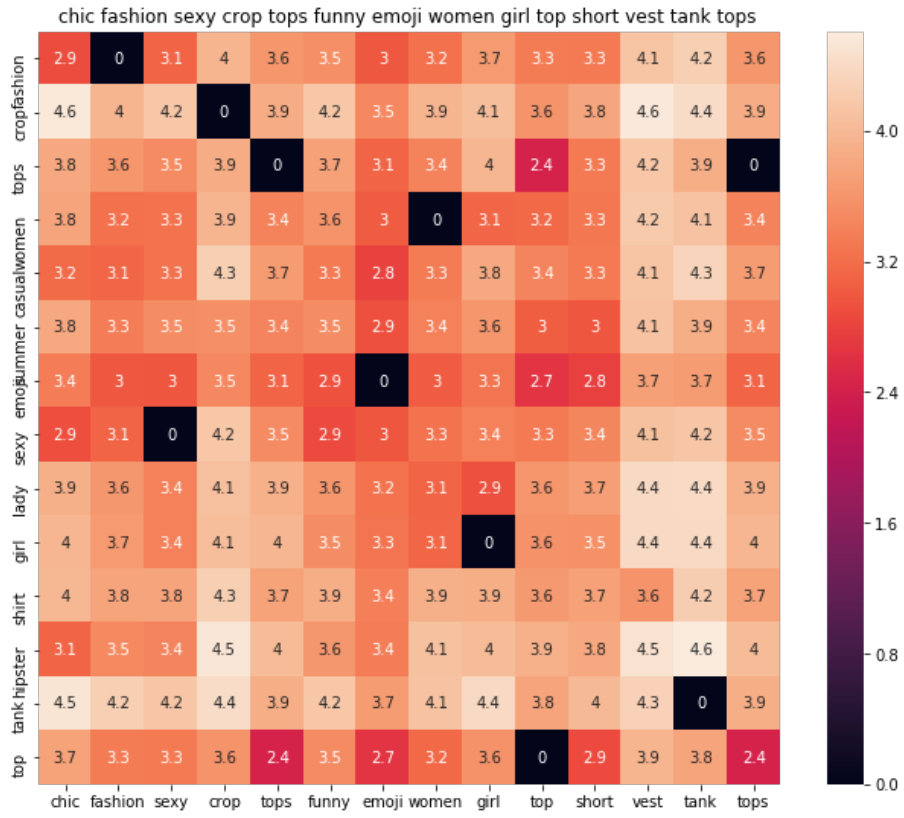
ASIN : B010V3A23U
BRAND : Doxi Supermall
euclidean distance from given input image : 0.5262786



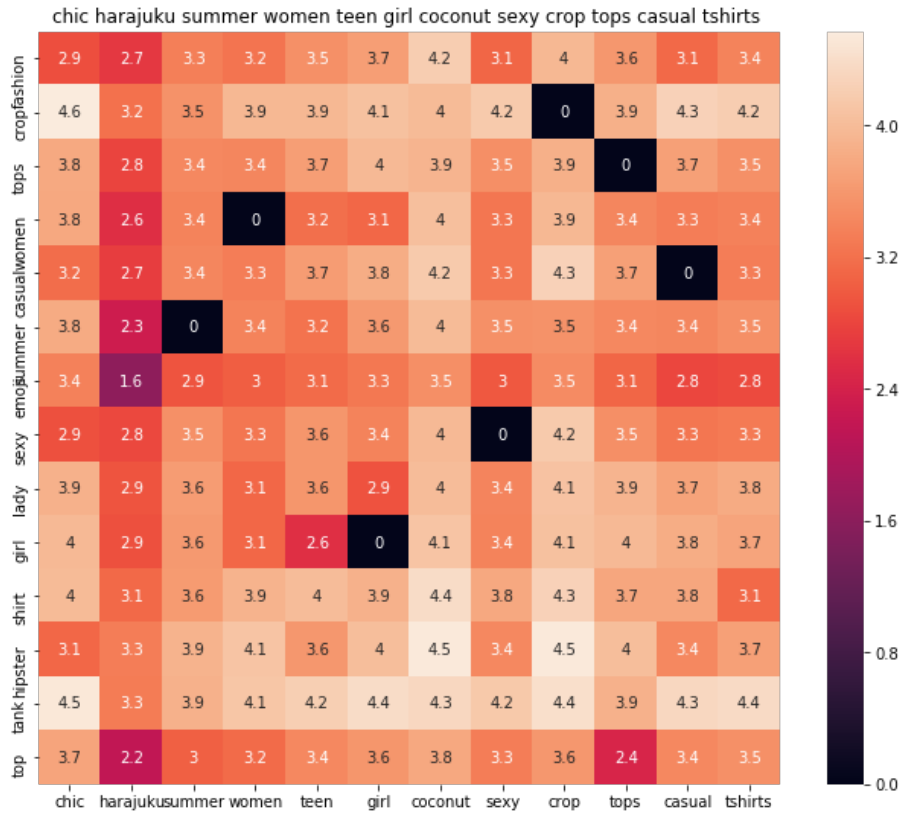
ASIN : B010V39146
BRAND : Doxi Supermall
euclidean distance from given input image : 0.5270717



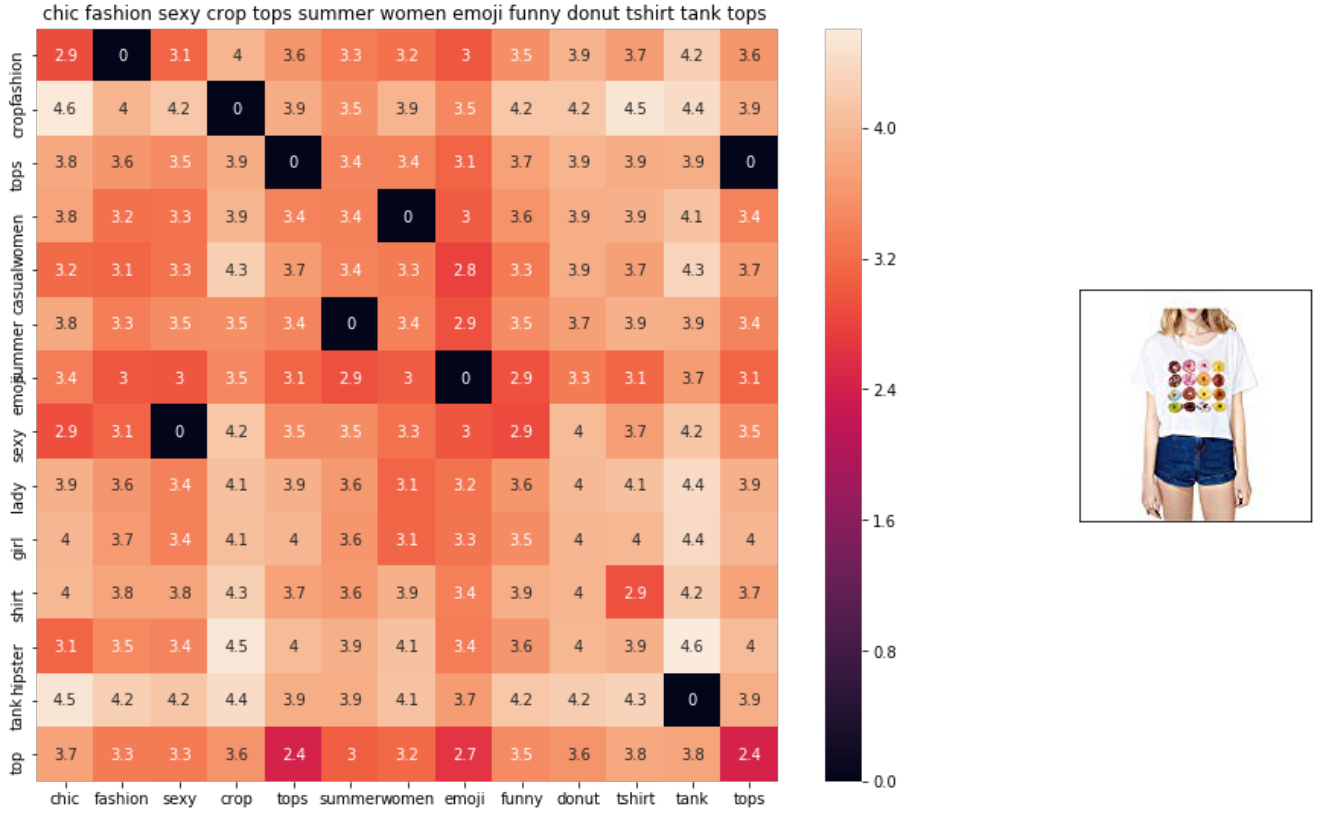
ASIN : B010V380LQ
BRAND : Doxi Supermall
euclidean distance from given input image : 0.5270717
==========
==========



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
==========================================================================================================
======================
```

IDF Weighted Word2Vec for Product Similarity

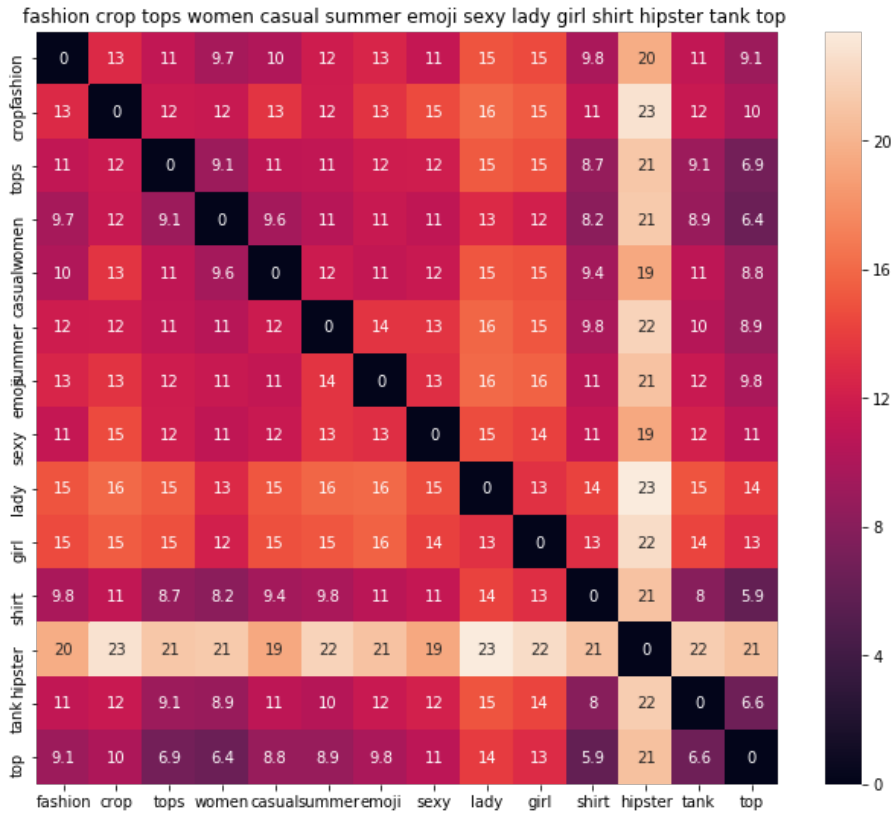
In [72]:

```
doc_id = 0
w2v_title_weight = []
for i in data['title']:
    w2v_title_weight.append(build_avg_vec(i, 300, doc_id,'weighted'))
    doc_id += 1
w2v_title_weight = np.array(w2v_title_weight)
```

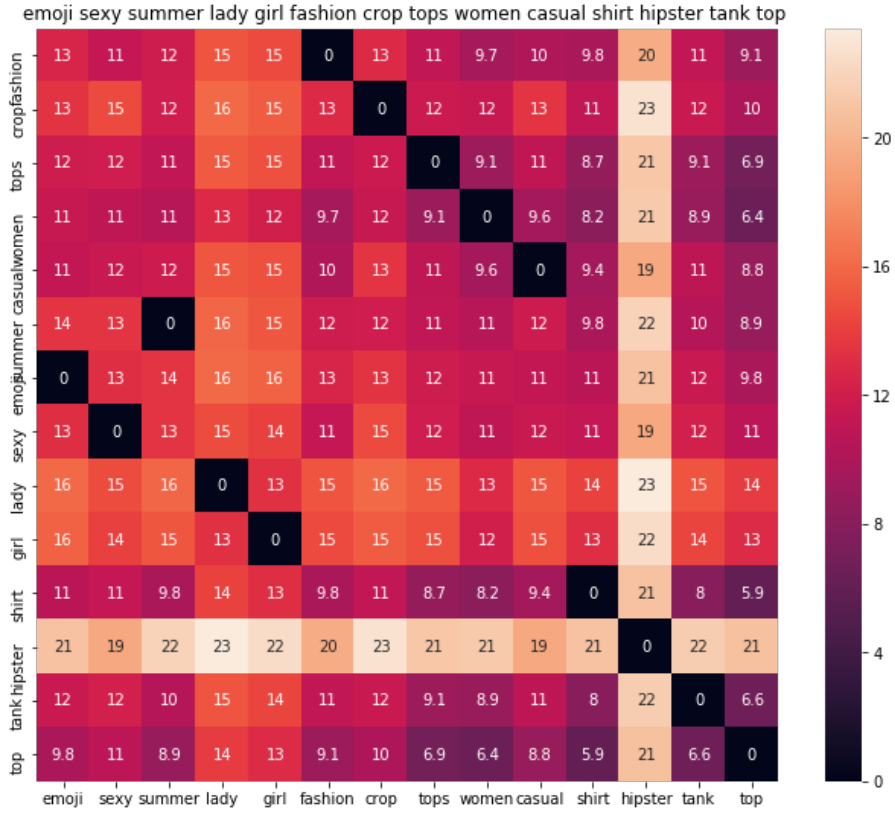
In [73]:

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

    pairwise_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]
    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['mediu
m_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)
```

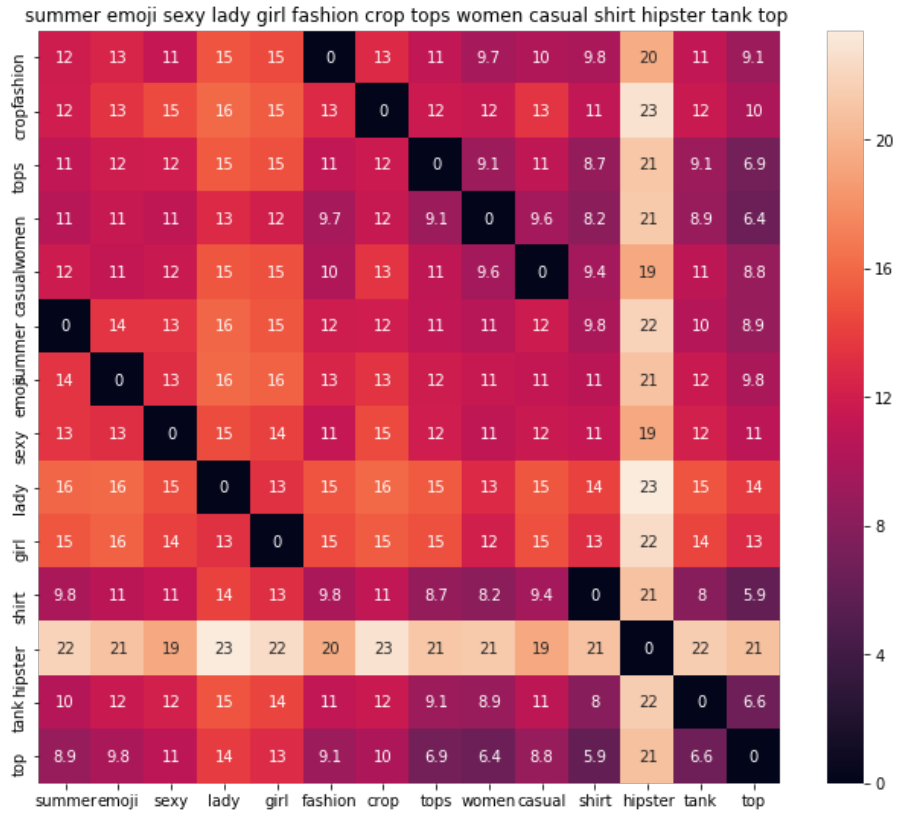



```
ASIN : B010V3B44G
Brand : Doxi Supermall
euclidean distance from input : 0.001953125
================================================================================================================
======================
```



```
ASIN : B010V3BLWQ
Brand : Doxi Supermall
euclidean distance from input : 0.001953125
================================================================================================================
```

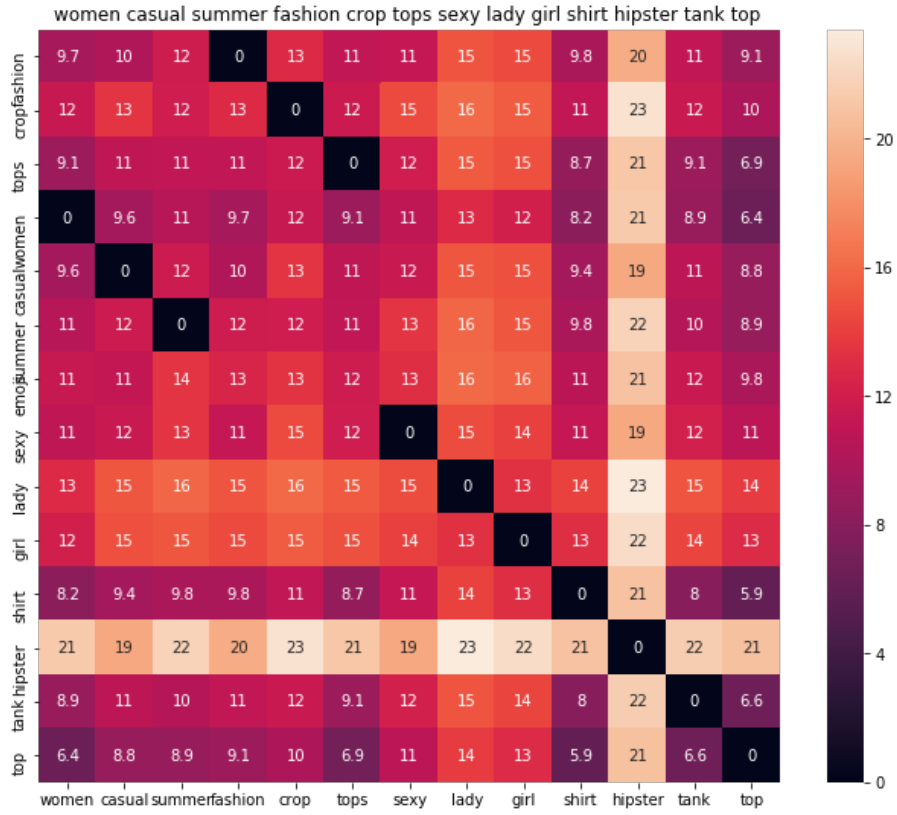
==========



ASIN : B010V3BDII
Brand : Doxi Supermall
euclidean distance from input : 0.001953125

==========

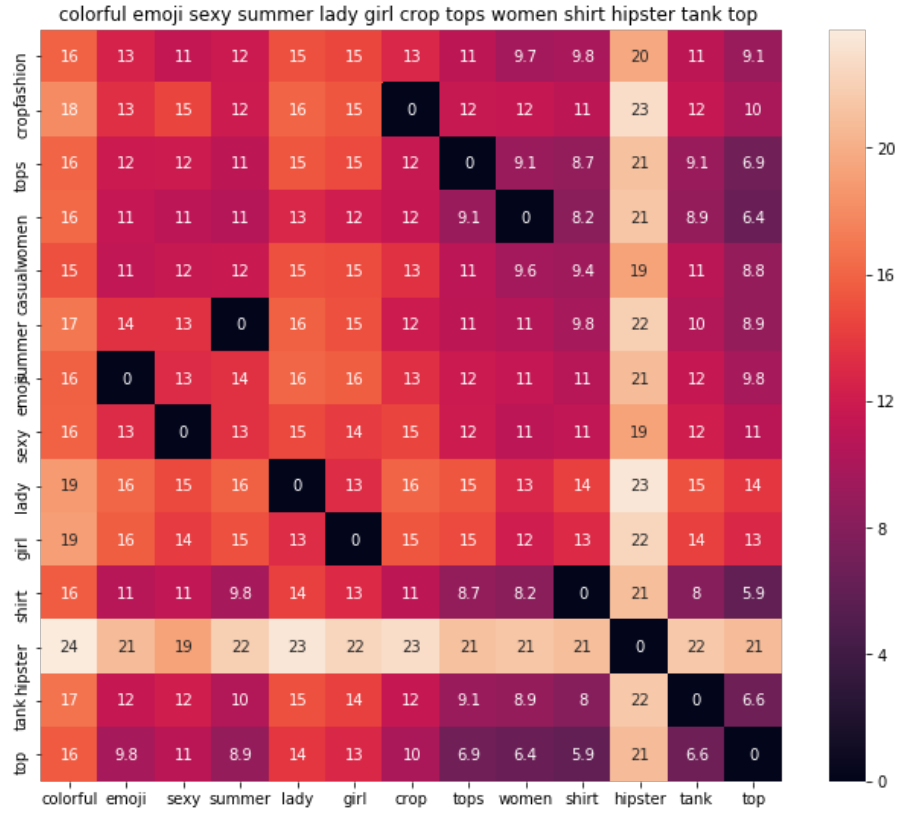
==========



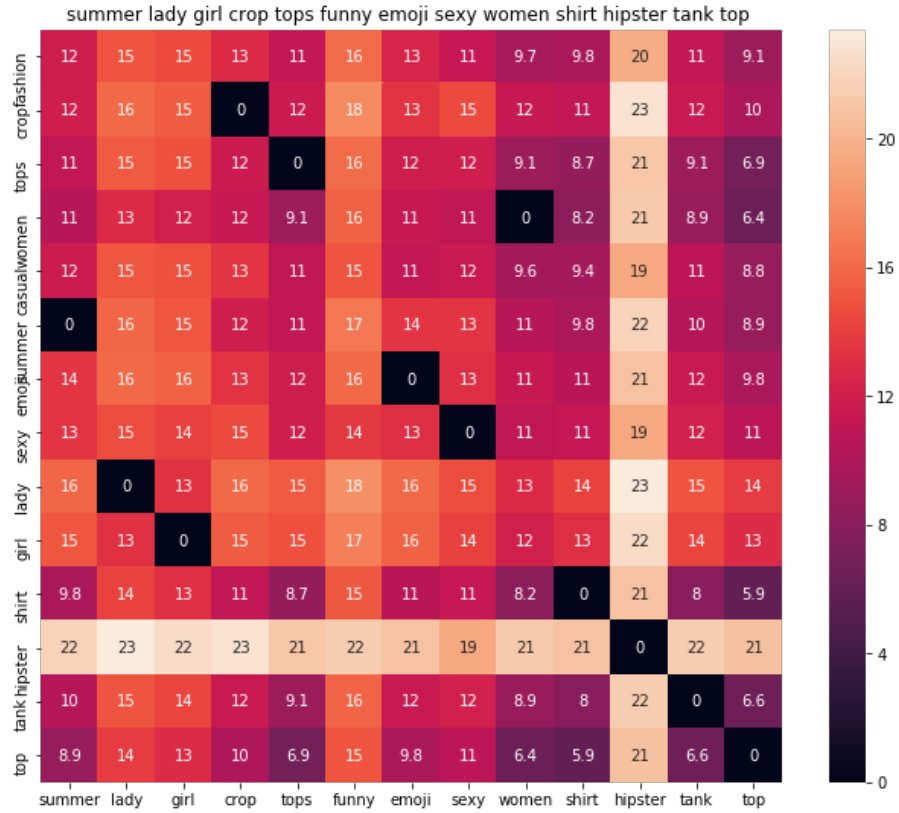
ASIN : B010V3AYSS
Brand : Doxi Supermall
euclidean distance from input : 0.6962308

==========

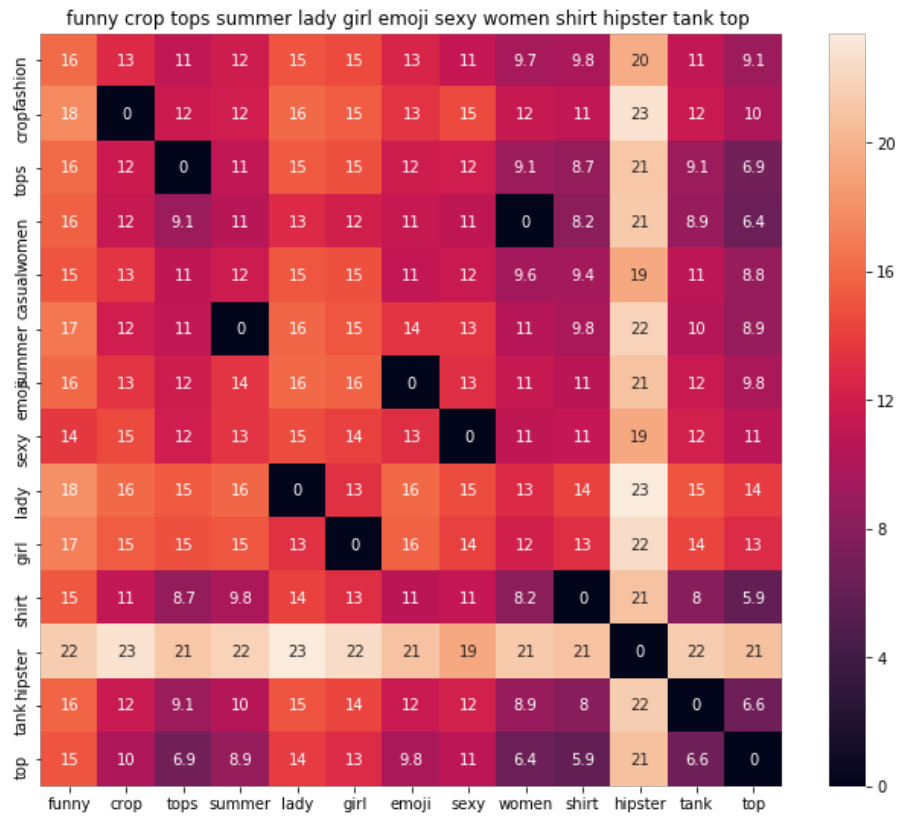
==========



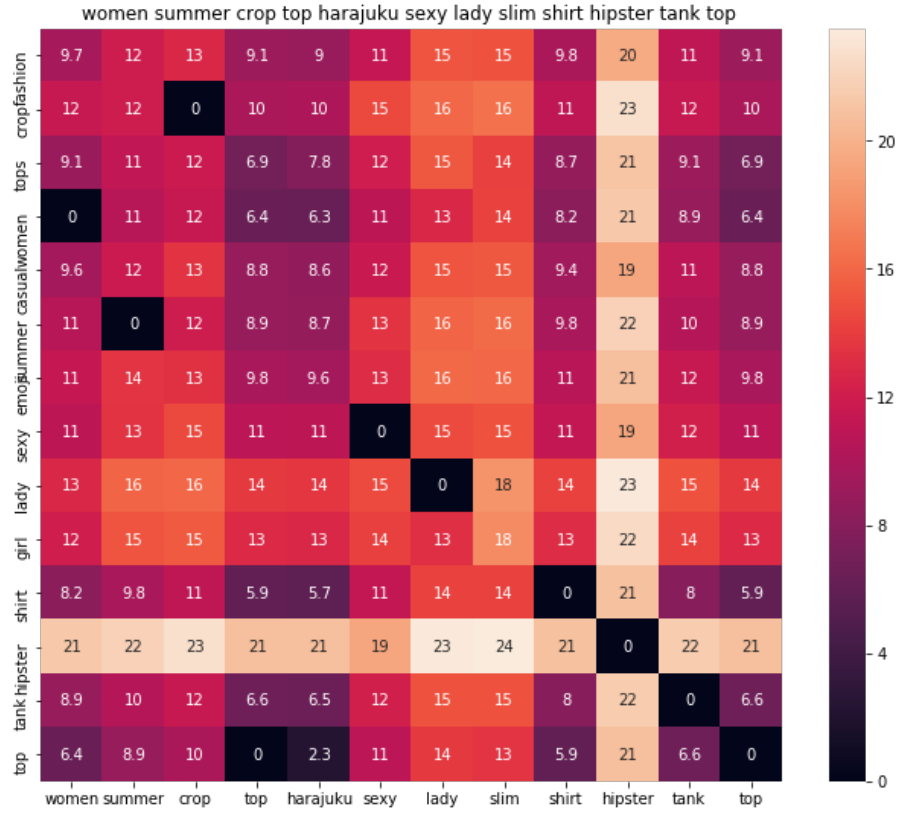
ASIN : B010V3BQZS
Brand : Doxi Supermall
euclidean distance from input : 1.2872716



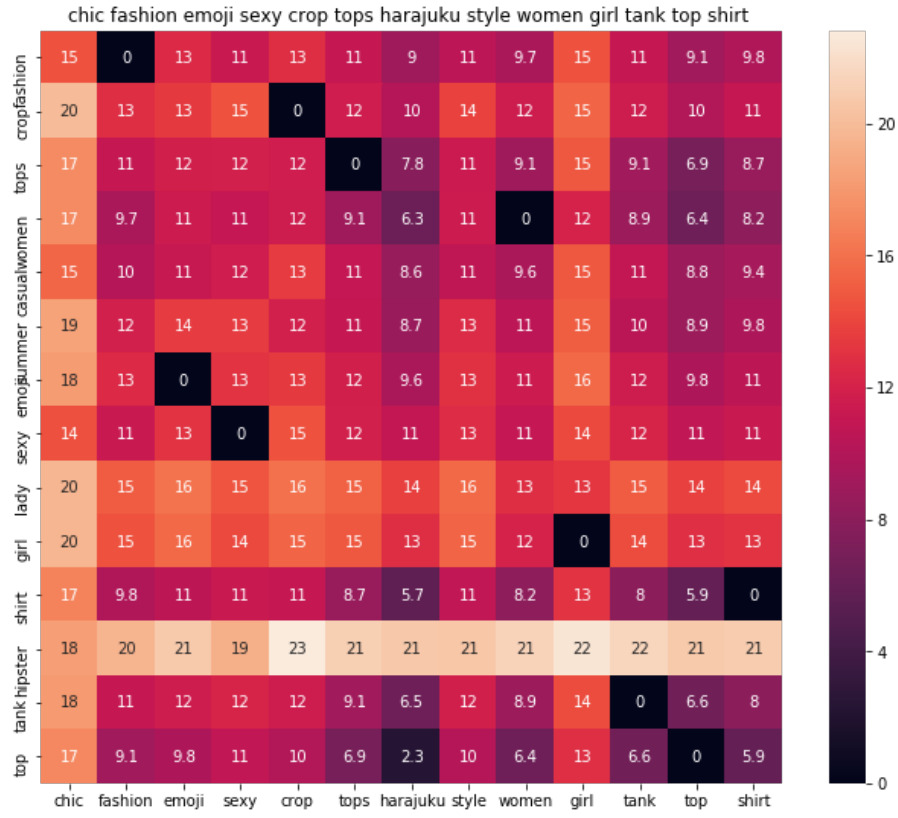
ASIN : B010V3BVMQ
Brand : Doxi Supermall
euclidean distance from input : 1.3530484



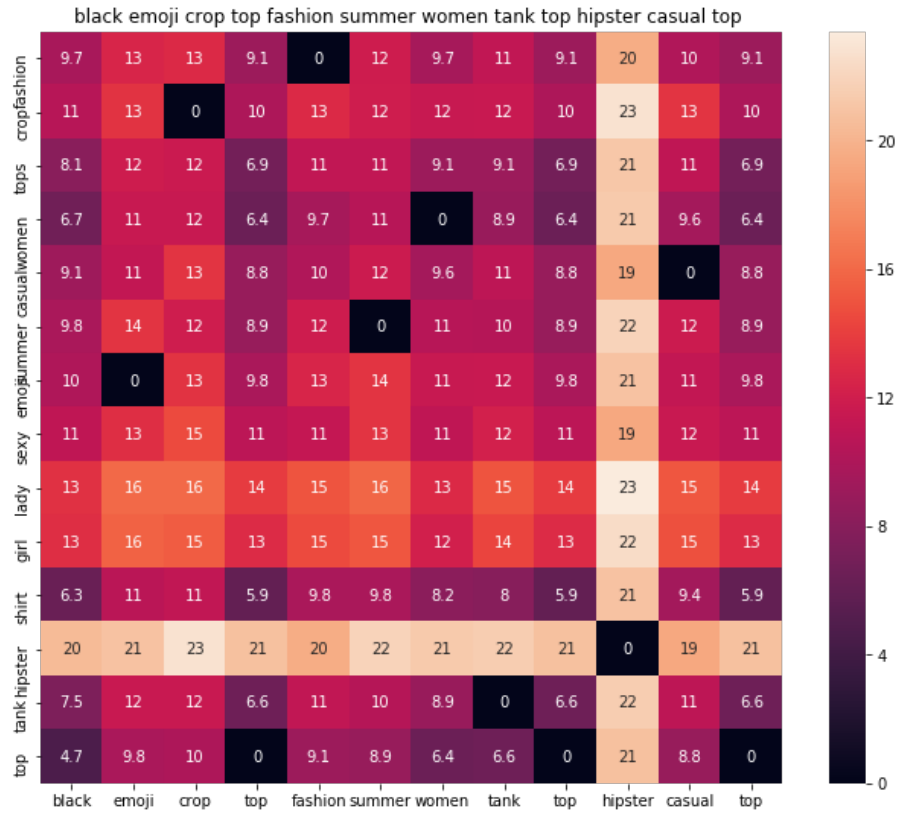
ASIN : B010V3C116
Brand : Doxi Supermall
euclidean distance from input : 1.3530484



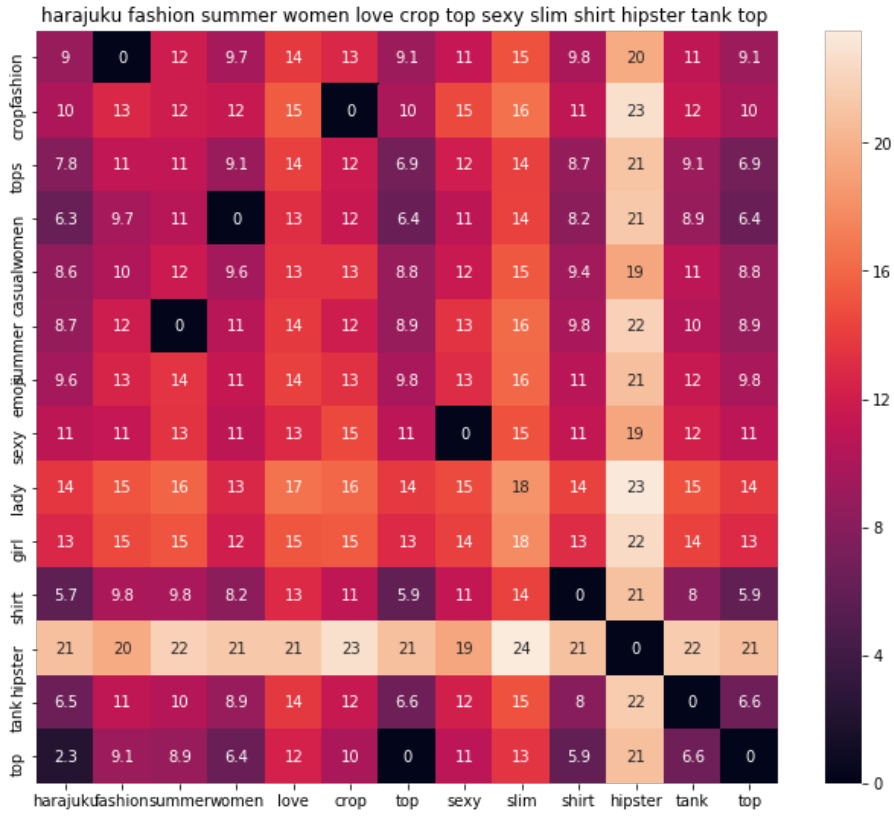
ASIN : B010V3EDEE
Brand : Doxi Supermall
euclidean distance from input : 1.7434149



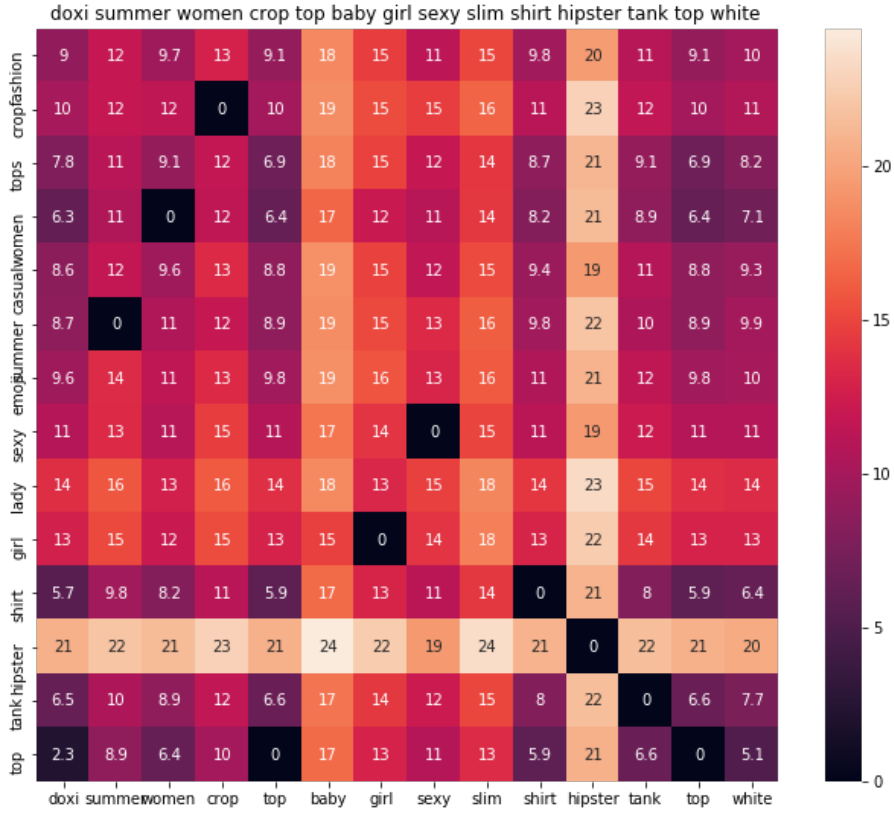
ASIN : B011RCJPR8
Brand : Chiclook Cool
euclidean distance from input : 1.9345262



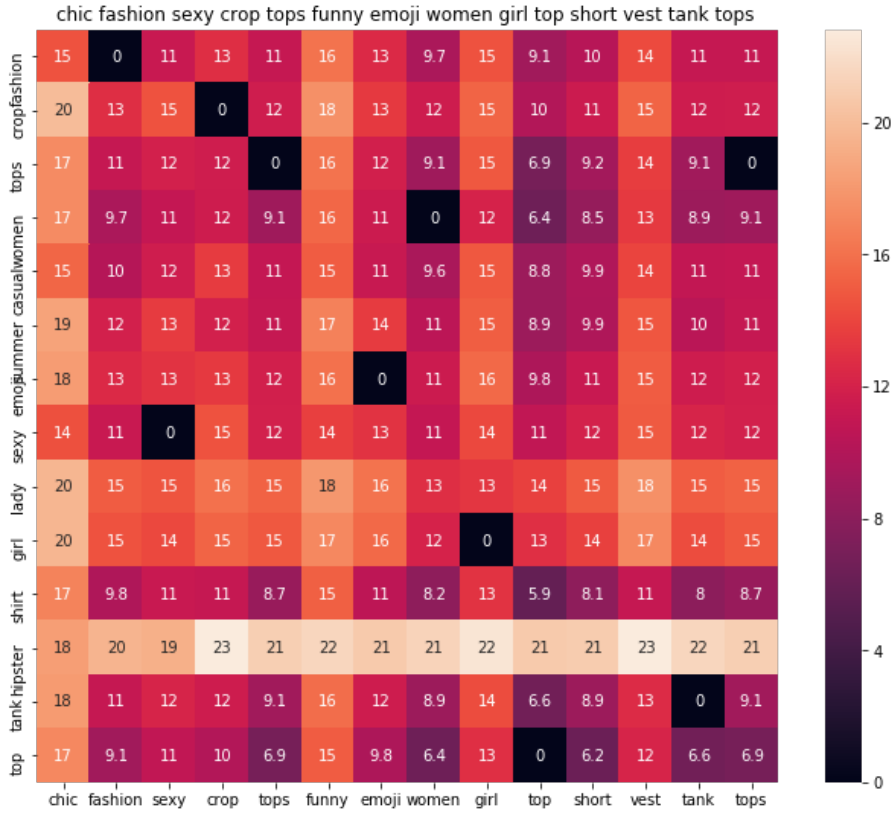
ASIN : B0124E80M4
Brand : Doxi Supermall
euclidean distance from input : 2.0321443



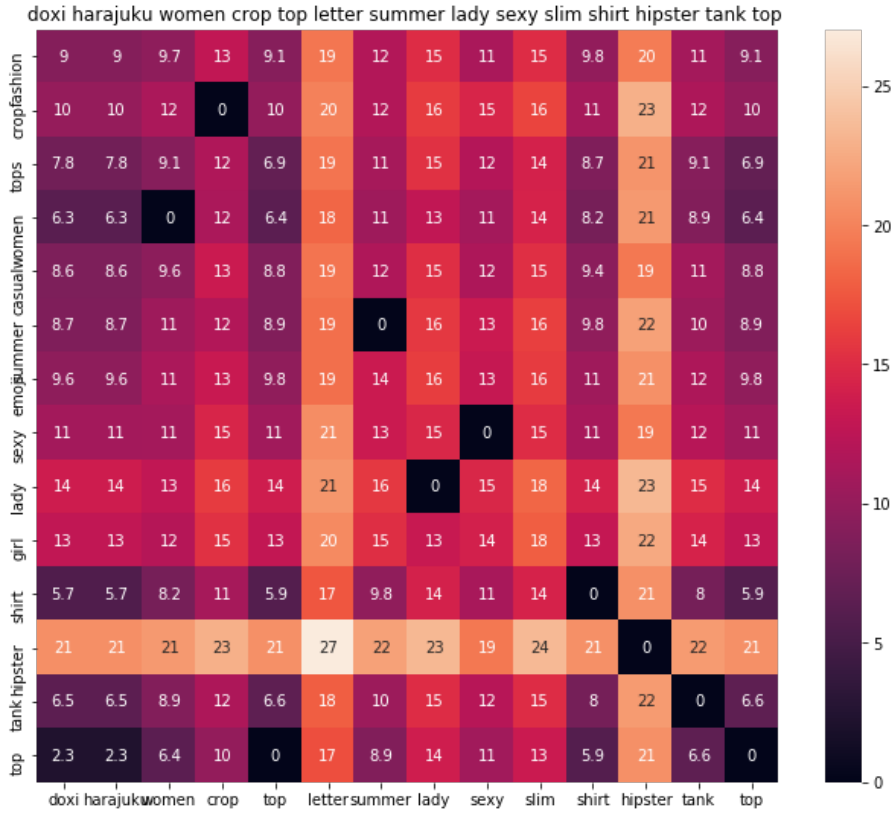
ASIN : B010V35OBU
Brand : Doxi Supermall
euclidean distance from input : 2.0631943



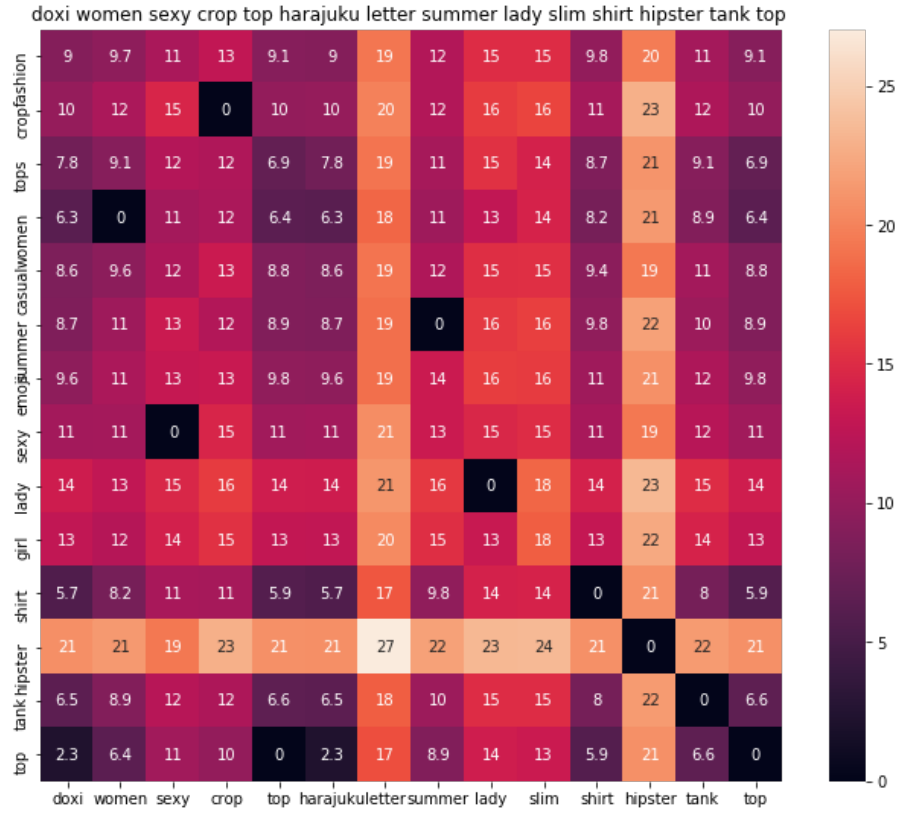
ASIN : B010V3A23U
Brand : Doxi Supermall
euclidean distance from input : 2.124369



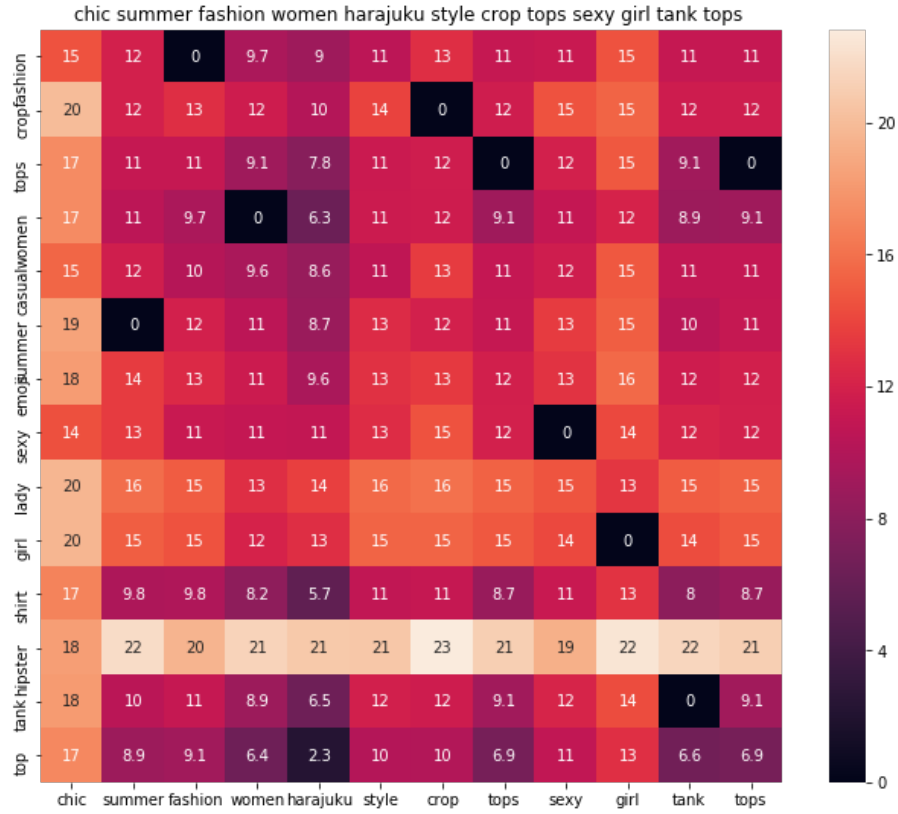
ASIN : B011RCJH58
Brand : Chiclook Cool
euclidean distance from input : 2.1507082



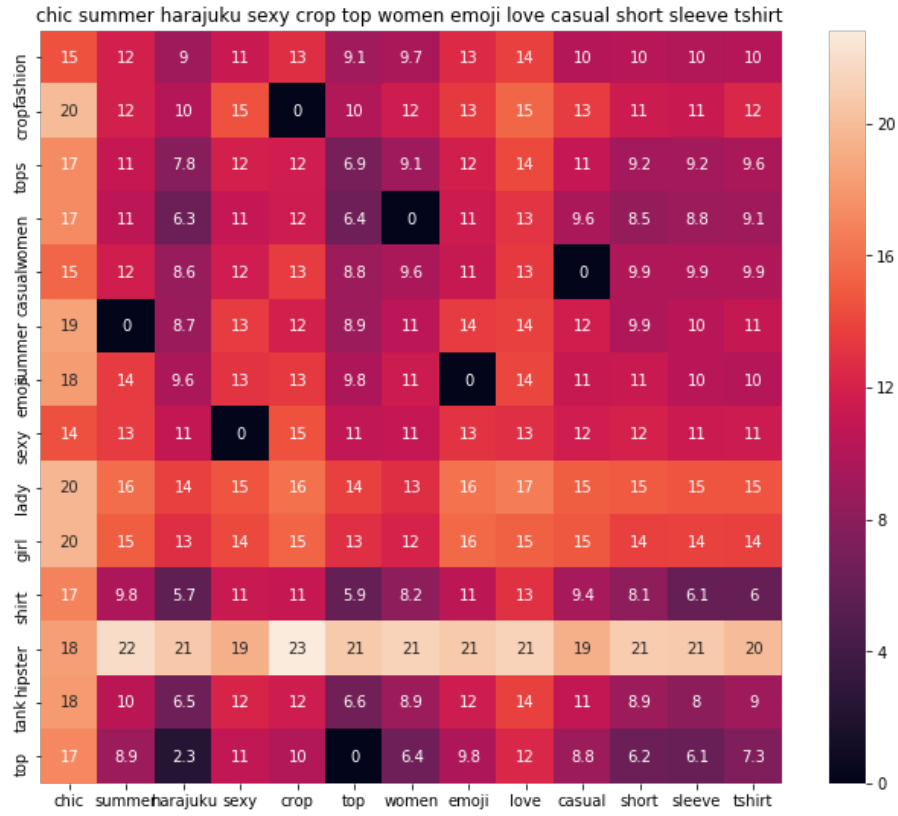
ASIN : B010V380LQ
Brand : Doxi Supermall
euclidean distance from input : 2.1674926



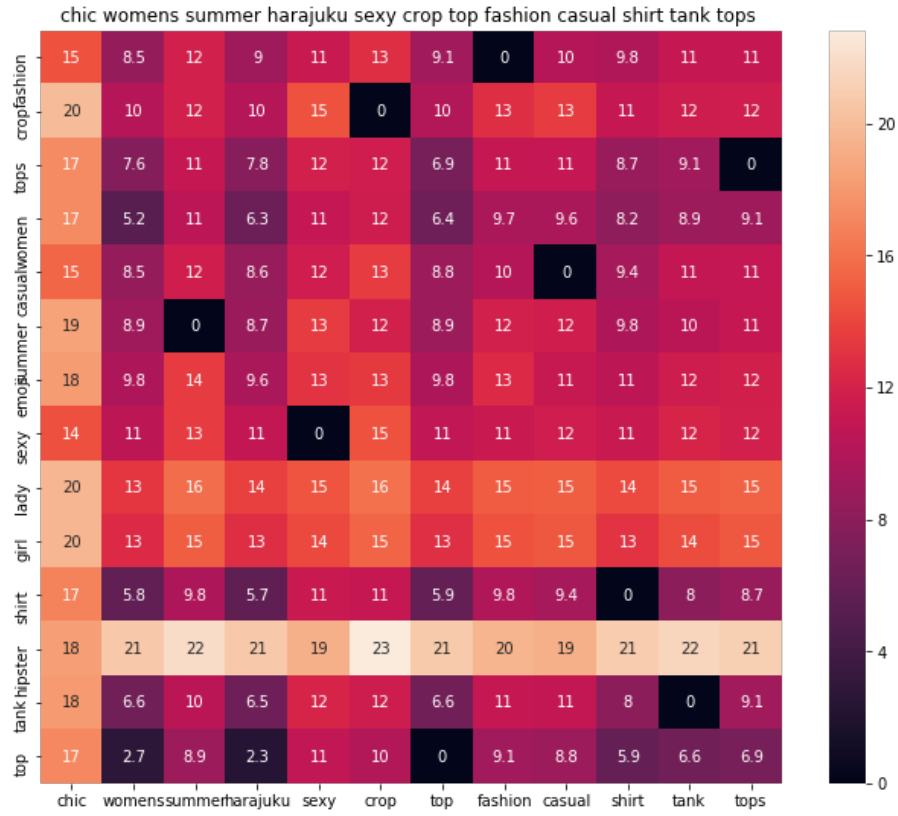
ASIN : B010V39146
Brand : Doxi Supermall
euclidean distance from input : 2.1674926



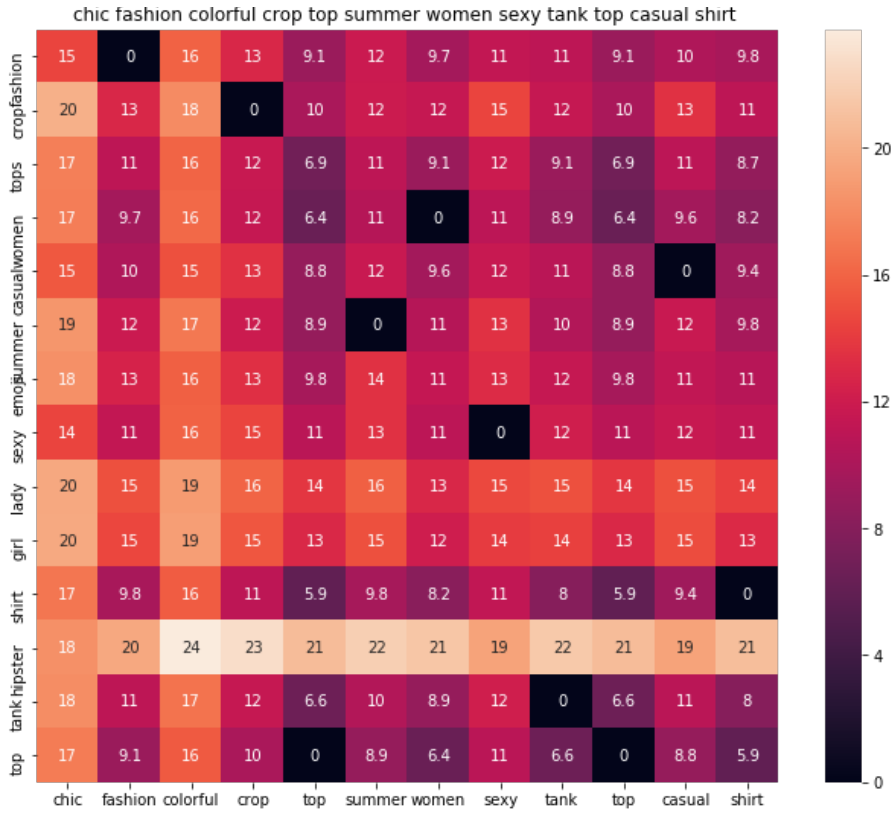
ASIN : B011OU51US
Brand : Chiclook Cool
euclidean distance from input : 2.2320182



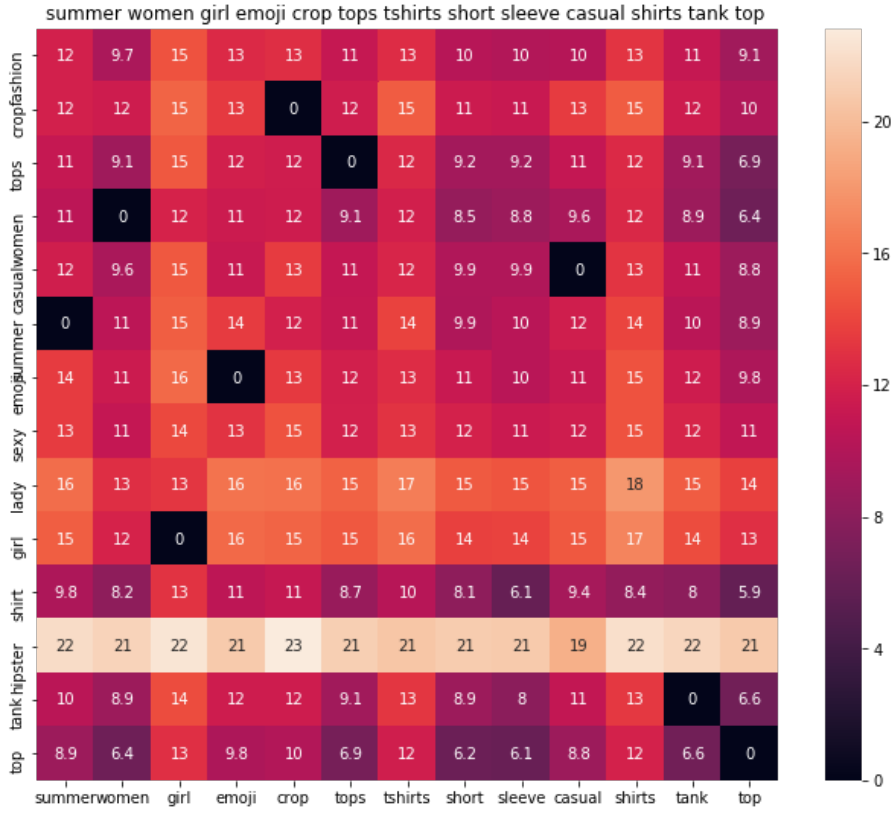
ASIN : B011UEVF4O
Brand : Chiclook Cool
euclidean distance from input : 2.313454
==========



ASIN : B011RCJEMO
Brand : Chiclook Cool
euclidean distance from input : 2.3378797
==========



ASIN : B011RCJ6UE
Brand : Chiclook Cool
euclidean distance from input : 2.5151842
==========



ASIN : B0124ECIU4
Brand : Doxi Supermall
euclidean distance from input : 2.552085
==========

Weighted Similarity using Brand and Color

In [74]:

```
# some of the brand values are empty.
# Replace Null with string "NULL"

data['brand'].fillna(value="Not given", inplace=True )

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

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

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

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

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

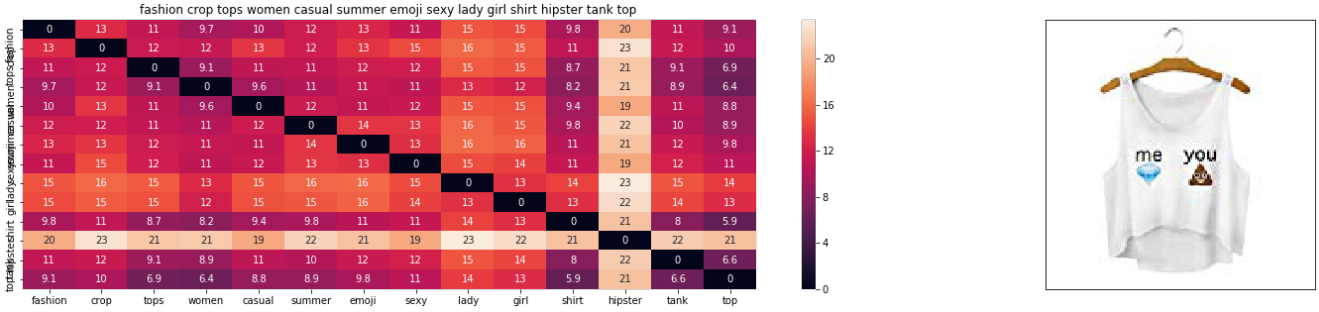
In [76]:

```
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: can have two values, 1. avg 2. weighted

    s1_vec = get_word_vec(sentance1, doc_id1, model)
    s2_vec = get_word_vec(sentance2, doc_id2, model)
    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]]] #
recommonded apparel's features

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

    # we create a table with the data_matrix
    table = ff.create_table(data_matrix, index=True, colorscale=colorscale)
    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, plot heatmap
    ax1 = plt.subplot(gs[:, :-5])
    # ploting the heap map based on the pairwise distances
    ax1 = sns.heatmap(np.round(s1_s2_dist,6), annot=True)
    # set the x axis labels as recommended apparels title
    ax1.set_xticklabels(sentance2.split())
    # set the y axis labels as input apparels title
    ax1.set_yticklabels(sentance1.split())
    # set title as recommended apparels title
    ax1.set_title(sentance2)

    # in last 25 * 10:15 grids, display image
    ax2 = plt.subplot(gs[:, 10:16])
    ax2.grid(False)
    ax2.set_xticks([])
    ax2.set_yticks([])
```

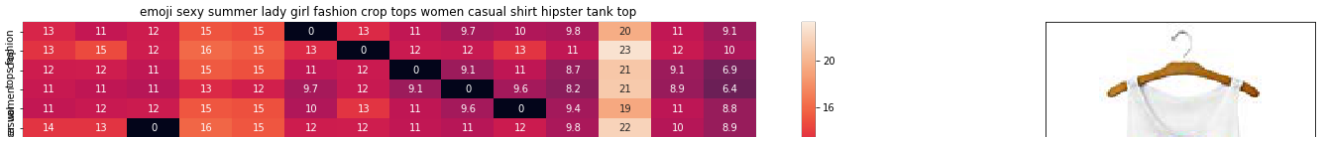
```
    # pass url to display image
    display_img(url, ax2, fig)

    plt.show()
```

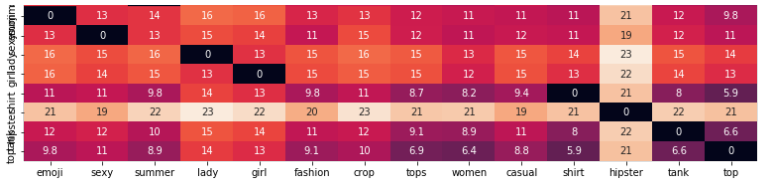
In [77]:

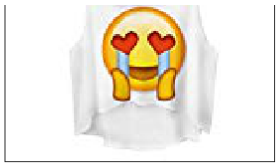
```
def idf_w2v_brand(doc_id, w1, w2, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand and color features

    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)

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


    for i in range(0, len(indices)):
        heat_map_w2v_brand(data['title'].loc[df_indices[0]],data['title'].loc[df_indices[i]], data[
'medium_image_url'].loc[df_indices[i]], indices[0], indices[i],df_indices[0], df_indices[i], 'weigh
ted')
        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)
```



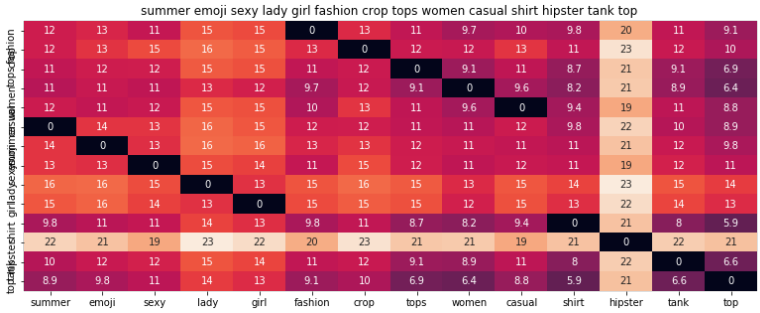

```
ASIN : B010V3B44G
Brand : Doxi Supermall
euclidean distance from input : 0.0009765625
================================================================================================================
======================
```



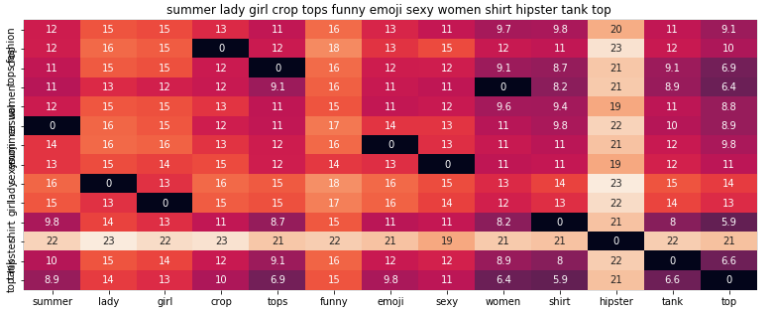




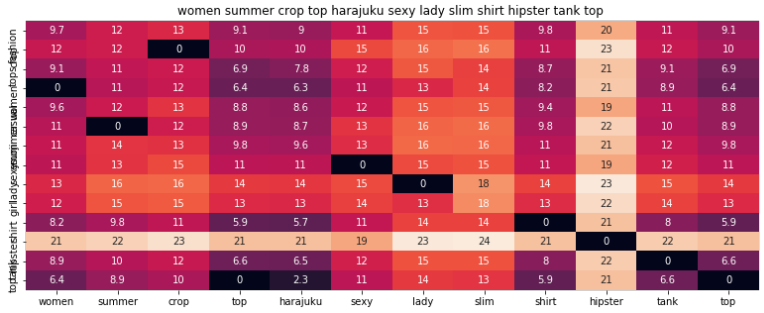
ASIN : B010V3BLWQ
Brand : Doxi Supermall
euclidean distance from input : 0.0009765625
==========
==========



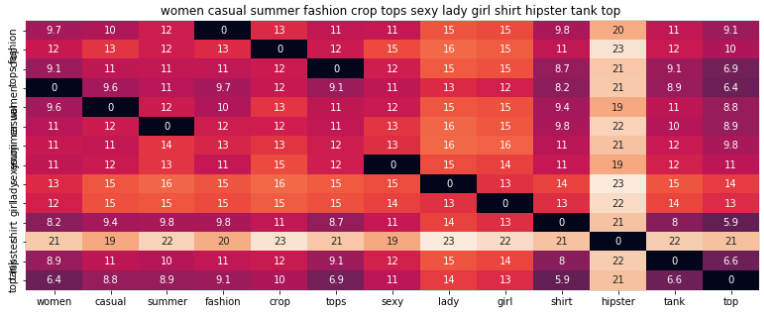
ASIN : B010V3BDII
Brand : Doxi Supermall
euclidean distance from input : 0.0009765625
==========
==========



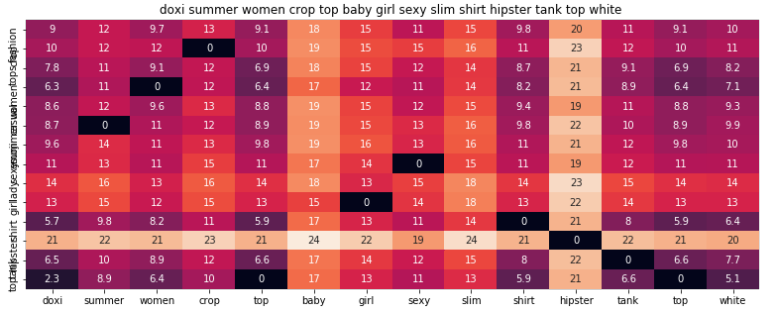
ASIN : B010V3BVMQ
Brand : Doxi Supermall
euclidean distance from input : 0.6765242099761963
==========
==========



ASIN : B010V3EDEE
Brand : Doxi Supermall
euclidean distance from input : 0.8717074394226074
==========



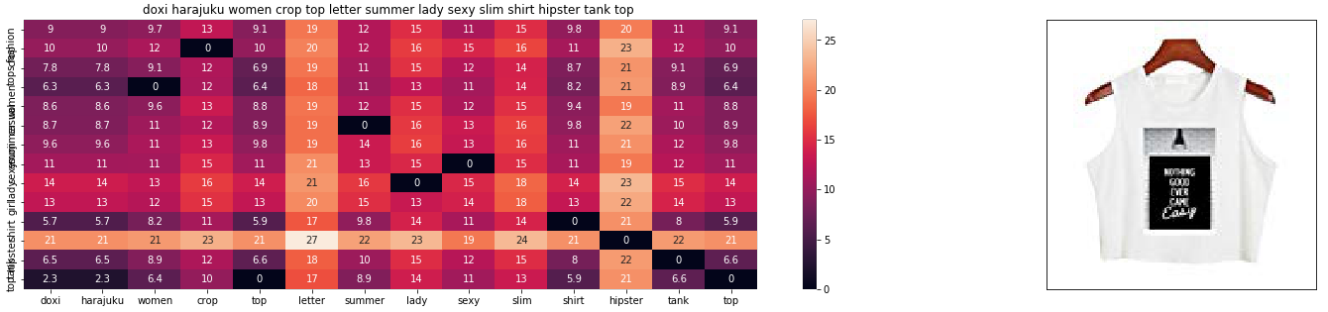
ASIN : B010V3AYSS
Brand : Doxi Supermall
euclidean distance from input : 1.0552222015280392
==========



ASIN : B010V3A23U
Brand : Doxi Supermall
euclidean distance from input : 1.0621844291687013

==========

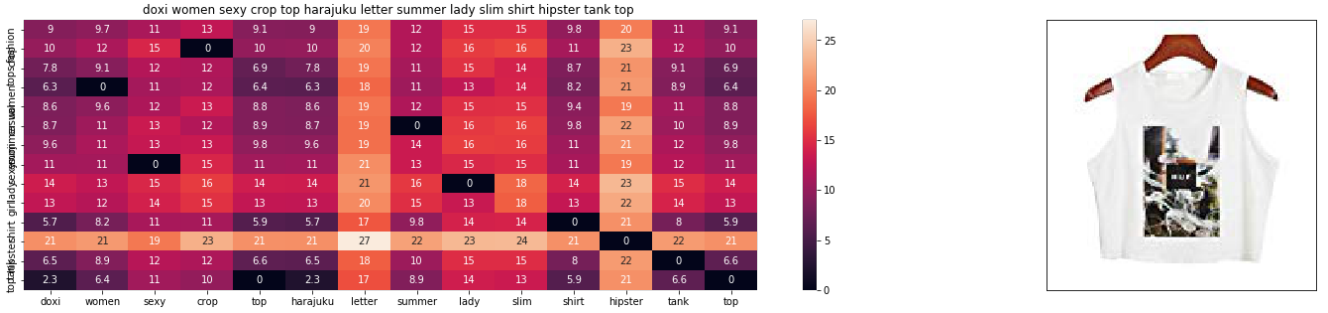
==========



ASIN : B010V380LQ
Brand : Doxi Supermall
euclidean distance from input : 1.083746337890625

==========

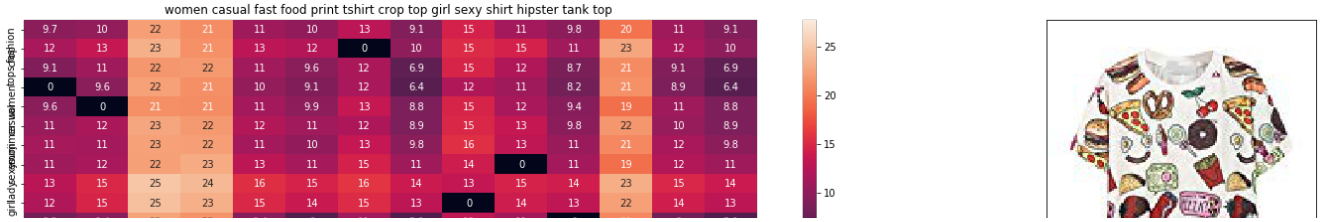
==========

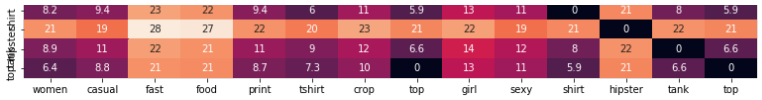


ASIN : B010V39146
Brand : Doxi Supermall
euclidean distance from input : 1.083746337890625

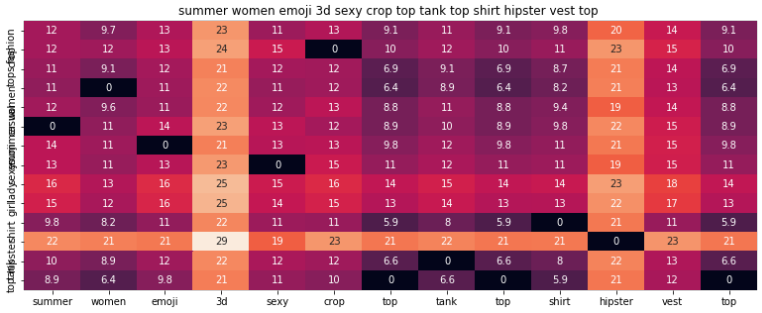
==========

==========

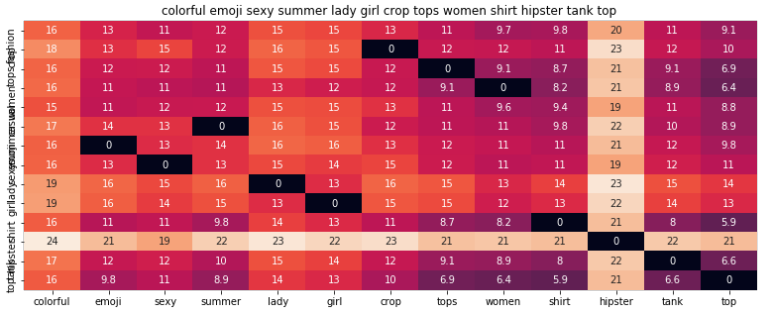




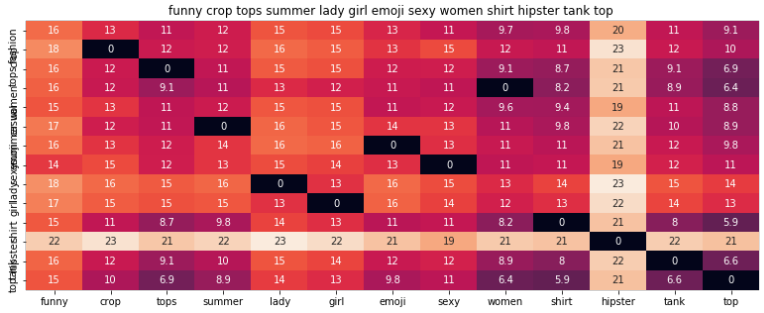
ASIN : B010V3AB5O
Brand : Doxi Supermall
euclidean distance from input : 1.3135900497436523
==========



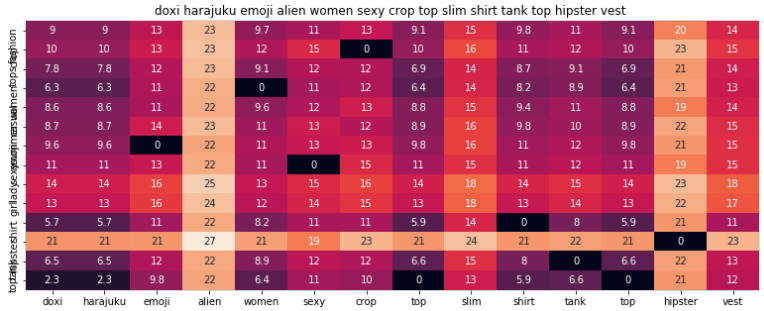
ASIN : B010V3E5EC
Brand : Doxi Supermall
euclidean distance from input : 1.3397350311279297
==========



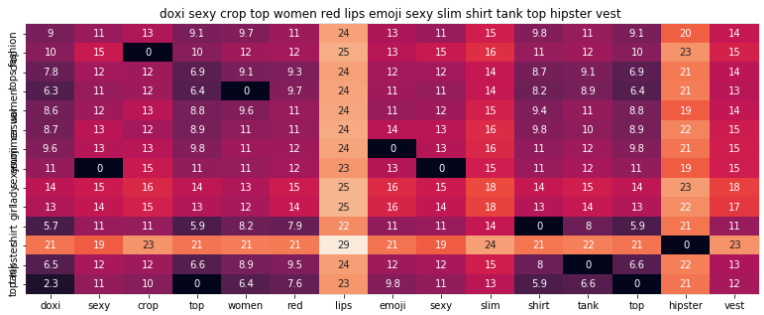
ASIN : B010V3BQZS
Brand : Doxi Supermall
euclidean distance from input : 1.3507425786871579
==========



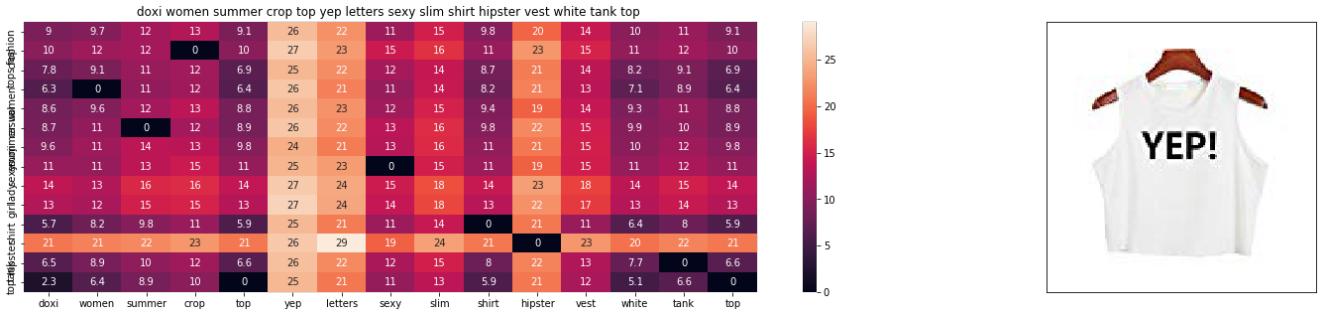
ASIN : B010V3C116
Brand : Doxi Supermall
euclidean distance from input : 1.3836309911627438
==========



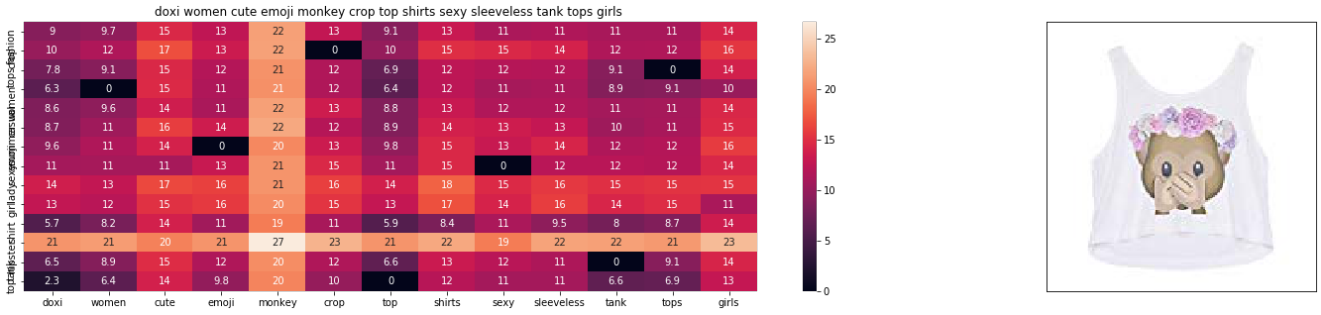
ASIN : B010TKXAI4
Brand : Doxi Supermall
euclidean distance from input : 1.3876009941101075
==========



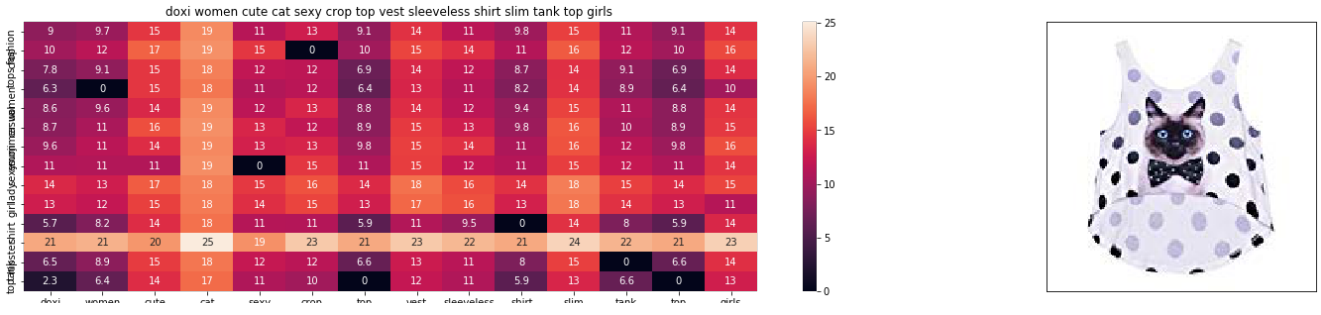
ASIN : B010TKXEHG
Brand : Doxi Supermall
euclidean distance from input : 1.4269559860229493
==========



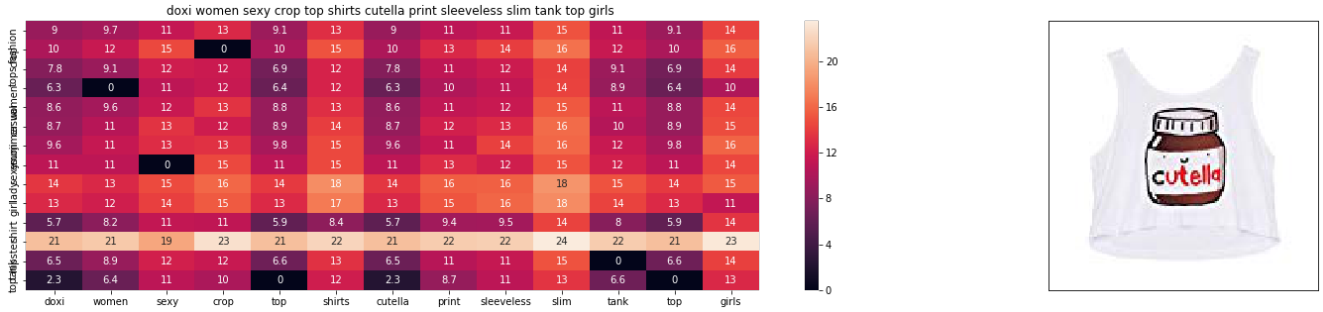
ASIN : B010V3487Q
Brand : Doxi Supermall
euclidean distance from input : 1.4681867599487304
==========



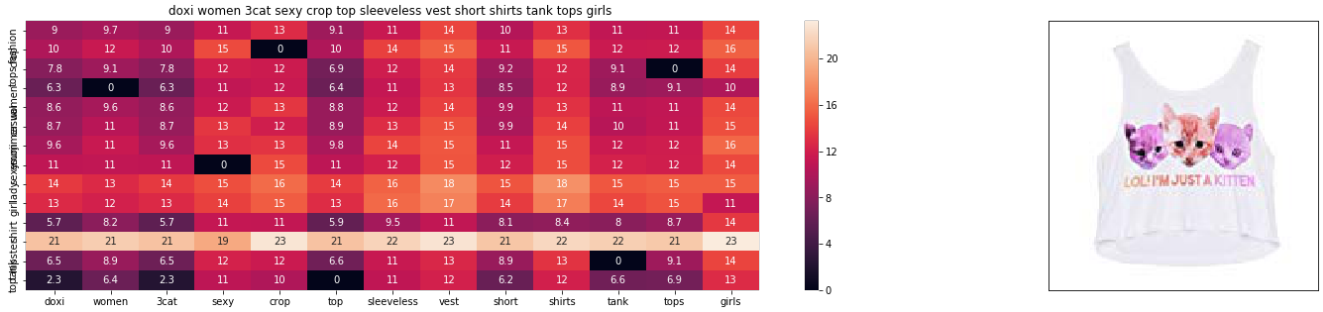
ASIN : B01LF90QTO
Brand : Doxi Supermall
euclidean distance from input : 1.4761534690856934
==========



ASIN : B01LF90ROI
Brand : Doxi Supermall
euclidean distance from input : 1.4821748733520508
==========



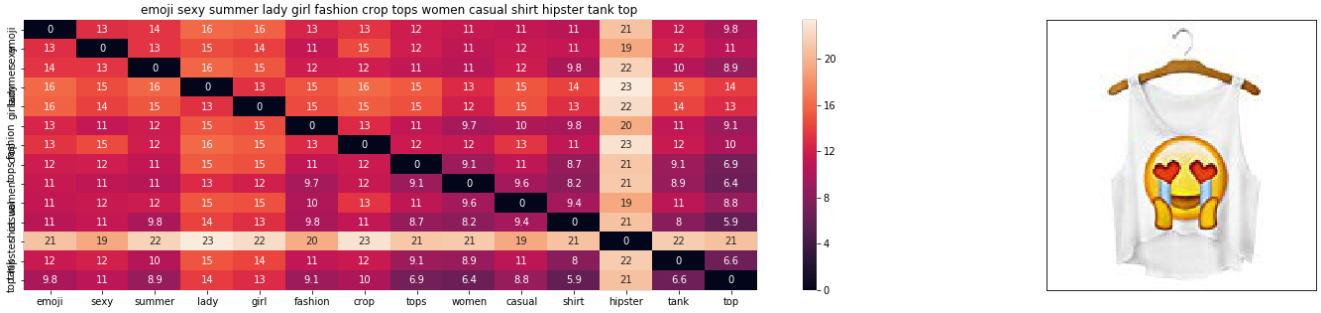
ASIN : B01LF90RKC
Brand : Doxi Supermall
euclidean distance from input : 1.5058938980102539
==========



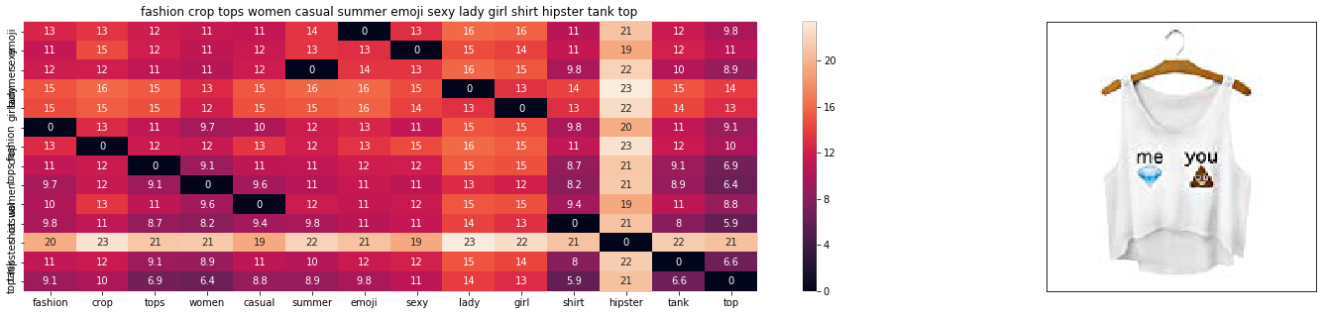
ASIN : B01LY4GQY0
Brand : Doxi Supermall
euclidean distance from input : 1.5098180770874023
==========

In [78]:

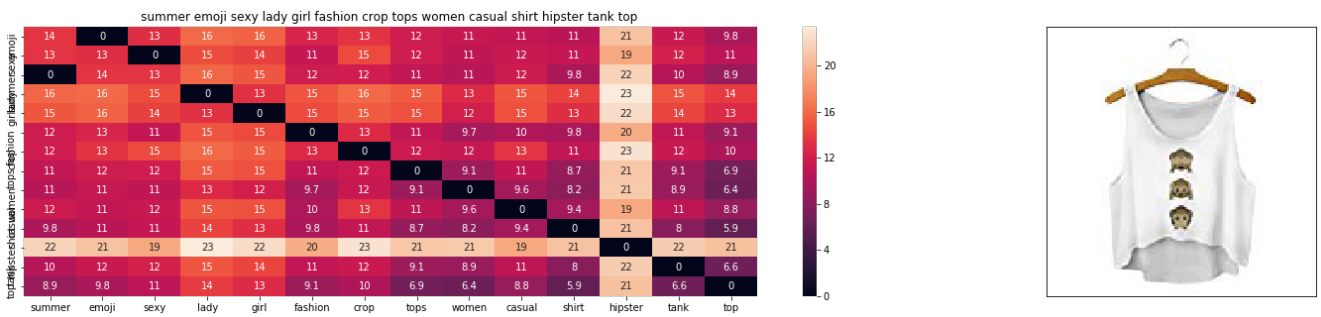
```
# brand and color weight =50
# title vector weight = 5

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



ASIN : B010V3BLWQ
Brand : Doxi Supermall
euclidean distance from input : 0.0001775568181818182
==========
==========

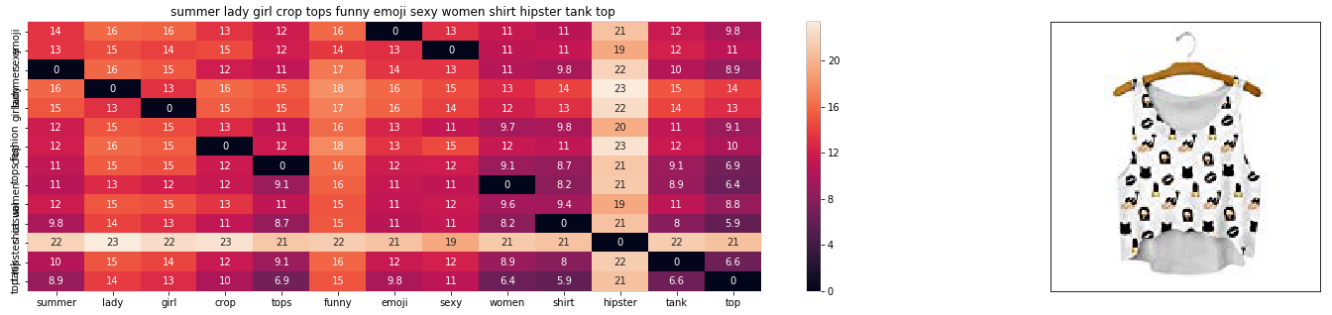


ASIN : B010V3B44G
Brand : Doxi Supermall
euclidean distance from input : 0.0001775568181818182
==========
==========

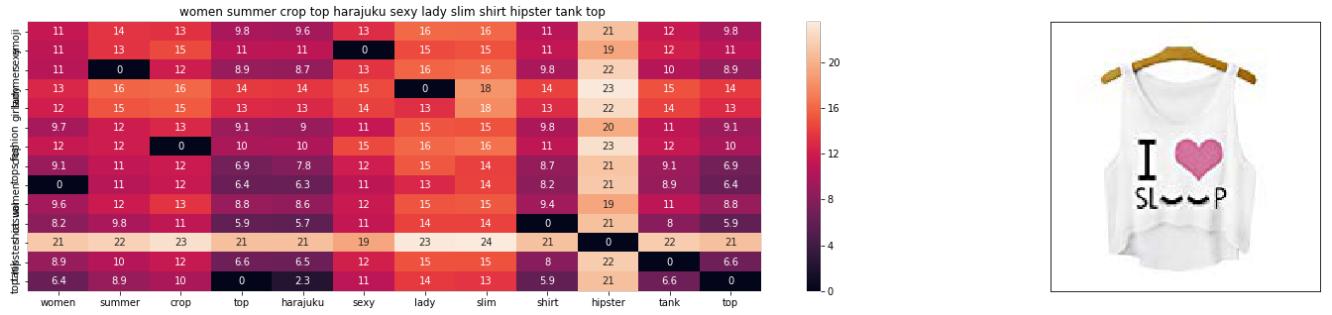


ASIN : B010V3BDII
Brand : Doxi Supermall

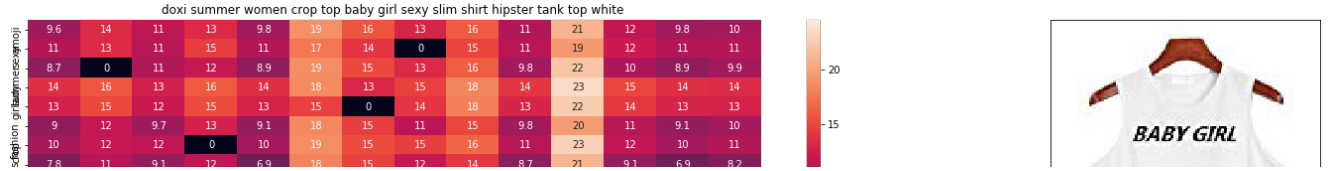
Brand : Doxi Supermall
euclidean distance from input : 0.0001775568181818182
==========
==========

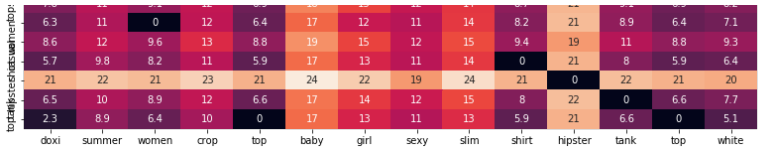


ASIN : B010V3BVMQ
Brand : Doxi Supermall
euclidean distance from input : 0.12300440181385387
==========
==========

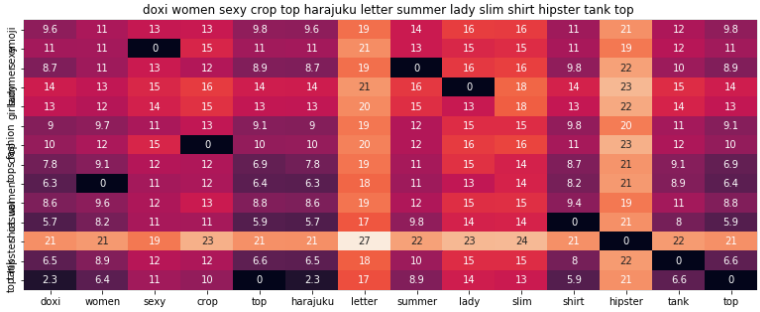


ASIN : B010V3EDEE
Brand : Doxi Supermall
euclidean distance from input : 0.15849226171320135
==========
==========

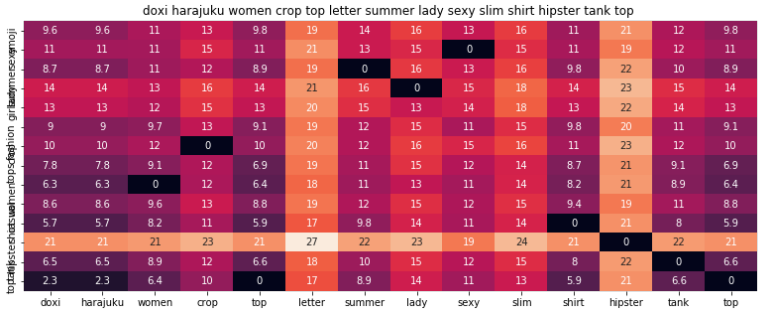




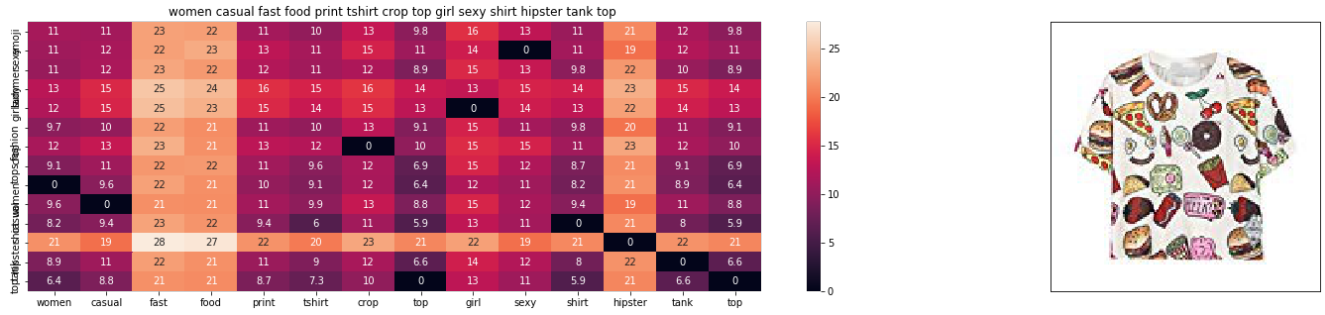
ASIN : B010V3A23U
Brand : Doxi Supermall
euclidean distance from input : 0.19312444166703657
==========
==========



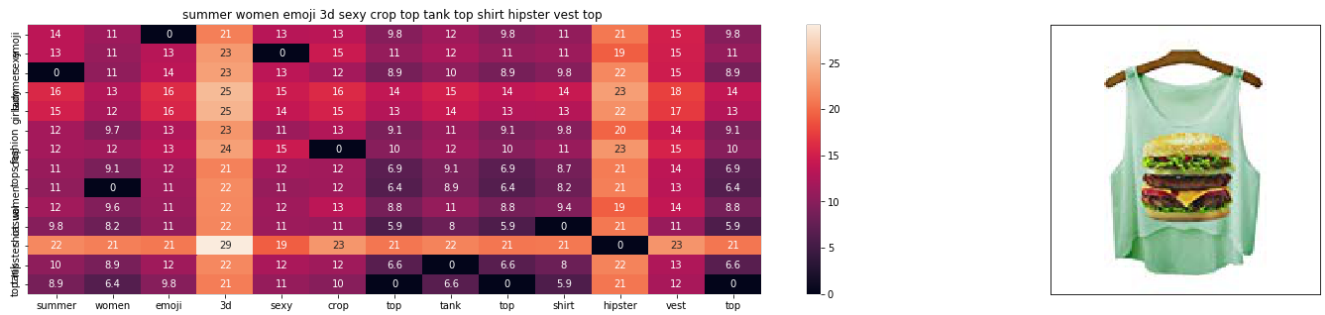
ASIN : B010V39146
Brand : Doxi Supermall
euclidean distance from input : 0.19704478870738637
==========
==========



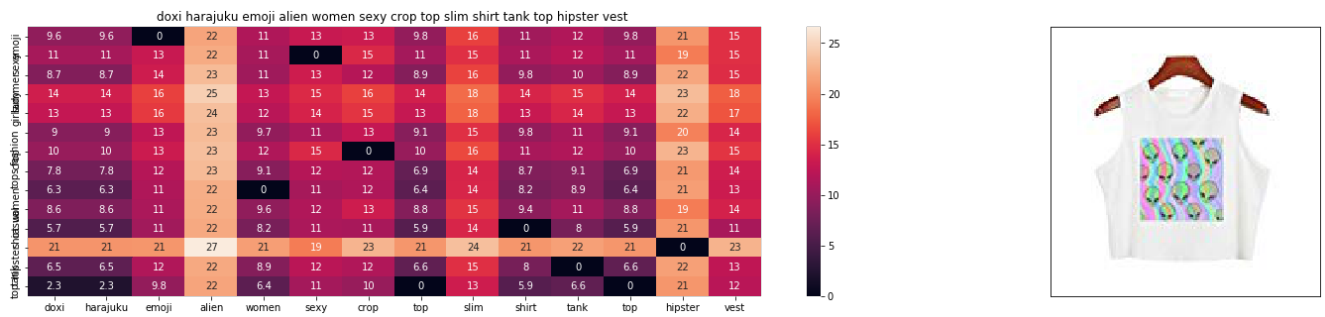
ASIN : B010V380LQ
Brand : Doxi Supermall
euclidean distance from input : 0.19704478870738637
==========
==========



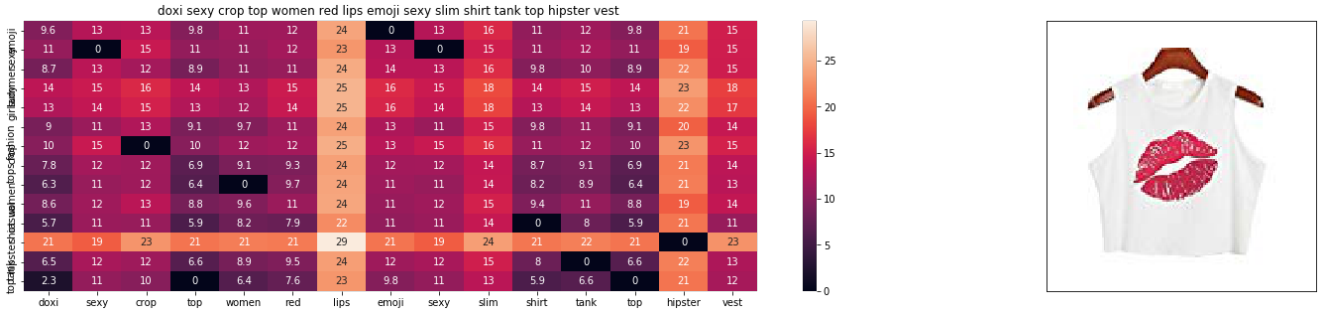
ASIN : B010V3AB5O
Brand : Doxi Supermall
euclidean distance from input : 0.23883455449884589
==========



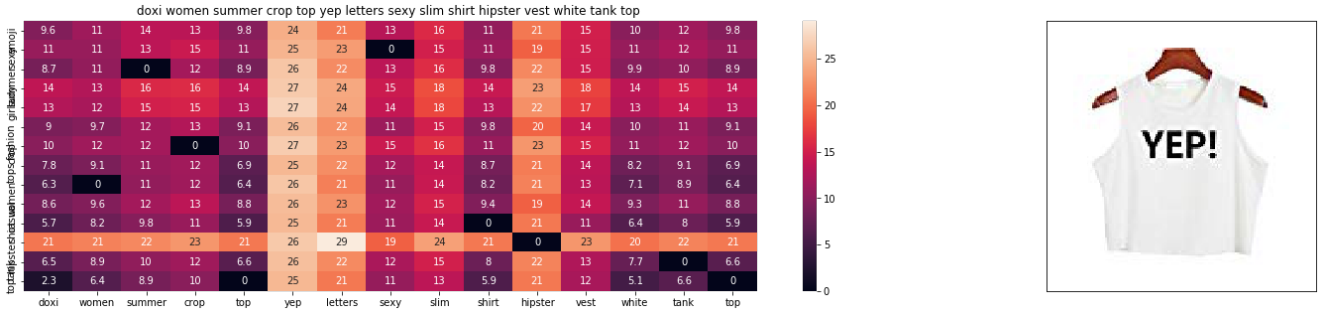
ASIN : B010V3E5EC
Brand : Doxi Supermall
euclidean distance from input : 0.2435881874778054
==========



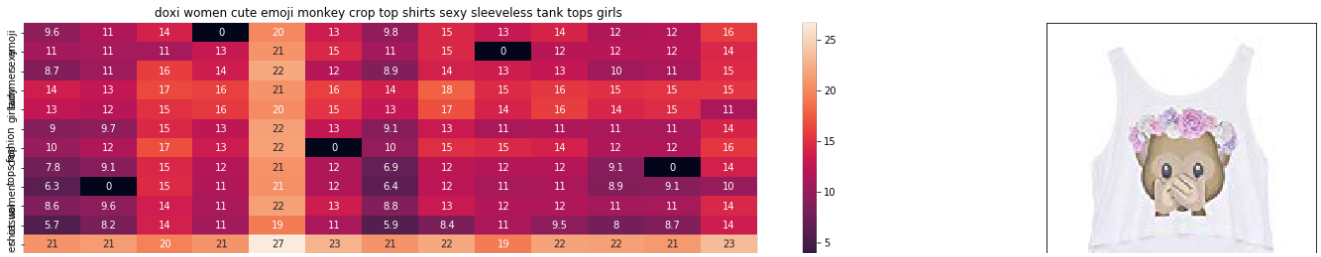
ASIN : B010TKXAI4
Brand : Doxi Supermall
euclidean distance from input : 0.25229108983820137
==========

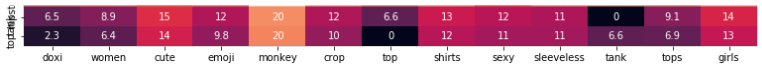


ASIN : B010TKXEHG
Brand : Doxi Supermall
euclidean distance from input : 0.25944654291326347
==========

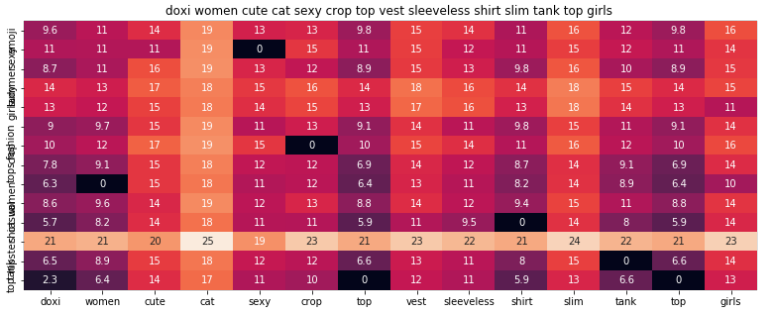


ASIN : B010V3487Q
Brand : Doxi Supermall
euclidean distance from input : 0.26694304726340556
==========

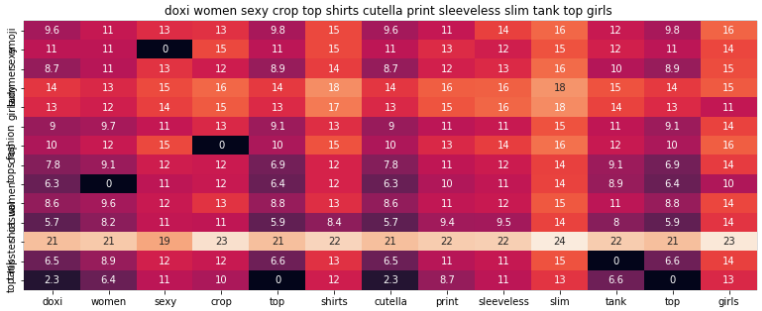




ASIN : B01LF90QTO
Brand : Doxi Supermall
euclidean distance from input : 0.2683915398337624



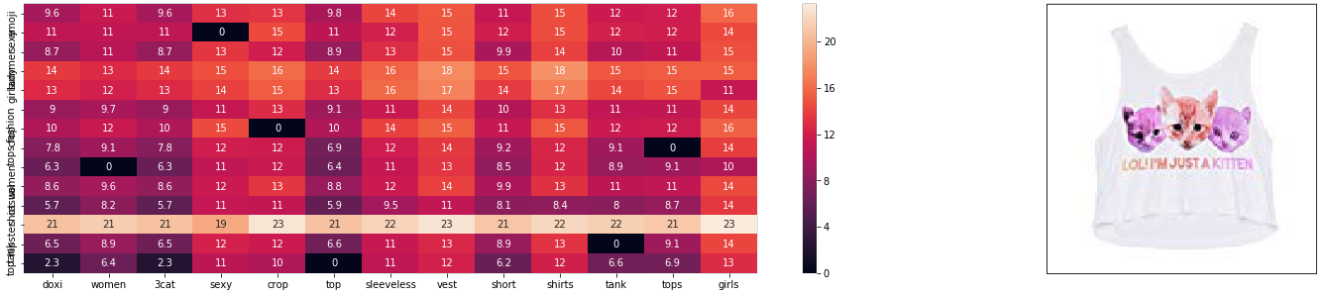
ASIN : B01LF90ROI
Brand : Doxi Supermall
euclidean distance from input : 0.2694863406094638



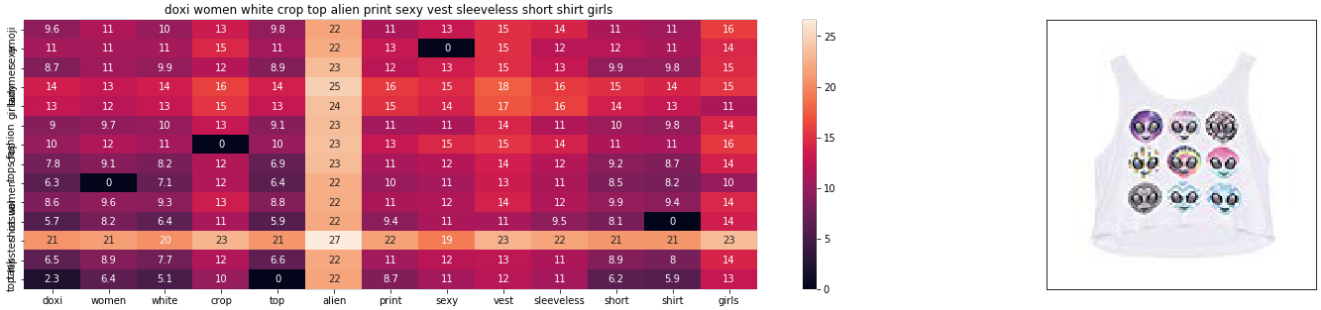
ASIN : B01LF90RKC
Brand : Doxi Supermall
euclidean distance from input : 0.2737988905473189

---

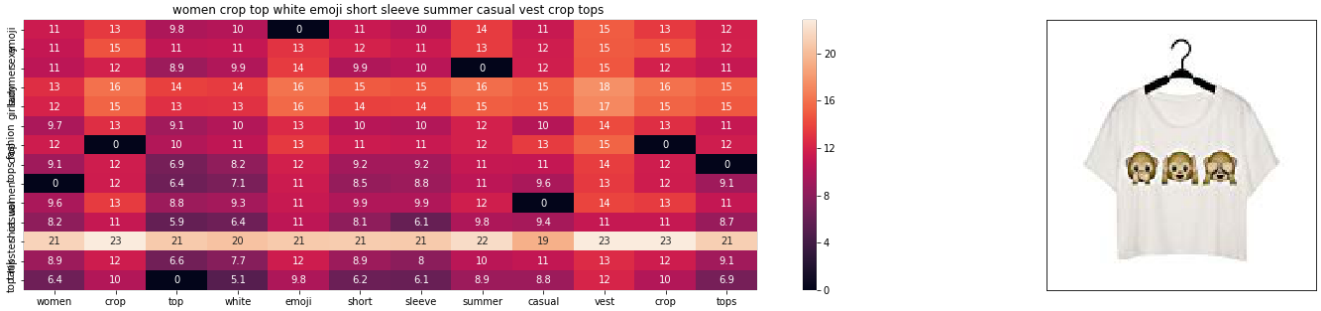
doxi women 3cat sexy crop top sleeveless vest short shirts tank tops girls



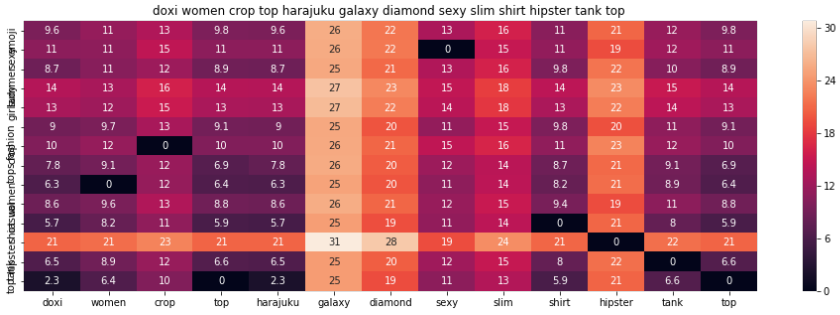
ASIN : B01LY4GQY0
Brand : Doxi Supermall
euclidean distance from input : 0.27451237765225495
==========



ASIN : B01LF90SCY
Brand : Doxi Supermall
euclidean distance from input : 0.2815914500843395
==========



ASIN : B0124E7MHS
Brand : Doxi Supermall
euclidean distance from input : 0.2871342745694247
==========



ASIN : B010V39EUM
Brand : Doxi Supermall
euclidean distance from input : 0.30813400962136006
==========
==========