Amazon Apparel Recommendations

[4.2] Data and Code:

https://drive.google.com/open?id=0BwNkduBnePt2VWhCYXhMV3p4dTg (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 [0]:

```
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 [0]:

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

Out[35]:

```
Index(['asin', 'author', 'availability', 'availability_type', 'brand', 'colo
r',
       'editorial_reivew', 'editorial_review', 'formatted_price',
       'large_image_url', 'manufacturer', 'medium_image_url', 'model',
       'product_type_name', 'publisher', 'reviews', 'sku', 'small_image_ur
l',
       '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 e
x: red and black stripes)
4. product_type_name (type of the apperal, ex: SHIRT/TSHIRT)
5. medium_image_url (url of the image)
6. title (title of the product.)
7. formatted_price (price of the product)

In [3]:

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

In [0]:

```
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[37]:

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

[5.1] Missing data for various features.

Basic stats for the feature: product_type_name

In [0]:

```
# 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 [0]:

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

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

In [0]:

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

Out[40]:

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

Basic stats for the feature: brand

In [0]:

```
# 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 [0]:

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

Out[42]:

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

Basic stats for the feature: color

In [0]:

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


# 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 [0]:

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

Out[44]:

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

Basic stats for the feature: formatted_price

In [0]:

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

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

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

In [0]:

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

Out[46]:

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

Basic stats for the feature: title

In [0]:

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

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

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

In [0]:

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

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

In [0]:

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

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

In [0]:

```
# consider products which have color information
# data['color'].isnull() => gives the information about the dataframe row's which have null
data =data.loc[~data['color'].isnull()]
print('Number of data points After eliminating color=NULL :', data.shape[0])
```

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

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

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

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

In [0]:

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

In [0]:

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


'''
from PIL import Image
import requests
from io import BytesIO

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


'''
```

Out[52]:

```
"\nfrom PIL import Image\nimport requests\nfrom io import BytesIO\n\nfor ind
ex, row in images.iterrows():\n        url = row['large_image_url']\n
response = requests.get(url)\n        img = Image.open(BytesIO(response.cont
ent))\n        img.save('workshop/images/28k_images/'+row['asin']+'.jpeg')\n
\n\n"
```

[5.2] Remove near duplicate items

[5.2.1] Understand about duplicates.

In [0]:

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

# find number of products that have duplicate titles.
print(sum(data.duplicated('title')))
# we have 2325 products which have same title but different color
```

2325

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

These shirts exactly same except in color

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

[5.2.2] Remove duplicates : Part 1

In [5]:

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

In [6]:

```
data.head()
```

Out[6]:

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

In [7]:

```
# 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 [8]:

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

Out[8]:

	asin	brand	color	medium_image_url	product_type_name	title
61973	B06Y1KZ2WB	Éclair	Black/Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	Éclair Women's Printed Thin Strap Blouse Black..
133820	B010RV33VE	xiaoming	Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Womens Sleeveless Loose Long T-shirts..
81461	B01DDSDLNS	xiaoming	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Women's White Long Sleeve Single Brea..
75995	B00X5LYO9Y	xiaoming	Red Anchors	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Stripes Tank Patch/Bear Sleeve Anchor..
151570	B00WPJG35K	xiaoming	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Sleeve Sheer Loose Tasse Kimono Woma..

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

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

Title 2:
25. tokidoki The Queen of Diamonds Women's Shirt X-Large
26. tokidoki The Queen of Diamonds Women's Shirt Small
27. tokidoki The Queen of Diamonds Women's Shirt Large

Title 3:
61. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal 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 [9]:

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

In [10]:

```
import itertools
stage1_dedupe_asins = []
i = 0
j = 0
num_data_points = data_sorted.shape[0]
while i < num_data_points and j < num_data_points:

    previous_i = i

    # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of'
    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',
        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
        # example: a =['a', 'b', 'c', 'd']
        # b = ['a', 'b', 'd']
        # itertools.zip_longest(a,b): will give [('a','a'), ('b','b'), ('c','d'), ('d', Non
        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
        # if the number of words in which both strings differ are < 2 , we are considering
        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 ina
            stage1_dedupe_asins.append(data_sorted['asin'].loc[indices[i]])

            # if the comaprision between is between num_data_points, num_data_points-1 stri
            if j == num_data_points-1: stage1_dedupe_asins.append(data_sorted['asin'].loc[i

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

In [0]:

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

We removed the dupliactes which differ only at the end.

In [0]:

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

```
Number of data points :  17593
```

In [0]:

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

[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 [11]:

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

In [12]:

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

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

stage2_dedupe_asins = []
while len(indices)!=0:
    i = indices.pop()
    stage2_dedupe_asins.append(data['asin'].loc[i])
    # consider the first apperal's title
    a = data['title'].loc[i].split()
    # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of'
    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',

        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
        # example: a =['a', 'b', 'c', 'd']
        # b = ['a', 'b', 'd']
        # itertools.zip_longest(a,b): will give [('a','a'), ('b','b'), ('c','d'), ('d', Non
        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
        if (length - count) < 3:
            indices.remove(j)
```

In [13]:

```
# from whole previous products we will consider only
# the products that are found in previous cell
data = data.loc[data['asin'].isin(stage2_dedupe_asins)]
```

In [14]:

```
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 [15]:

```
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 [16]:

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

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

In [17]:

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

In [18]:

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

```
3.7546838679991197 seconds
```

In [0]:

```
data.head()
```

Out[6]:

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

In [19]:

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

Stemming

In [20]:

```
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 [21]:

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

Out[21]:

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

In [22]:

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


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

#plotting code to understand the algorithm's decision.
def plot_heatmap(keys, values, labels, url, text):
        # keys: list of words of recommended title
        # values: len(values) ==  len(keys), values(i) represents the occurence of the word
        # labels: len(labels) == len(keys), the values of labels depends on the model we ar
                # if model == 'bag of words': labels(i) = values(i)
                # if model == 'tfidf weighted bag of words':labels(i) = tfidf(keys(i))
                # if model == 'idf weighted bag of words':labels(i) = idf(keys(i))
        # url : apparel's url

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

        # 1st, ploting heat map that represents the count of commonly ocurred words in titl
        ax = plt.subplot(gs[0])
        # it displays a cell in white color if the word is intersection(lis of words of tit
        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 a
    intersection = set(vec1.keys()) & set(vec2.keys())
```

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

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

    # labels: len(labels) == len(keys), the values of labels depends on the model we are us
        # 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 val
            if x in  tfidf_title_vectorizer.vocabulary_:
                labels.append(tfidf_title_features[doc_id, tfidf_title_vectorizer.vocabular
            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 o
            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 wrods along with the frequency of each
# word given "text"
def text_to_vector(text):
    word = re.compile(r'\w+')
    words = word.findall(text)
    # words stores list of all words in given string, you can try 'words = text.split()' th
    return Counter(words) # Counter counts the occurence of each word in list, it returns a



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 [23]:

```
from sklearn.feature_extraction.text import CountVectorizer
title_vectorizer = CountVectorizer()
title_features   = title_vectorizer.fit_transform(data['title'])
title_features.get_shape() # get number of rows and columns in feature matrix.
# title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(corpus) returns
# the a sparase matrix of dimensions #data_points * #words_in_corpus

# What is a sparse vector?

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

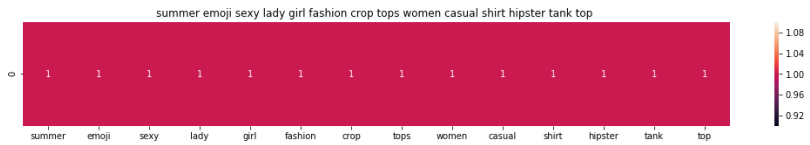
Out[23]:

```
(16435, 12684)
```

In [24]:

```
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 appar
    # the metric we used here is cosine, the coside distance is mesured as K(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_indice
        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 hips
ter tank top
Euclidean similarity with the query image : 0.0
============================================================
```

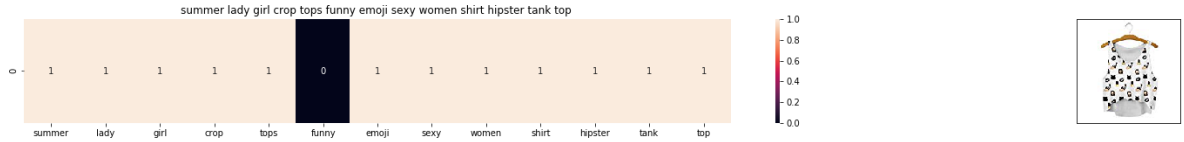


```
ASIN : B010V3BDII
Brand: Doxi Supermall
Title: summer emoji sexy lady girl fashion crop tops women casual shirt hips
ter tank top
Euclidean similarity with the query image : 0.0
============================================================
```

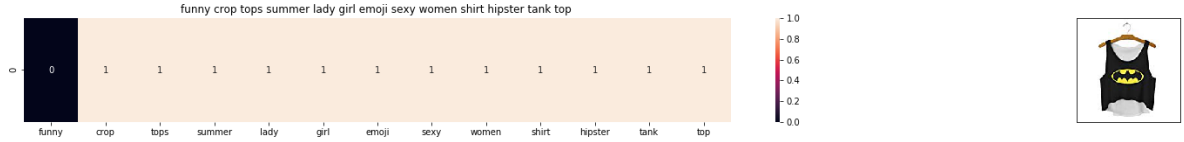
ASIN : B010V3BLWQ
Brand: Doxi Supermall
Title: emoji sexy summer lady girl fashion crop tops women casual shirt hips
ter tank top
Euclidean similarity with the query image : 0.0
==

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

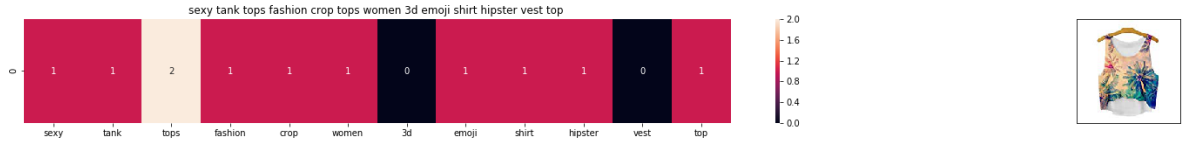
ASIN : B010V3BQZS
Brand: Doxi Supermall
Title: colorful emoji sexy summer lady girl crop tops women shirt hipster ta
nk top
Euclidean similarity with the query image : 1.7320508075688772
==



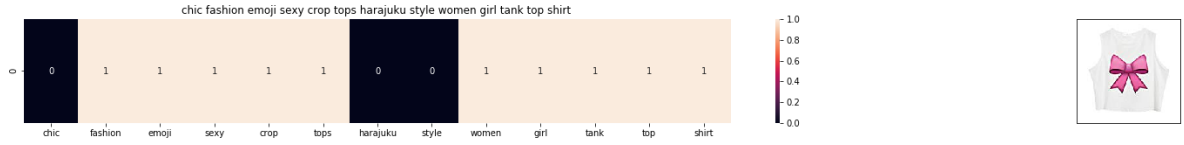
ASIN : B010V3BVMQ
Brand: Doxi Supermall
Title: summer lady girl crop tops funny emoji sexy women shirt hipster tank
top
Euclidean similarity with the query image : 1.7320508075688772
==



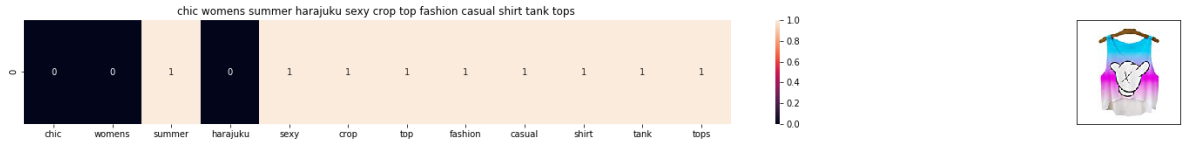
ASIN : B010V3C116
Brand: Doxi Supermall
Title: funny crop tops summer lady girl emoji sexy women shirt hipster tank
top
Euclidean similarity with the query image : 1.7320508075688772
==



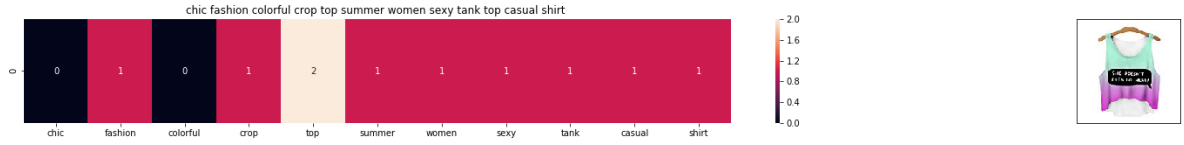
ASIN : B010V3DB9C
Brand: Doxi Supermall
Title: sexy tank tops fashion crop tops women 3d emoji shirt hipster vest to
p
Euclidean similarity with the query image : 2.6457513110645907
==



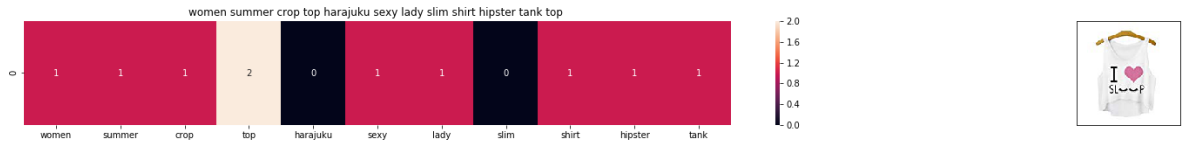
ASIN : B011RCJPR8
Brand: Chiclook Cool
Title: chic fashion emoji sexy crop tops harajuku style women girl tank top
shirt
Euclidean similarity with the query image : 2.6457513110645907
==



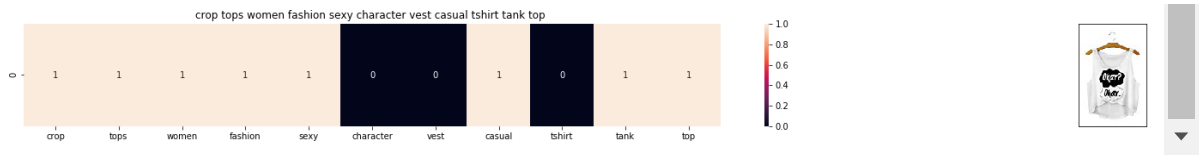
ASIN : B011RCJEMO
Brand: Chiclook Cool
Title: chic womens summer harajuku sexy crop top fashion casual shirt tank t
ops
Euclidean similarity with the query image : 2.8284271247461903
==



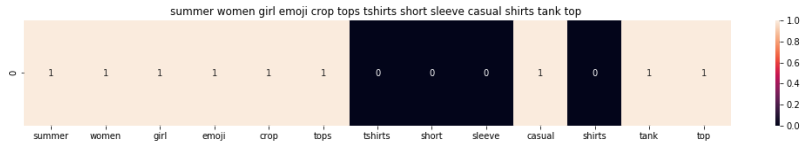
ASIN : B011RCJ6UE
Brand: Chiclook Cool
Title: chic fashion colorful crop top summer women sexy tank top casual shir
t
Euclidean similarity with the query image : 2.8284271247461903
==



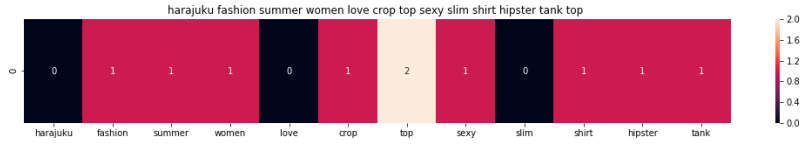
ASIN : B010V3EDEE
Brand: Doxi Supermall
Title: women summer crop top harajuku sexy lady slim shirt hipster tank top
Euclidean similarity with the query image : 2.8284271247461903
==



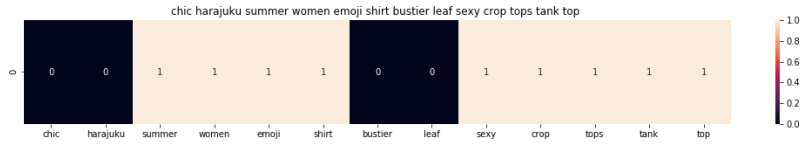
ASIN : B0107UEPVM
Brand: Mang GO
Title: crop tops women fashion sexy character vest casual tshirt tank top
Euclidean similarity with the query image : 3.0
==========



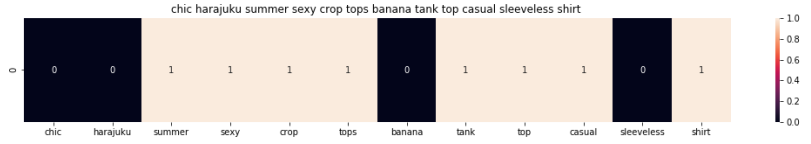
ASIN : B0124ECIU4
Brand: Doxi Supermall
Title: summer women girl emoji crop tops tshirts short sleeve casual shirts tank top
Euclidean similarity with the query image : 3.0
==========



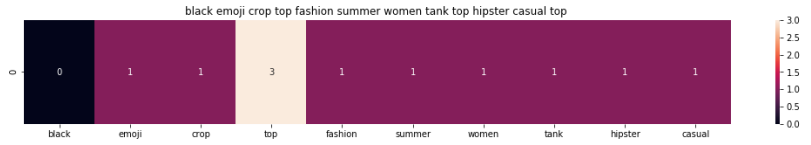
ASIN : B010V35OBU
Brand: Doxi Supermall
Title: harajuku fashion summer women love crop top sexy slim shirt hipster t ank top
Euclidean similarity with the query image : 3.0
==========



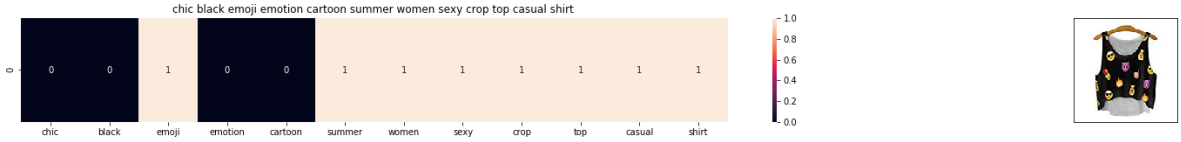
ASIN : B011UEXGH8
Brand: Chiclook Cool
Title: chic harajuku summer women emoji shirt bustier leaf sexy crop tops ta nk top
Euclidean similarity with the query image : 3.0
==========



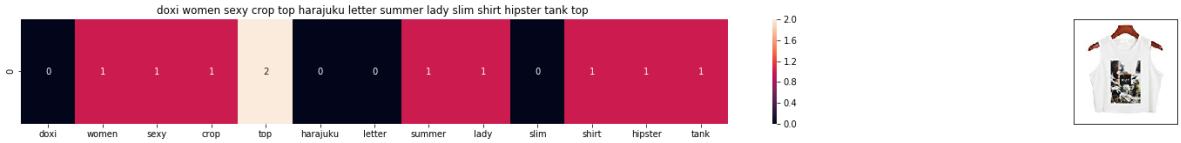
ASIN : B011RCIQBE
Brand: Chiclook Cool
Title: chic harajuku summer sexy crop tops banana tank top casual sleeveless shirt
Euclidean similarity with the query image : 3.1622776601683795
==========



```
ASIN : B0124E80M4
Brand: Doxi Supermall
Title: black emoji crop top fashion summer women tank top hipster casual t
op
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 sh
irt
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 hipst
er tank top
Euclidean similarity with the query image : 3.1622776601683795
============================================================
```

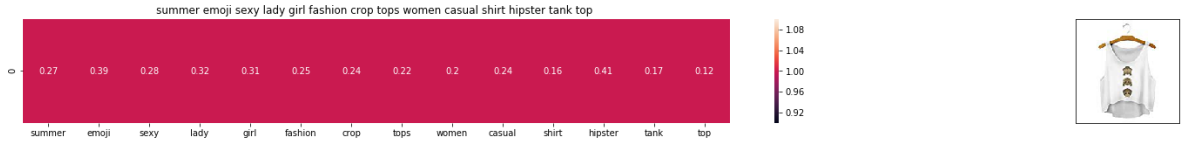
[8.5] TF-IDF based product similarity

In [25]:

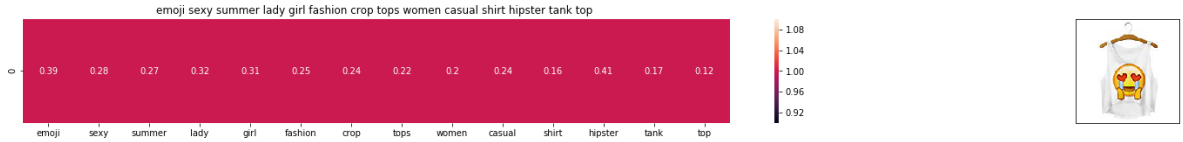
```
tfidf_title_vectorizer = TfidfVectorizer(min_df = 0)
tfidf_title_features = tfidf_title_vectorizer.fit_transform(data['title'])
# tfidf_title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(courpus) returns the a sparase matrix of dimensions #data
# tfidf_title_features[doc_id,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

In [26]:

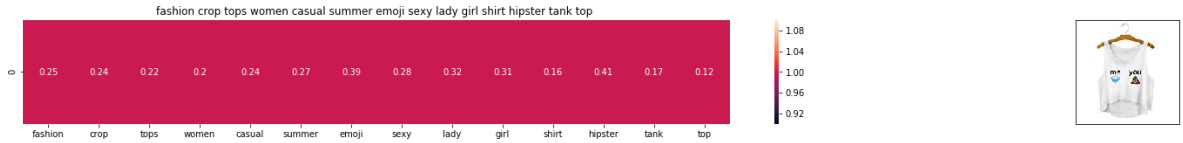
```
def tfidf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance from given input apparel to all remaining appar
    # the metric we used here is cosine, the coside distance is mesured as K(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_indic
        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 colo
```



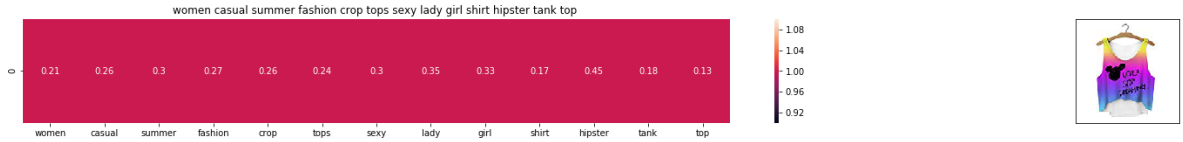

```
ASIN : B010V3BDII
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.0
=============================================================================
===================================================
```



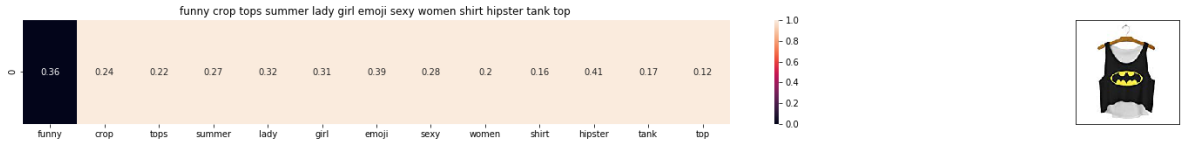

```
ASIN : B010V3BLWQ
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.0
=============================================================================
===================================================
```



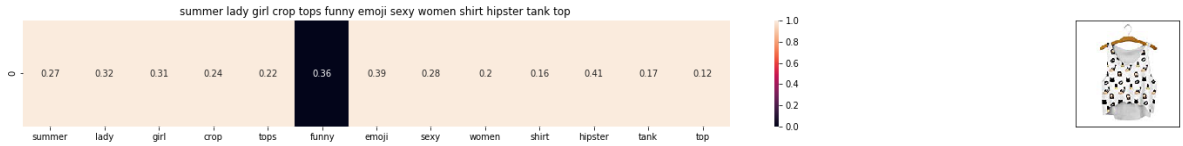

```
ASIN : B010V3B44G
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.0
=============================================================================
===================================================
```



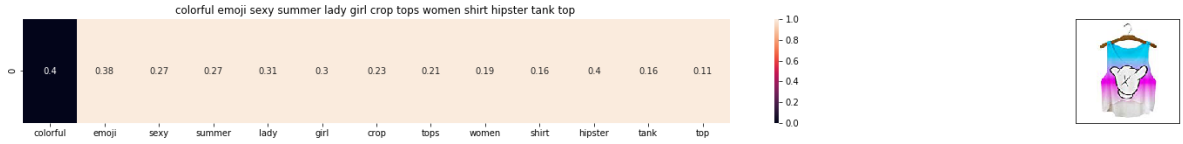
ASIN : B010V3AYSS
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.40138594750234946
==========
==========



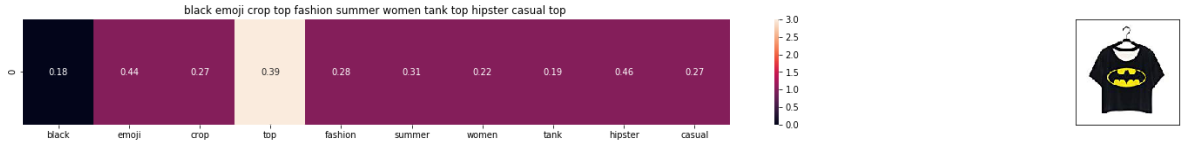
ASIN : B010V3C116
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.4954421553895553
==========
==========



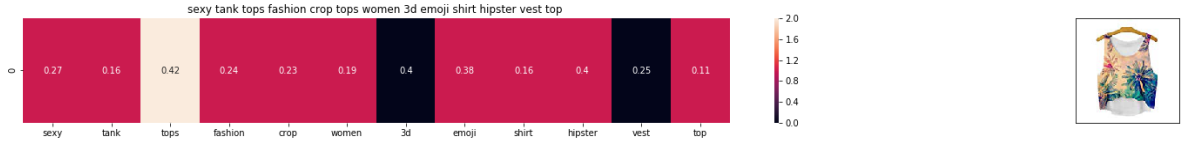
ASIN : B010V3BVMQ
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.4954421553895553
==========
==========



ASIN : B010V3BQZS
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.5241689140292635
==========
==========

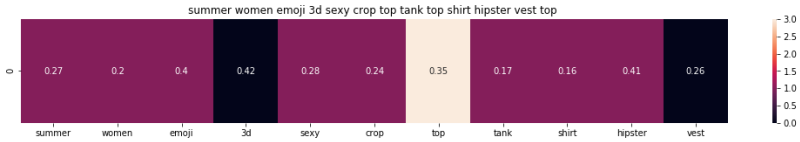


ASIN : B0124E80M4
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.6841581626642753
==========
==========

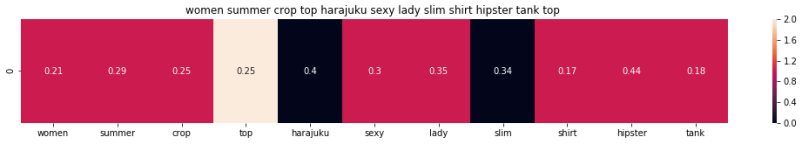


ASIN : B010V3DB9C
BRAND : Doxi Supermall

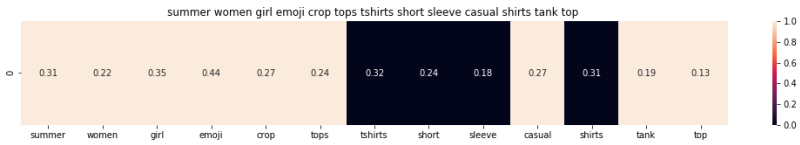
Eucliden distance from the given image : 0.7731343455110793



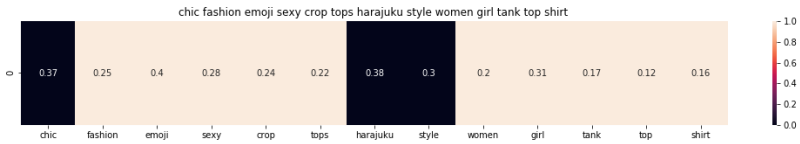
ASIN : B010V3E5EC
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.8128754313990525



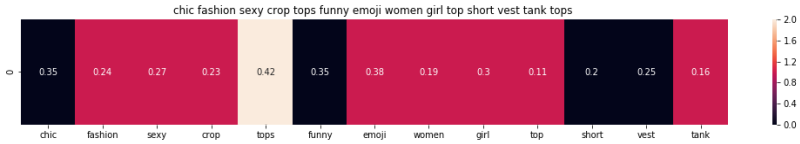
ASIN : B010V3EDEE
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.8437056912864757



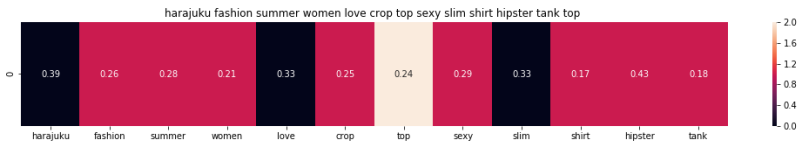
ASIN : B0124ECIU4
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.8553483954246196



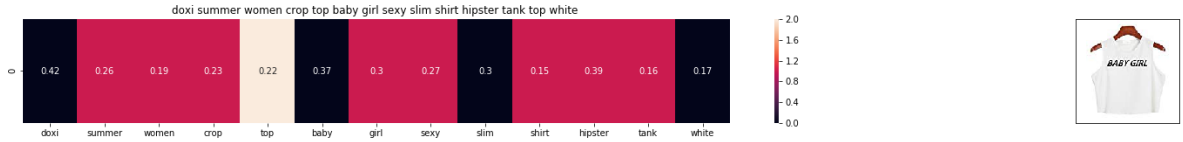
ASIN : B011RCJPR8
BRAND : Chiclook Cool
Eucliden distance from the given image : 0.8826321316686155



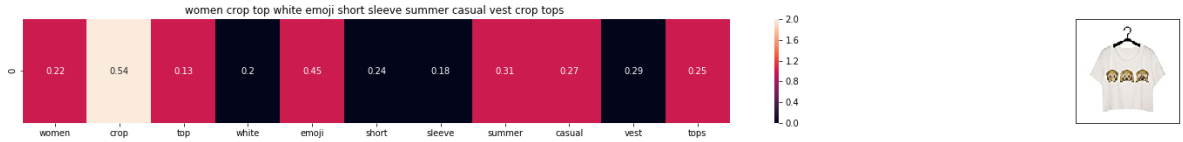
ASIN : B011RCJH58
BRAND : Chiclook Cool
Eucliden distance from the given image : 0.900401746801386



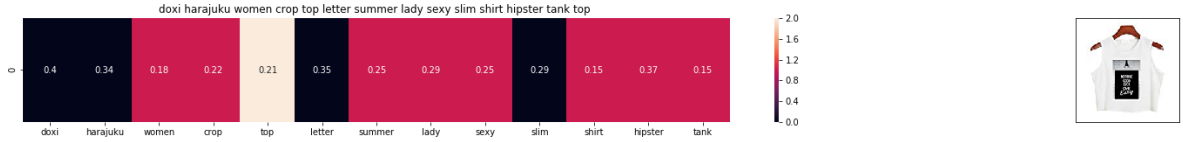
ASIN : B010V35OBU
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.9172816572673459
==
==



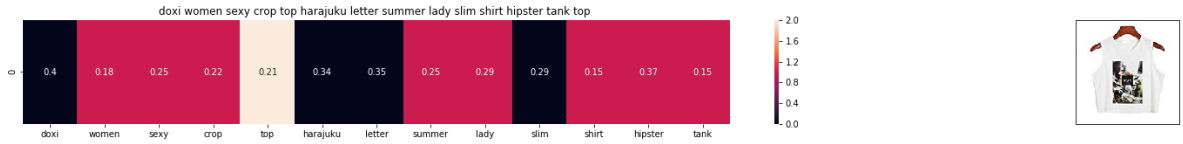
ASIN : B010V3A23U
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.9303848929561549
==
==



ASIN : B0124E7MHS
BRAND : Doxi Supermall
Eucliden distance from the given image : 0.9334421080980748
==
==



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 [27]:

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

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

In [28]:

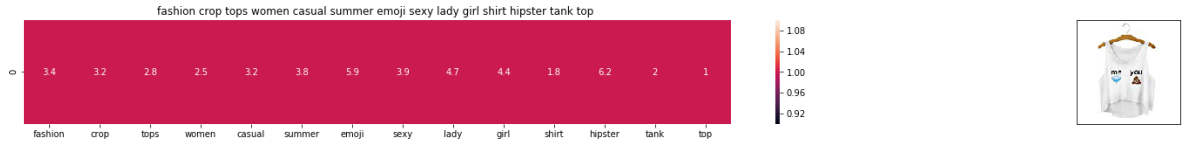
```
def n_containing(word):
    # return the number of documents which had the given word
    return sum(1 for blob in data['title'] if word in blob.split())

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

In [29]:

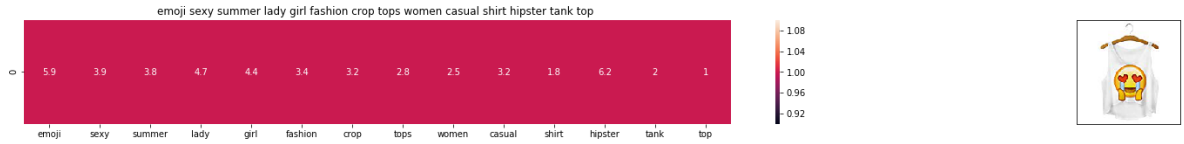
```
# we need to convert the values into float
idf_title_features  = idf_title_features.astype(np.float)

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 valu
    # idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0] will return a
    for j in idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0]:

        # we replace the count values of word i in document j with  idf_value of word i
        # idf_title_features[doc_id, index_of_word_in_courpus] = idf value of word
        idf_title_features[j,idf_title_vectorizer.vocabulary_[i]] = idf_val
```

In [30]:

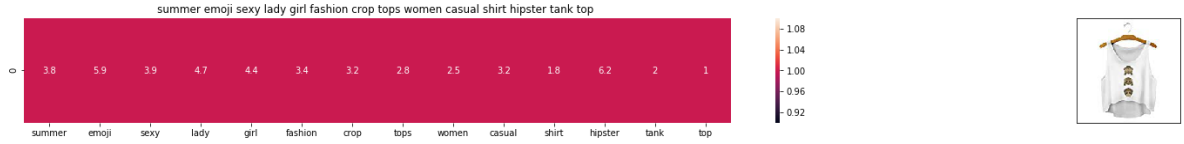
```
def idf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance from given input apparel to all remaining appar
    # the metric we used here is cosine, the coside distance is mesured as K(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_indice
        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
```



ASIN : B010V3B44G
Brand : Doxi Supermall
euclidean distance from the given image : 0.0
==========
==========

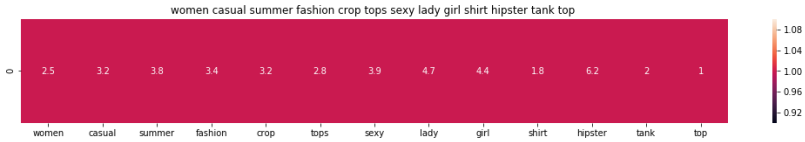


ASIN : B010V3BLWQ
Brand : Doxi Supermall
euclidean distance from the given image : 0.0
==========
==========

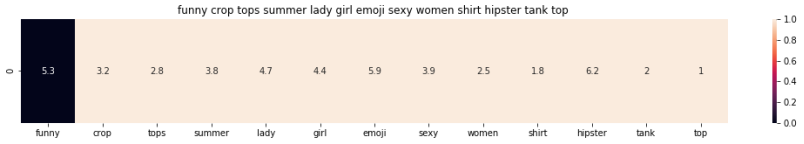


ASIN : B010V3BDII
Brand : Doxi Supermall
euclidean distance from the given image : 0.0

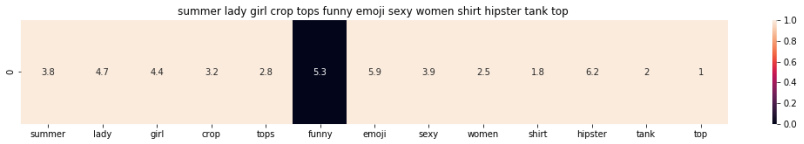
==========
==========



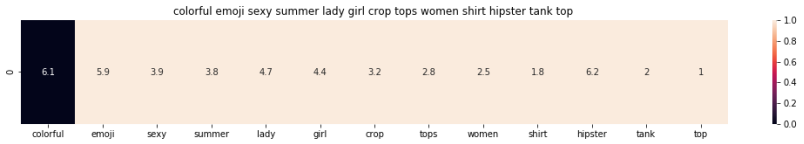
ASIN : B010V3AYSS
Brand : Doxi Supermall
euclidean distance from the given image : 5.922978852180059
==========
==========



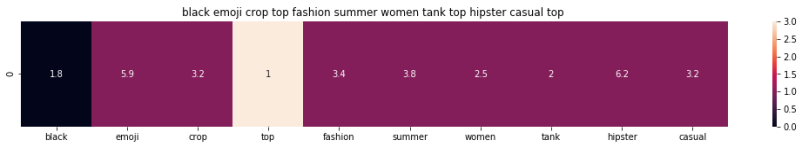
ASIN : B010V3C116
Brand : Doxi Supermall
euclidean distance from the given image : 7.037272185298332
==========
==========



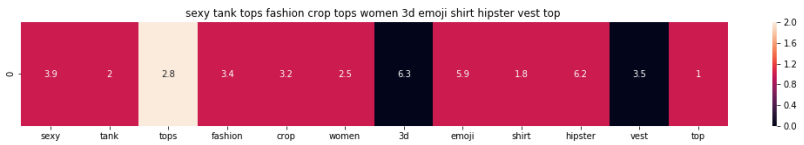
ASIN : B010V3BVMQ
Brand : Doxi Supermall
euclidean distance from the given image : 7.037272185298332
==========
==========



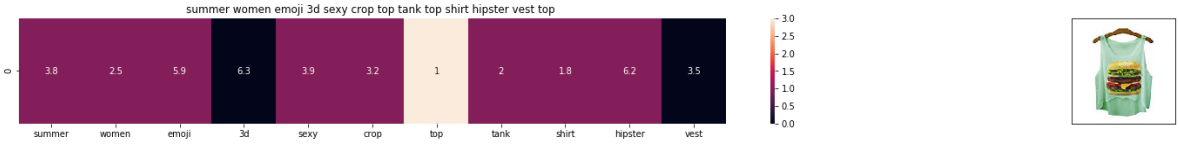
ASIN : B010V3BQZS
Brand : Doxi Supermall
euclidean distance from the given image : 7.667939547088641
==========
==========



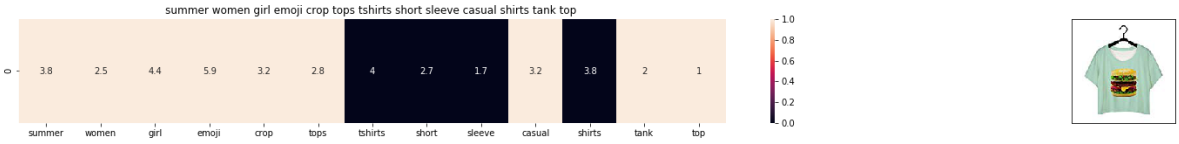
ASIN : B0124E80M4
Brand : Doxi Supermall
euclidean distance from the given image : 8.39863603811248
==========
==========



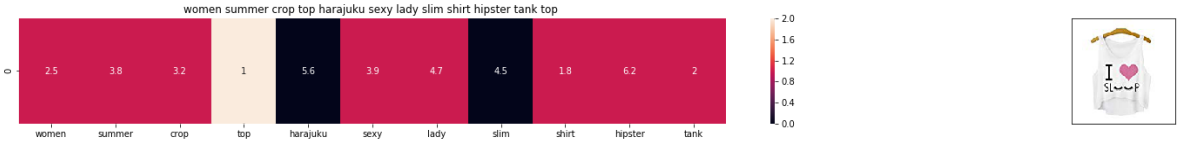
ASIN : B010V3DB9C
Brand : Doxi Supermall
euclidean distance from the given image : 10.835090137343855
==========
==========



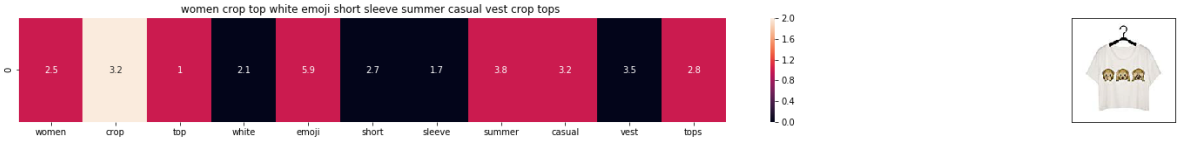
ASIN : B010V3E5EC
Brand : Doxi Supermall
euclidean distance from the given image : 11.060530175472811
==========
==========



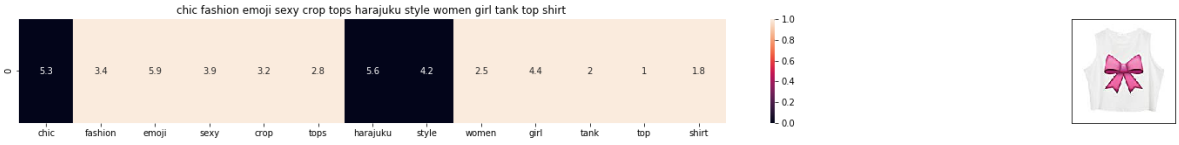
ASIN : B0124ECIU4
Brand : Doxi Supermall
euclidean distance from the given image : 11.456017724459604
==========
==========



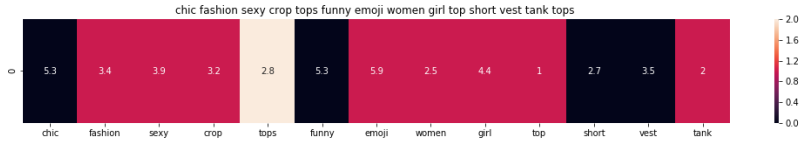
ASIN : B010V3EDEE
Brand : Doxi Supermall
euclidean distance from the given image : 11.635265990125815
==========
==========



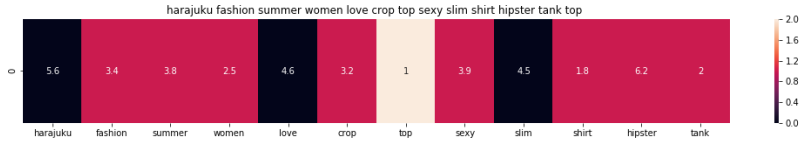
ASIN : B0124E7MHS
Brand : Doxi Supermall
euclidean distance from the given image : 11.844834283822053
==========
==========



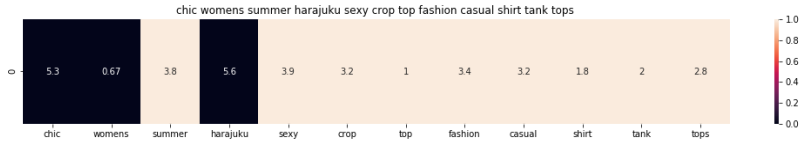
ASIN : B011RCJPR8
Brand : Chiclook Cool
euclidean distance from the given image : 12.760692810178545
==========
==========



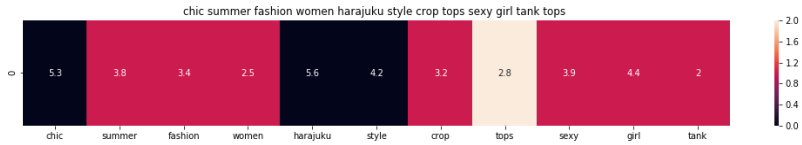
ASIN : B011RCJH58
Brand : Chiclook Cool
euclidean distance from the given image : 12.829128504563066
==========
==========



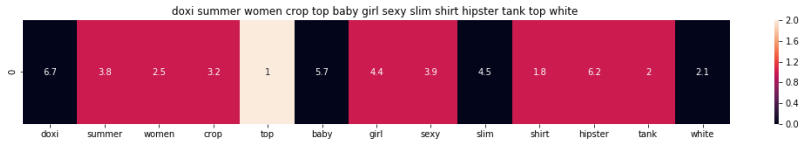
ASIN : B010V35OBU
Brand : Doxi Supermall
euclidean distance from the given image : 12.92180694297083
==========
==========



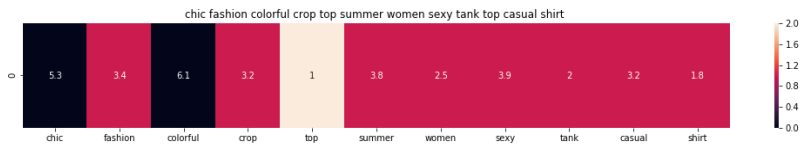
ASIN : B011RCJEMO
Brand : Chiclook Cool
euclidean distance from the given image : 13.474555736106787
==========
==========



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 [44]:

```
# 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[44]:

```
'\n# Set values for various parameters\nnum_features = 300    # Word vector
dimensionality                      \nmin_word_count = 1    # Minimum word c
ount                          \nnum_workers = 4       # Number of threads to r
un in parallel\ncontext = 10          # Context window size
\ndownsampling = 1e-3   # Downsample setting for frequent words\n\n# Initial
ize and train the model (this will take some time)\nfrom gensim.models impor
t word2vec\nprint ("Training model...")\nmodel = word2vec.Word2Vec(sen_corpu
s, workers=num_workers,            size=num_features, min_count = min_word_
count,            window = context)\n    \n'
```

In [45]:

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

# in this project we are using a pretrained model by google
# its 3.3G file, once you load this into your memory
# it occupies ~9Gb, so please do this step only if you have >12G of ram
# we will provide a pickle file wich contains a dict ,
# and it contains all our courpus words as keys and  model[word] as values
# To use this code-snippet, download "GoogleNews-vectors-negative300.bin"
# from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
# it's 1.9GB in size.

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

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

In [46]:

```
# 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]]
            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
            vec.append(np.zeros(shape=(300,)))
    # we will return a numpy array of shape (#number of words in title * 300 ) 300 = len(w2
    # each row represents the word2vec representation of each word (weighted/avg) in given
    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 co
    # vec2 = np.array(#number_of_words_title2 * 300), each row is a vector of length 300 co

    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) o
    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) o
    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
```

```
    gs = gridspec.GridSpec(2, 2, width_ratios=[4,1],height_ratios=[2,1])
    fig = plt.figure(figsize=(15,15))

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

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

    plt.show()
```

In [47]:

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

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

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

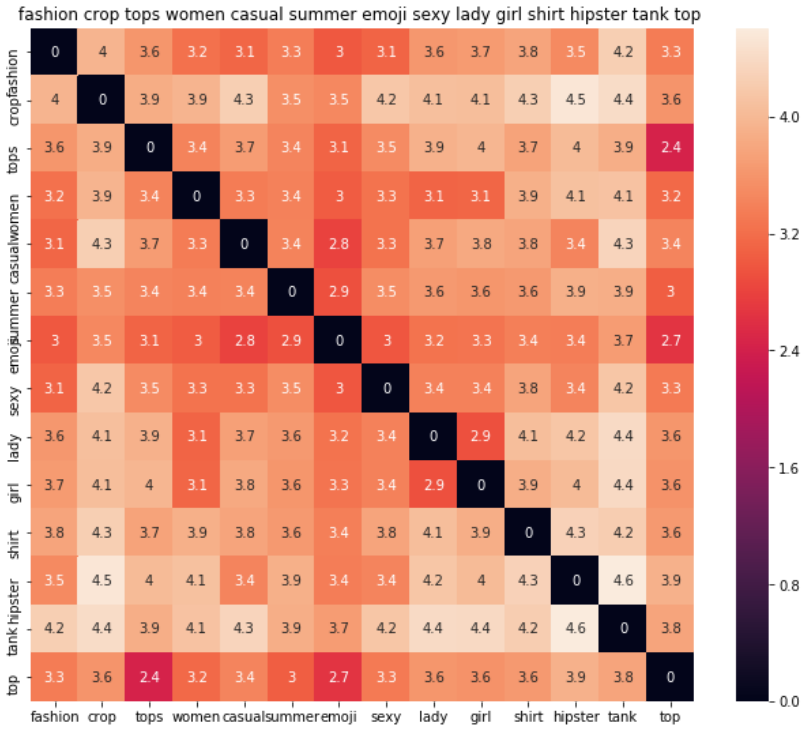
[9.2] Average Word2Vec product similarity.

In [48]:

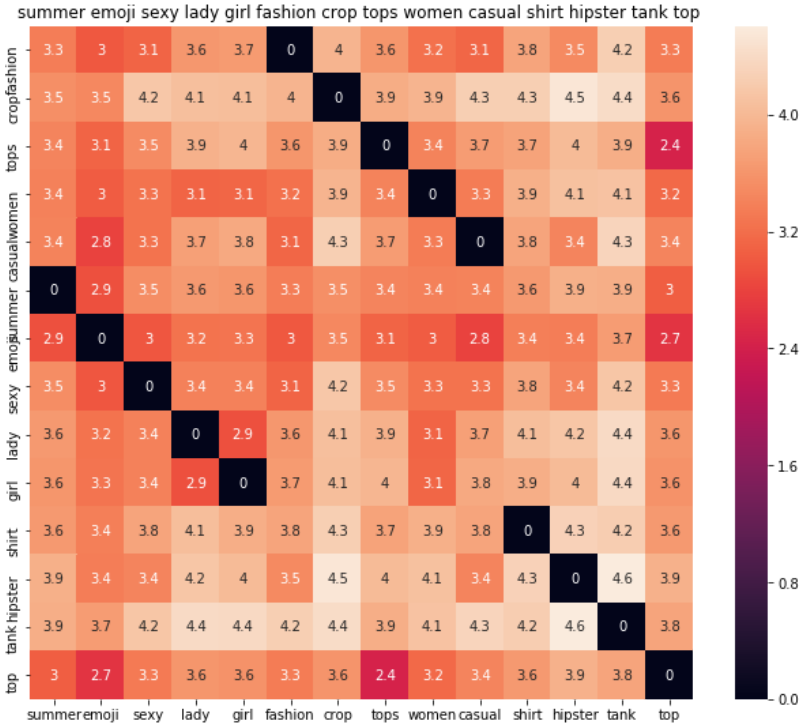
```
doc_id = 0
w2v_title = []
# for every title we build a avg vector representation
for i in data['title']:
    w2v_title.append(build_avg_vec(i, 300, doc_id,'avg'))
    doc_id += 1

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

In [49]:

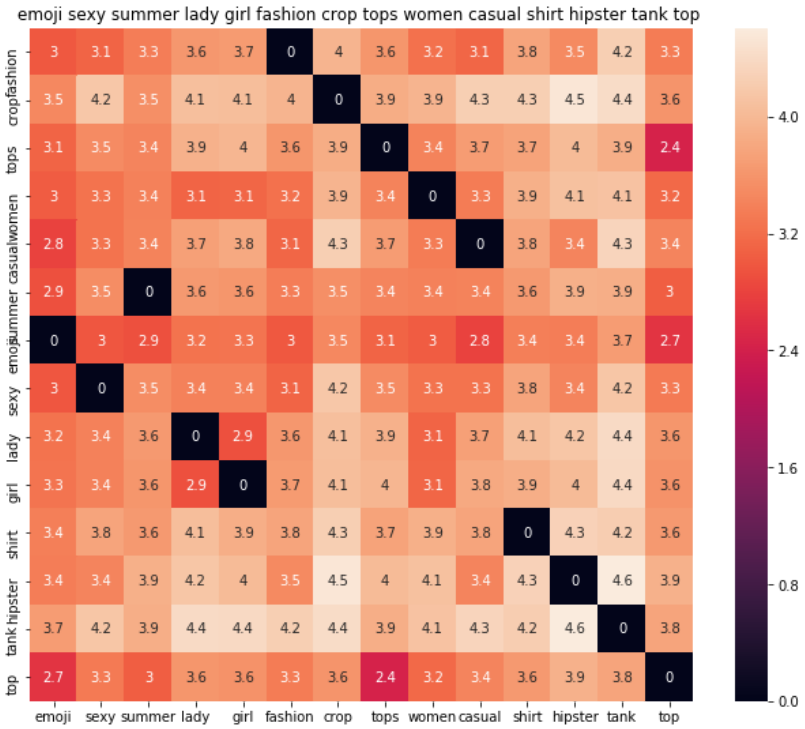
```
def avg_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

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

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

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

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



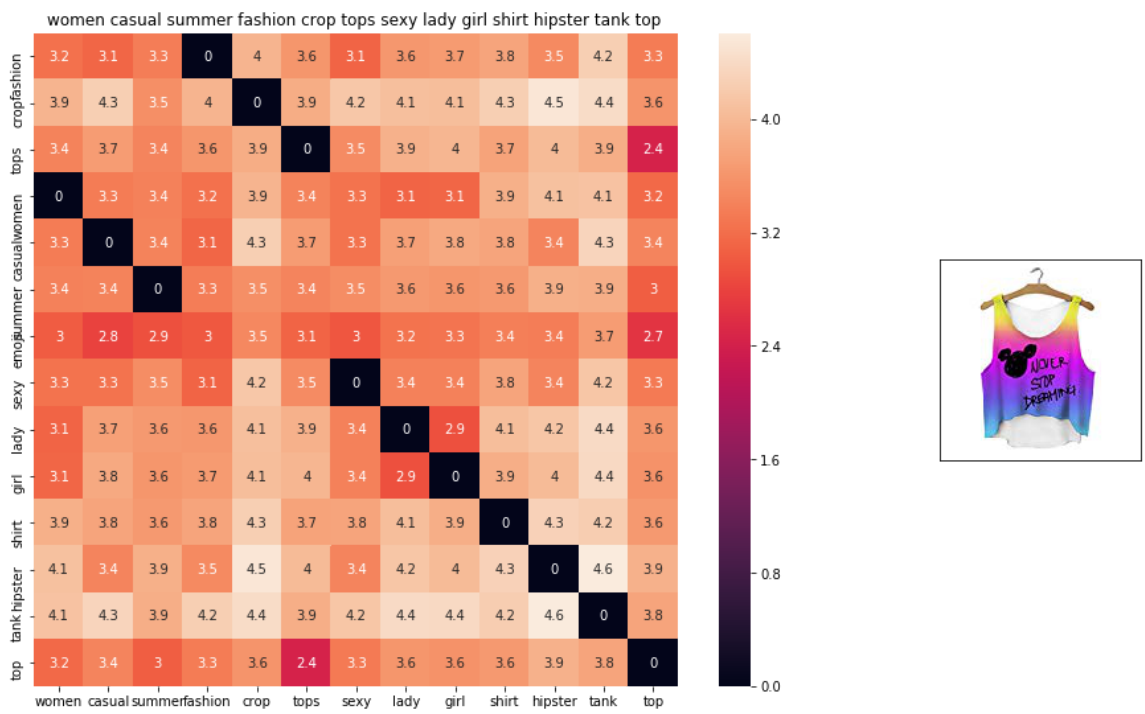

```
ASIN : B010V3B44G
BRAND : Doxi Supermall
euclidean distance from given input image : 0.0
=============================================================================
=============================================
```



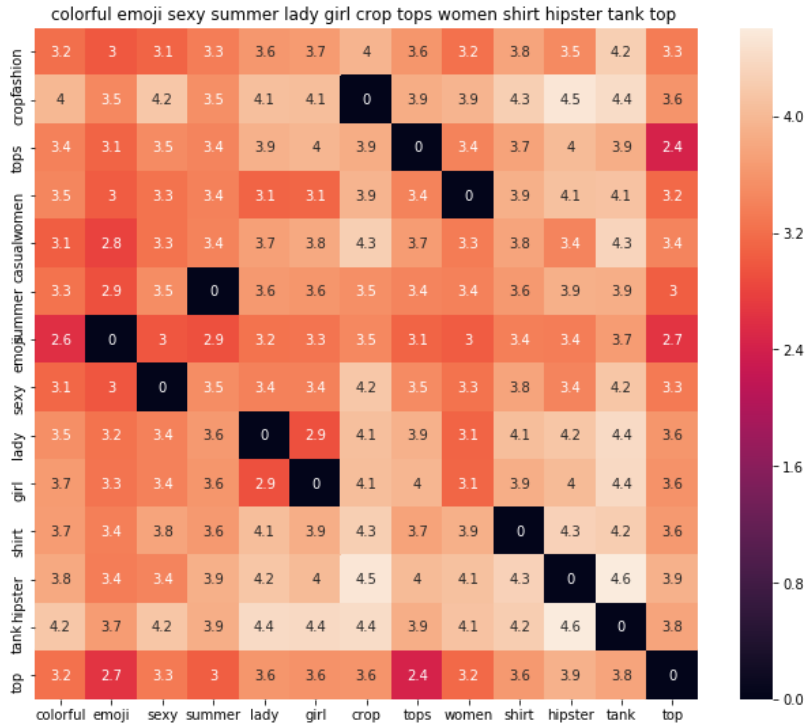
ASIN : B010V3BDII
BRAND : Doxi Supermall
euclidean distance from given input image : 0.0
==========
==========



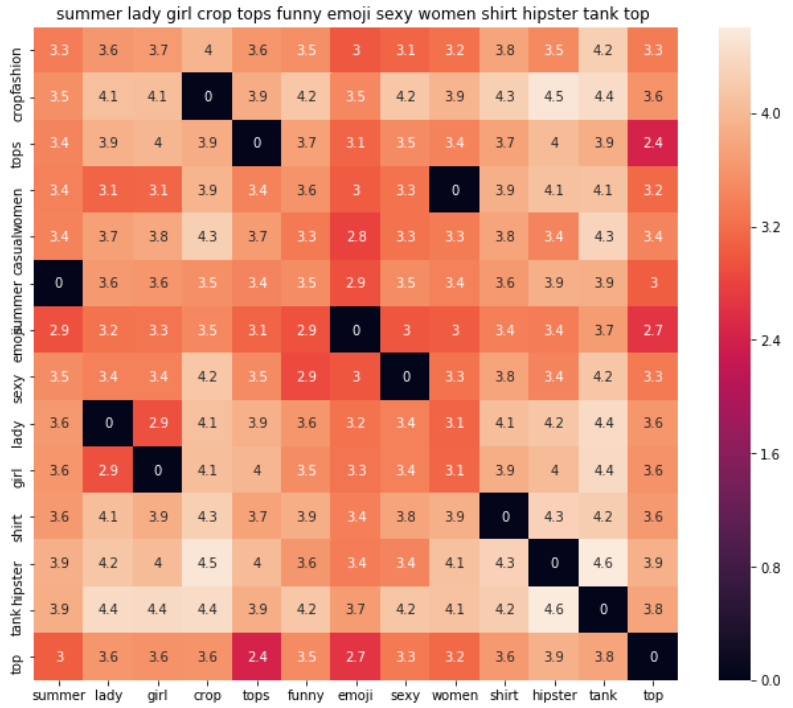
ASIN : B010V3BLWQ
BRAND : Doxi Supermall
euclidean distance from given input image : 0.0
==========
==========



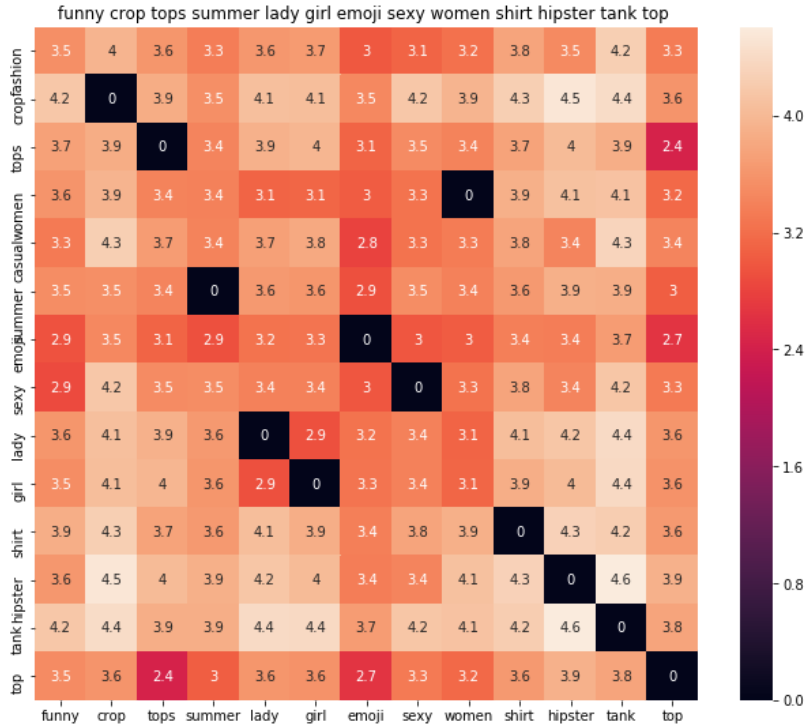
ASIN : B010V3AYSS
BRAND : Doxi Supermall
euclidean distance from given input image : 0.13368031
==========
==========



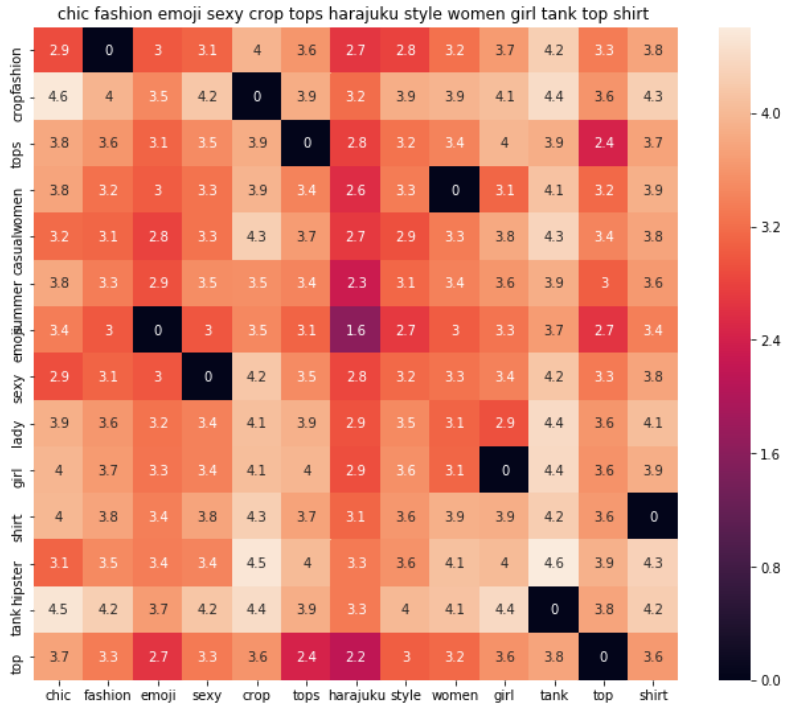
ASIN : B010V3BQZS
BRAND : Doxi Supermall
euclidean distance from given input image : 0.2983774
==========
==========



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

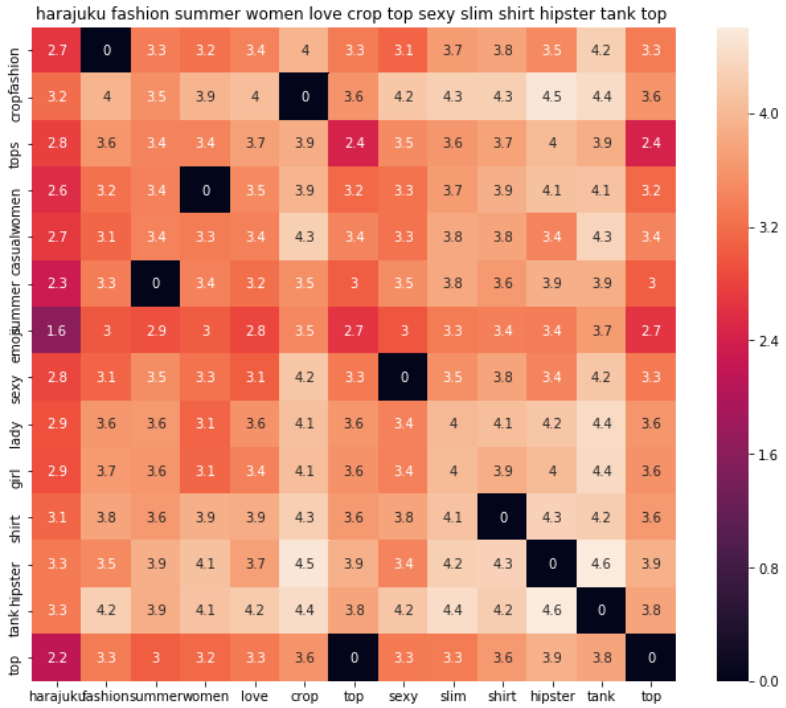


ASIN : B010V3C116
BRAND : Doxi Supermall
euclidean distance from given input image : 0.3274633
==========
==========

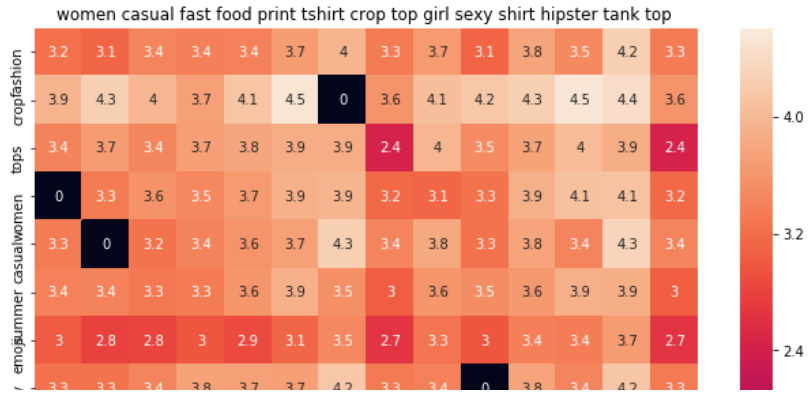


ASIN : B011RCJPR8
BRAND : Chiclook Cool
euclidean distance from given input image : 0.4097583
==========
==========

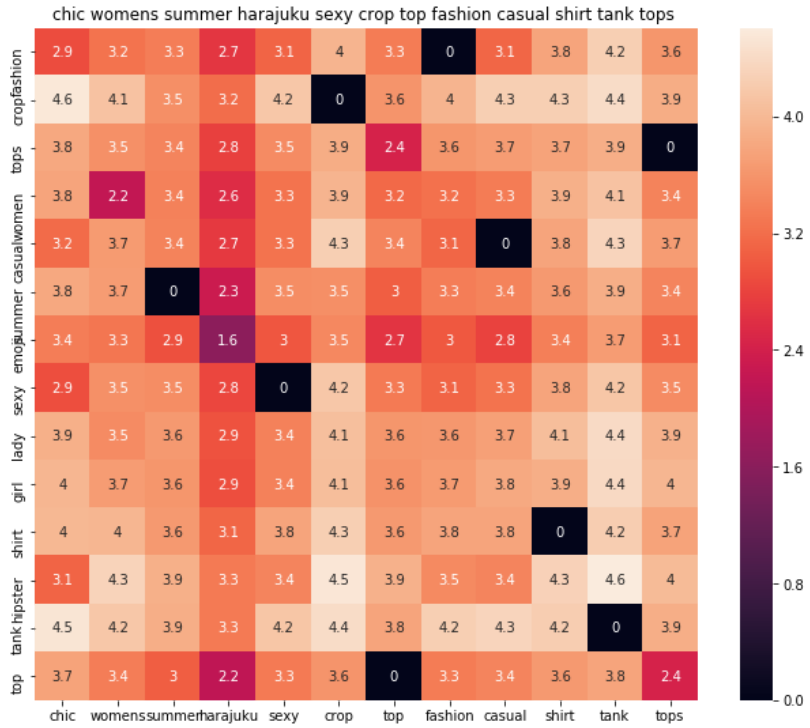
ASIN : B010V3EDEE
BRAND : Doxi Supermall
euclidean distance from given input image : 0.46806565
==========
==========



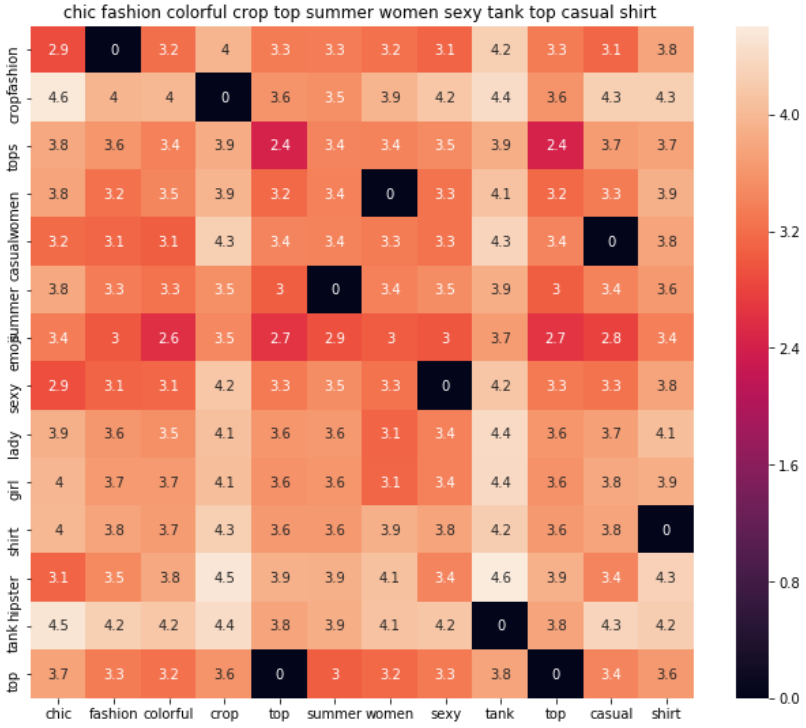
ASIN : B010V35OBU
BRAND : Doxi Supermall
euclidean distance from given input image : 0.49573147
==========
==========



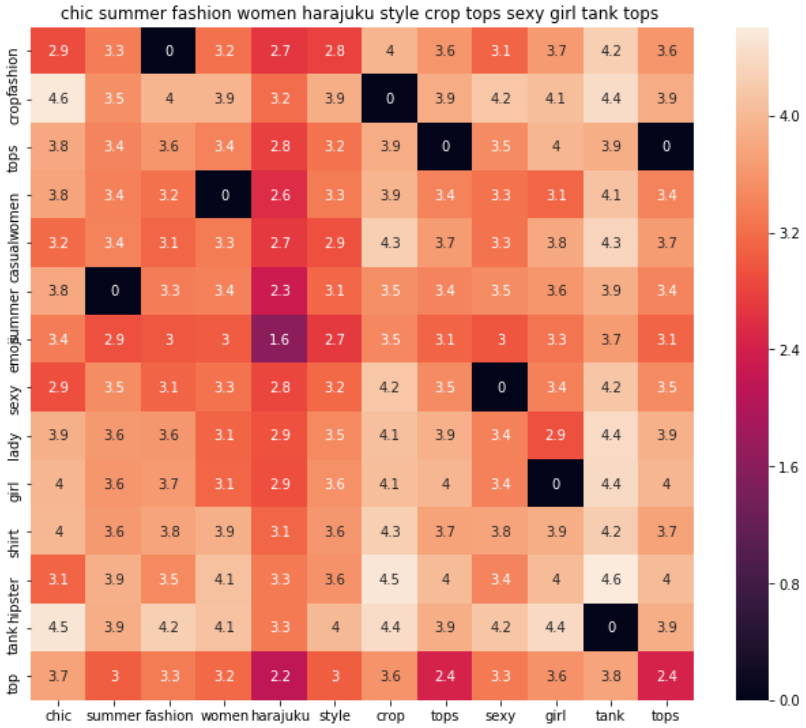
ASIN : B010V3AB5O
BRAND : Doxi Supermall
euclidean distance from given input image : 0.5010753
==========
==========



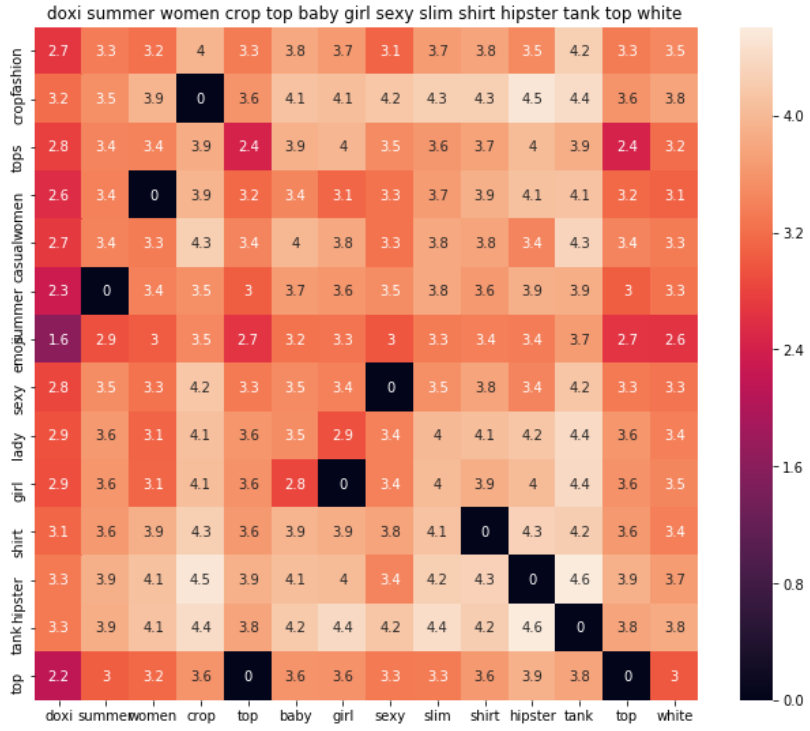
ASIN : B011RCJEMO
BRAND : Chiclook Cool
euclidean distance from given input image : 0.50272816
==========
==========



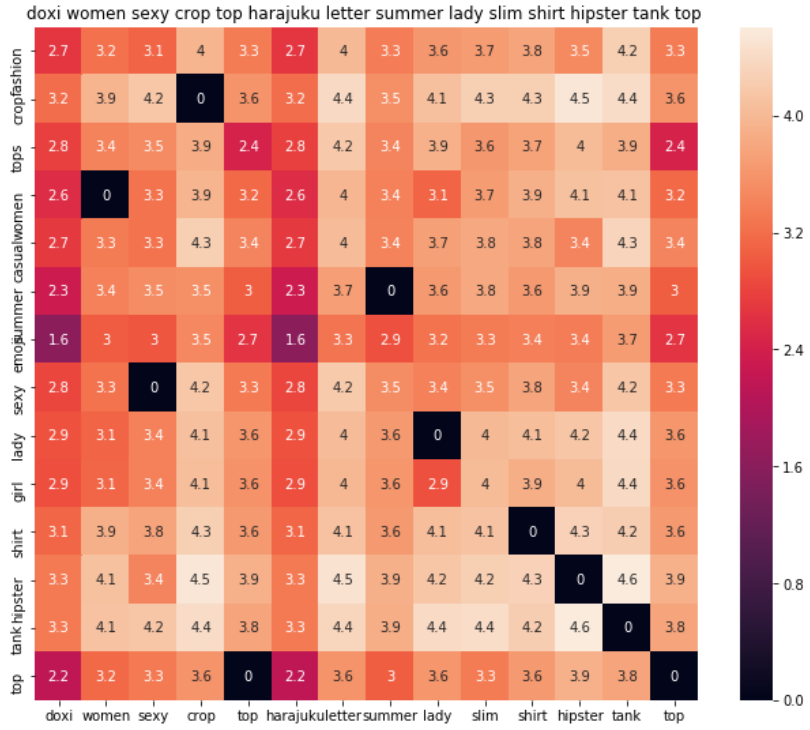
ASIN : B011RCJ6UE
BRAND : Chiclook Cool
euclidean distance from given input image : 0.5073643
==========
==========



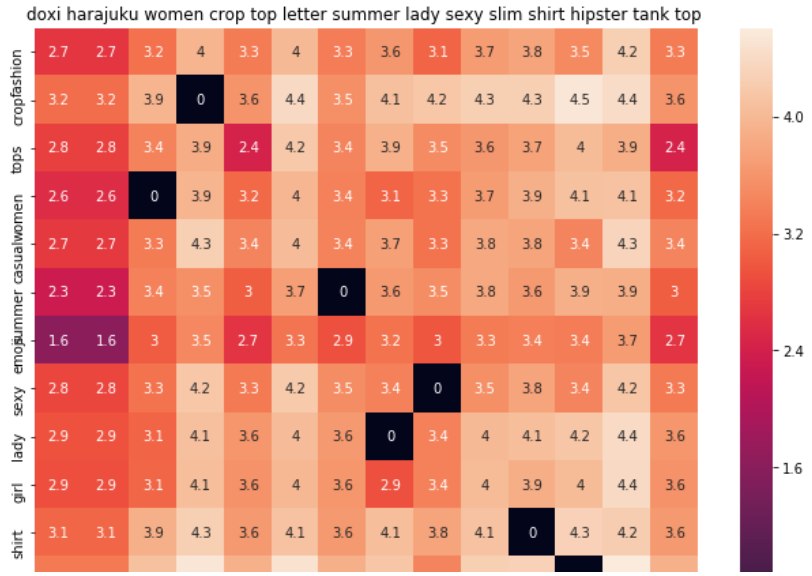
ASIN : B011OU51US
BRAND : Chiclook Cool
euclidean distance from given input image : 0.516089
==========
==========



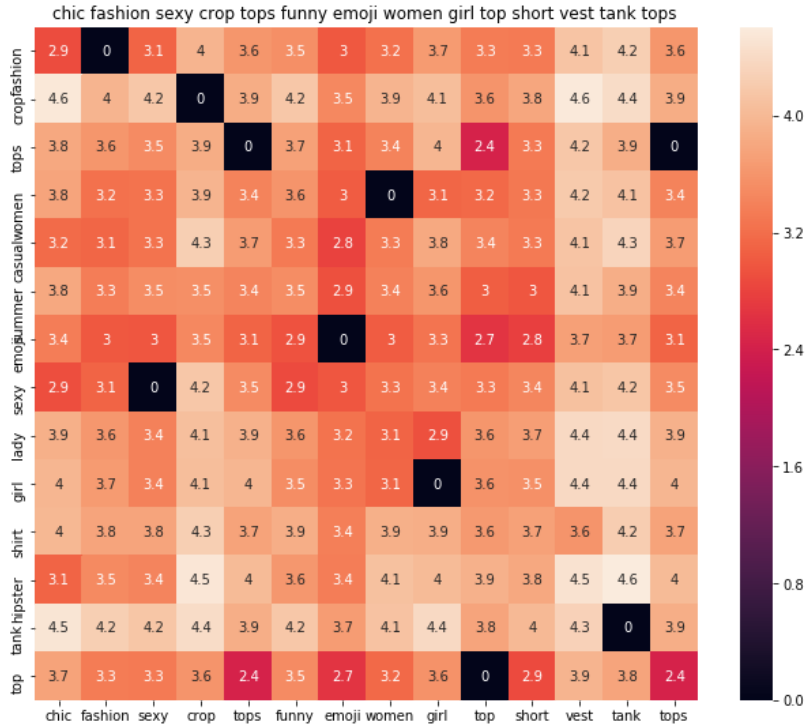
ASIN : B010V3A23U
BRAND : Doxi Supermall
euclidean distance from given input image : 0.5262786
==========
==========



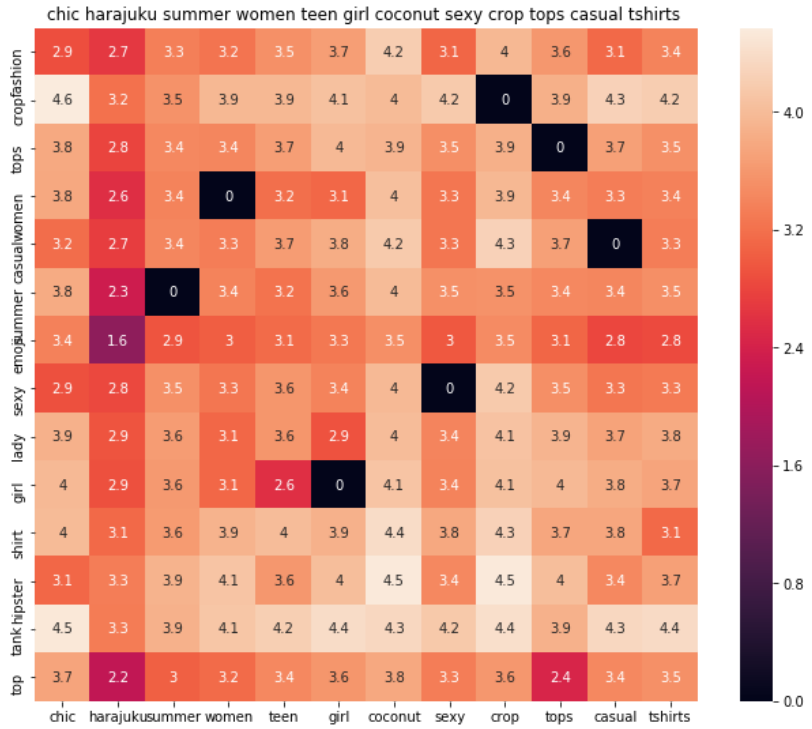
ASIN : B010V39146
BRAND : Doxi Supermall
euclidean distance from given input image : 0.5270717
==========
==========



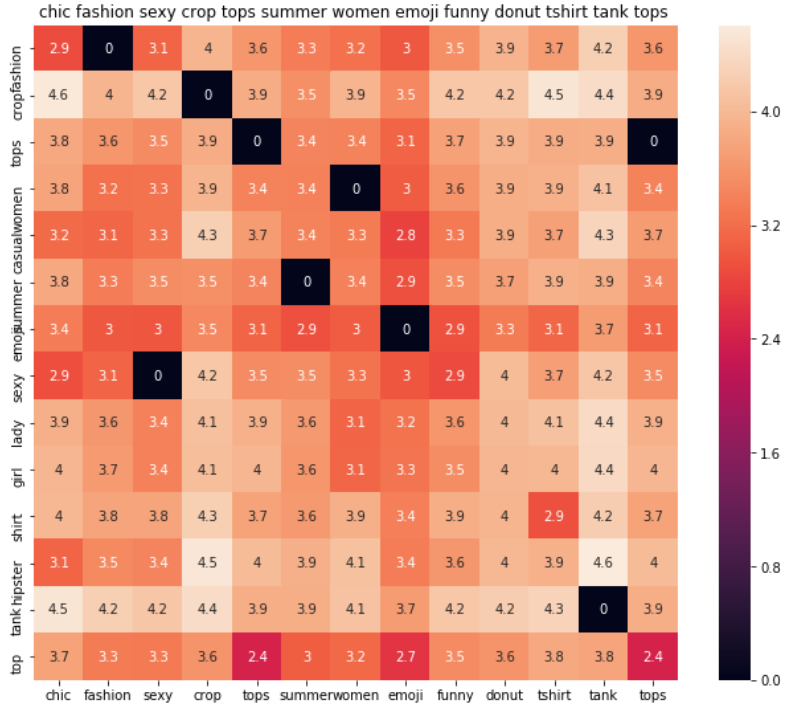
ASIN : B010V380LQ
BRAND : Doxi Supermall
euclidean distance from given input image : 0.5270717
==========
==========



ASIN : B011RCJH58
BRAND : Chiclook Cool
euclidean distance from given input image : 0.52816975
==========
==========



ASIN : B011OU4R08
BRAND : Chiclook Cool
euclidean distance from given input image : 0.55456346
==========
==========



```
ASIN : B011UEUTQE
BRAND : Chiclook Cool
euclidean distance from given input image : 0.57349175
============================================================================
================================================
```

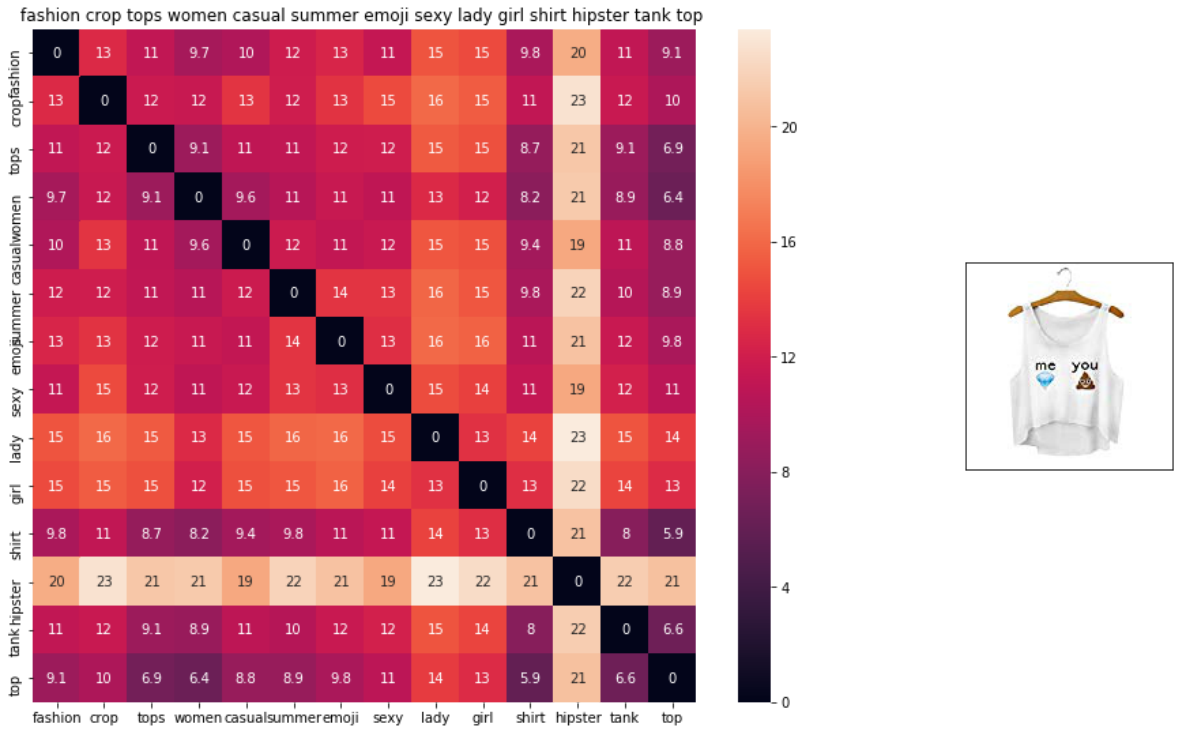
[9.4] IDF weighted Word2Vec for product similarity

In [50]:

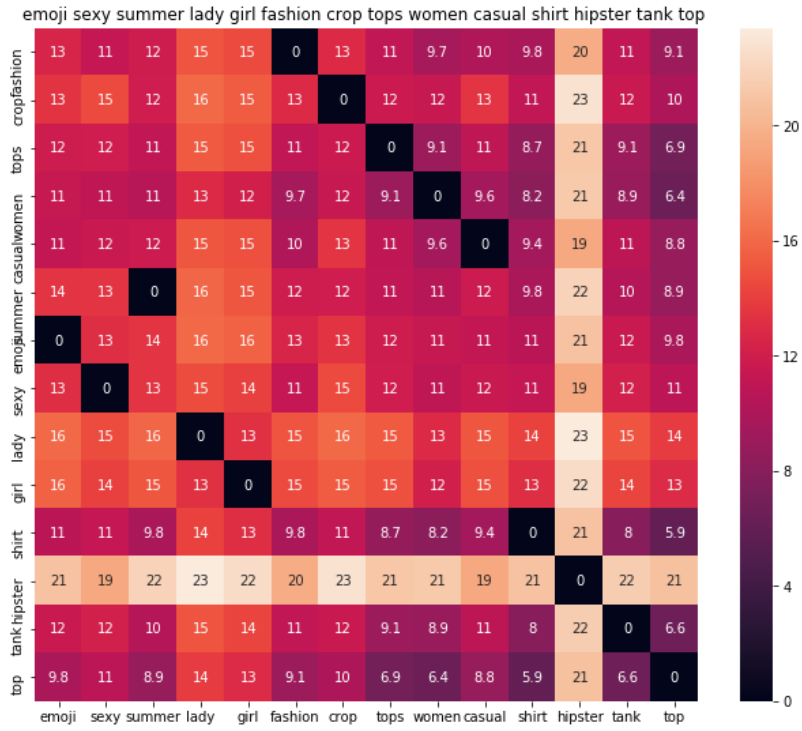
```
doc_id = 0
w2v_title_weight = []
# for every title we build a weighted vector representation
for i in data['title']:
    w2v_title_weight.append(build_avg_vec(i, 300, doc_id,'weighted'))
    doc_id += 1
# w2v_title = np.array(# number of doc in courpus * 300), each row corresponds to a doc
w2v_title_weight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

In [51]:

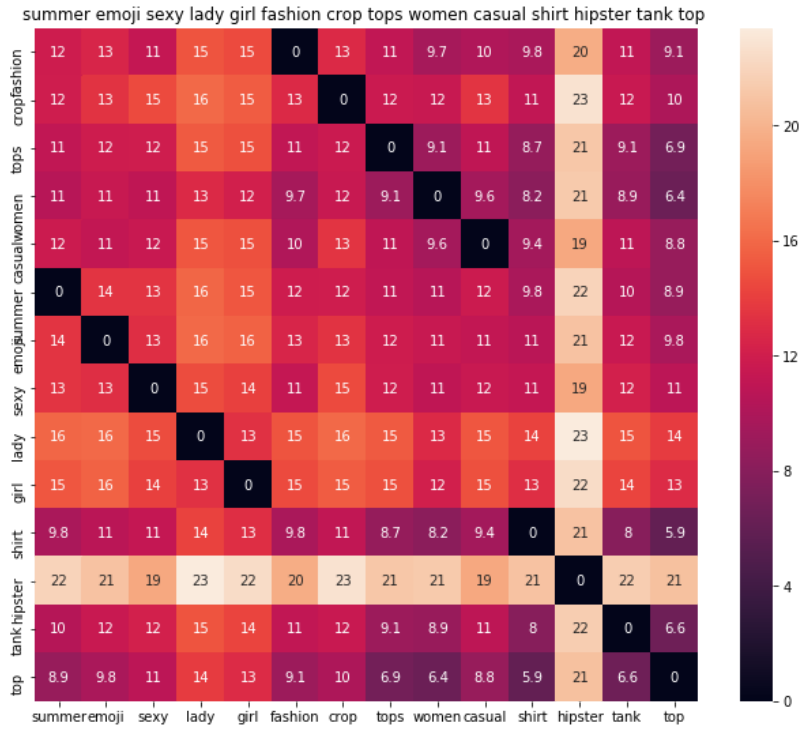
```
def weighted_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance from given input apparel to all remaining appar
    # the metric we used here is cosine, the coside distance is mesured as K(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

    # 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]], dat
        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
```



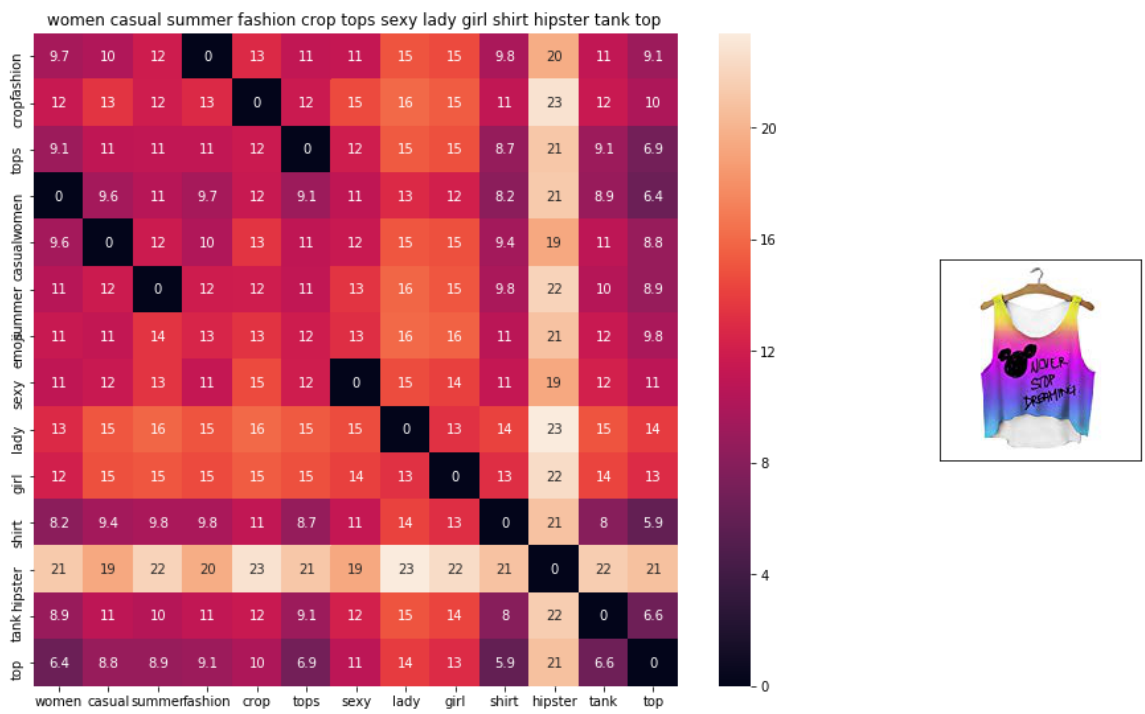

```
ASIN : B010V3B44G
Brand : Doxi Supermall
euclidean distance from input : 0.0
=============================================================================
===============================================
```



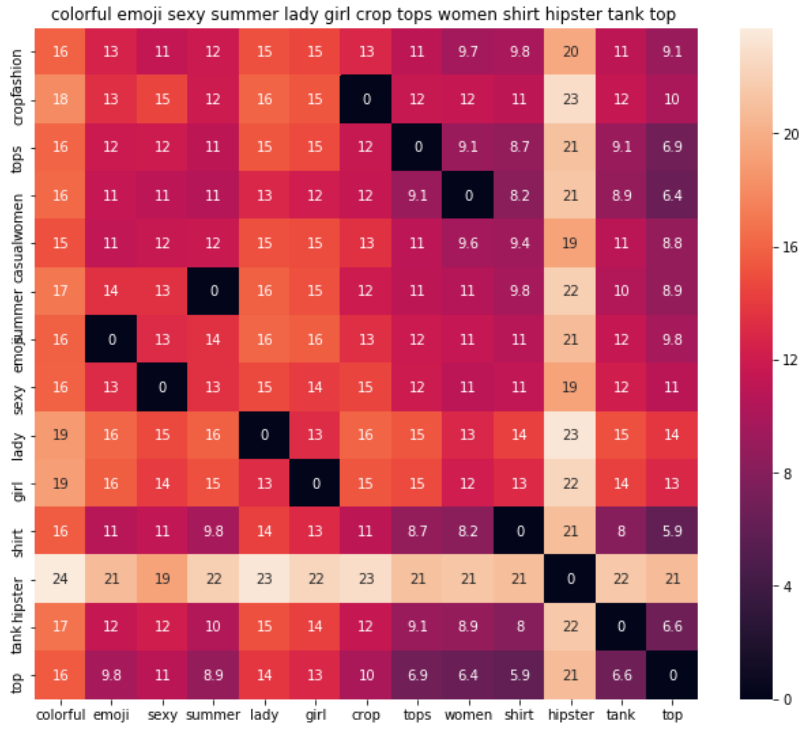
ASIN : B010V3BLWQ
Brand : Doxi Supermall
euclidean distance from input : 0.0
==========
==========



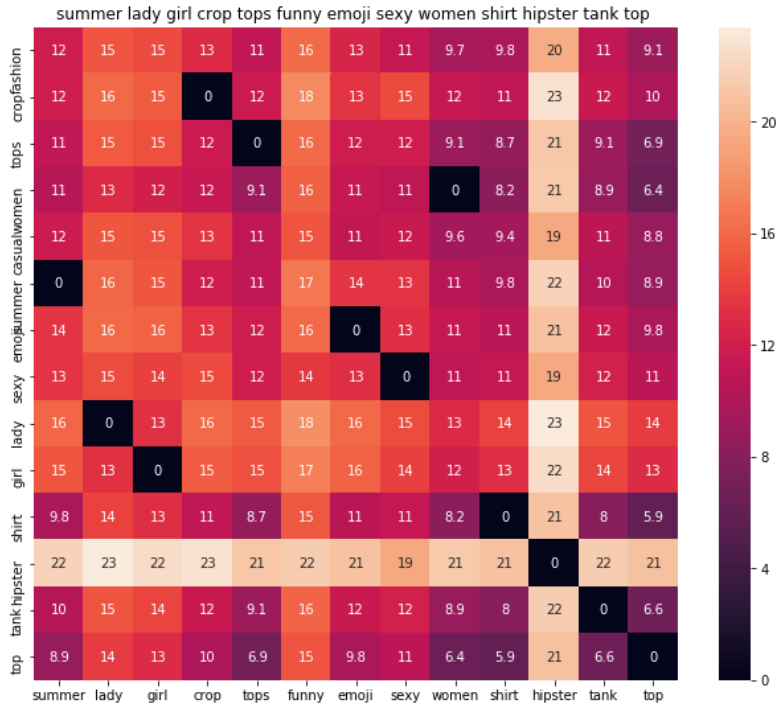
ASIN : B010V3BDII
Brand : Doxi Supermall
euclidean distance from input : 0.0
==========
==========



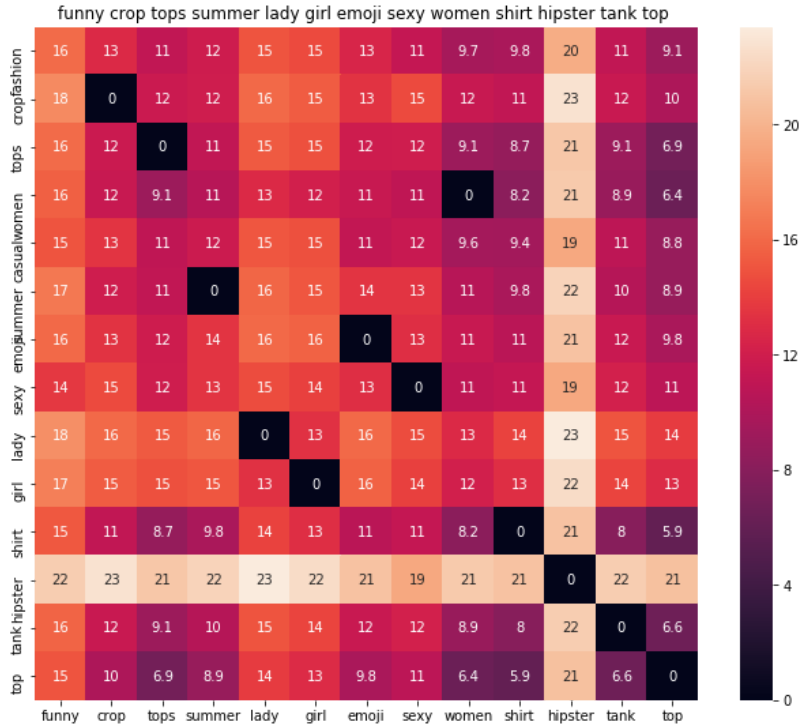
ASIN : B010V3AYSS
Brand : Doxi Supermall
euclidean distance from input : 0.6962336
==========
==========



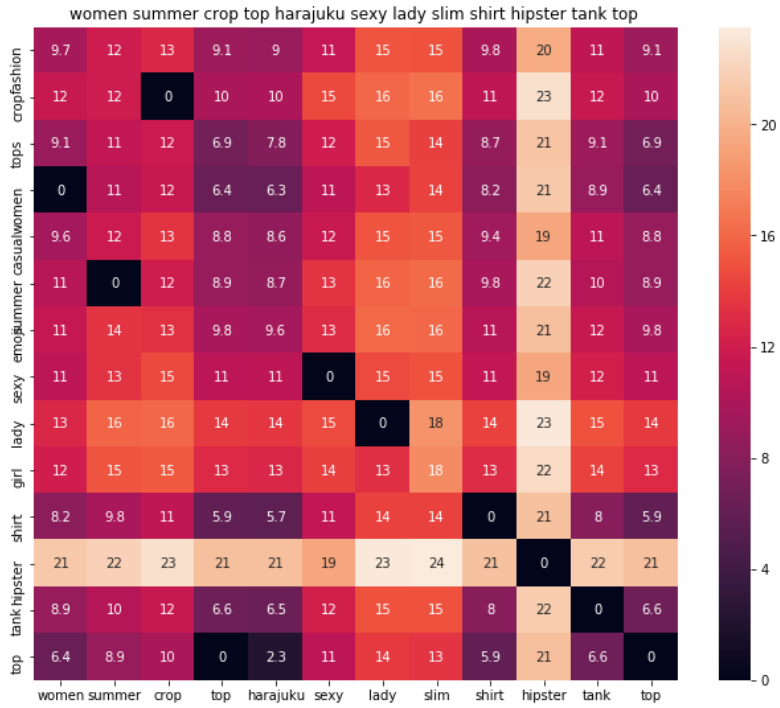
ASIN : B010V3BQZS
Brand : Doxi Supermall
euclidean distance from input : 1.287273
==========
==========



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

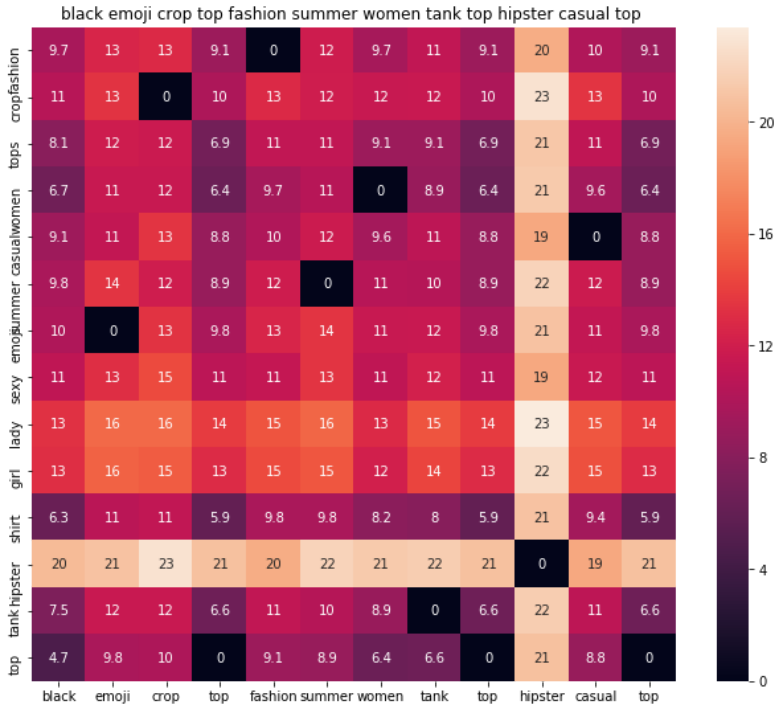


ASIN : B010V3C116
Brand : Doxi Supermall
euclidean distance from input : 1.3530484
==========
==========

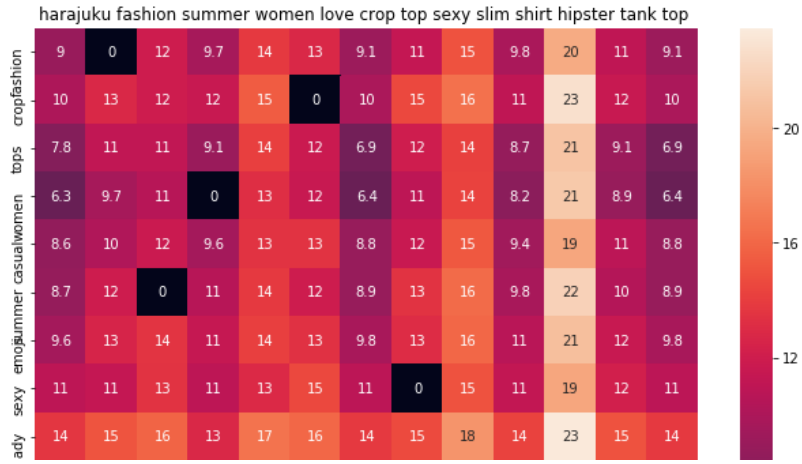


ASIN : B010V3EDEE
Brand : Doxi Supermall
euclidean distance from input : 1.743416
==========
==========

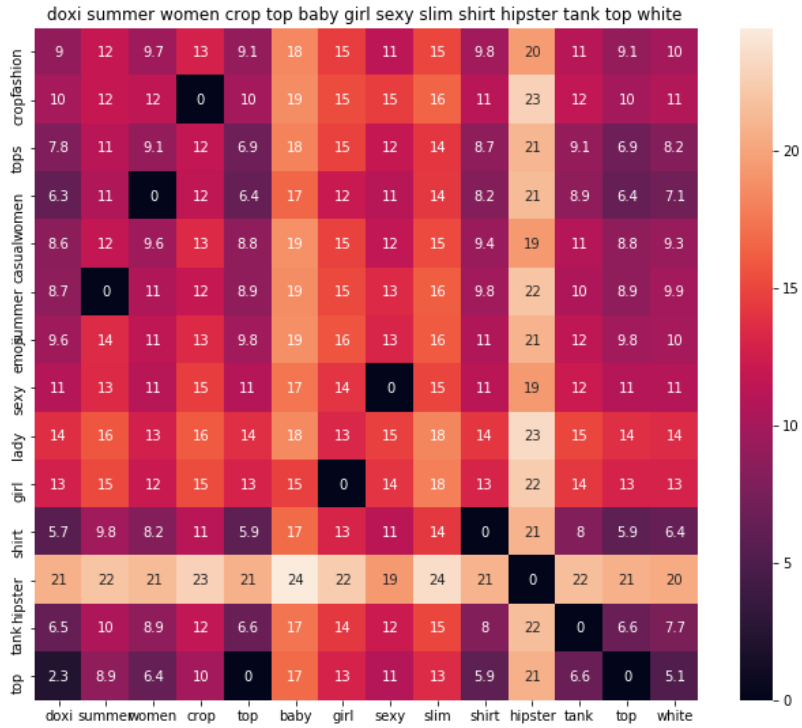
ASIN : B011RCJPR8
Brand : Chiclook Cool
euclidean distance from input : 1.9345253
==========
==========



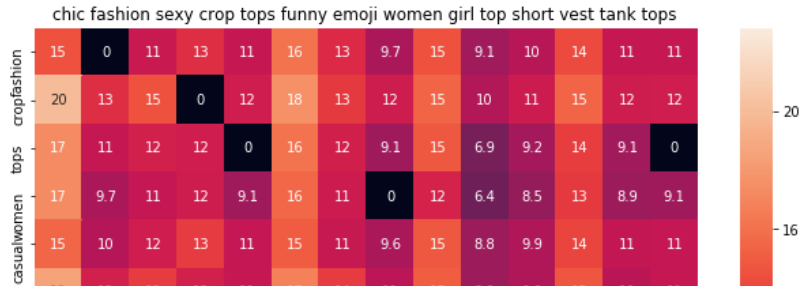
ASIN : B0124E80M4
Brand : Doxi Supermall
euclidean distance from input : 2.032144
==========
==========



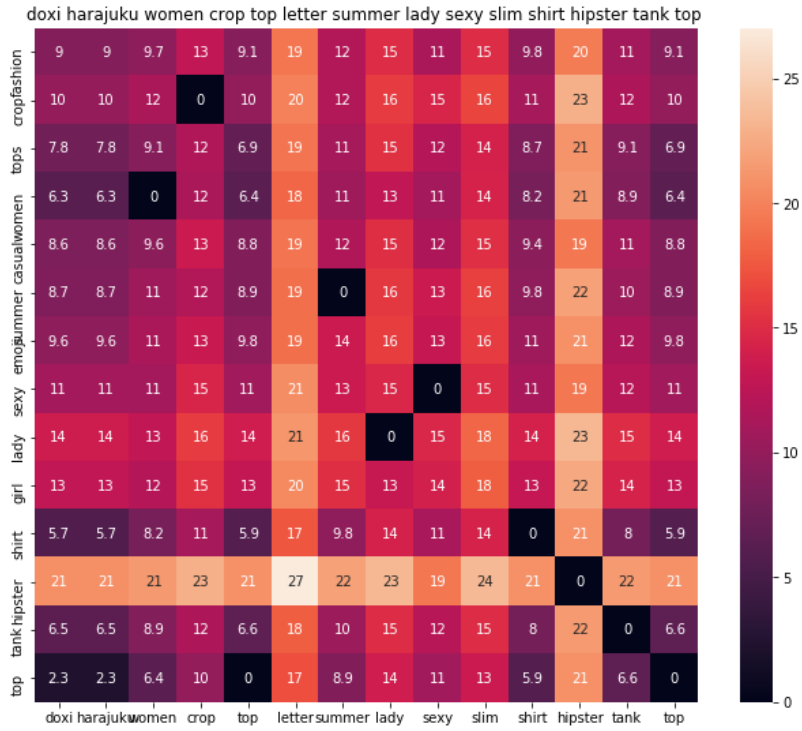
ASIN : B010V35OBU
Brand : Doxi Supermall
euclidean distance from input : 2.0631933
==========
==========



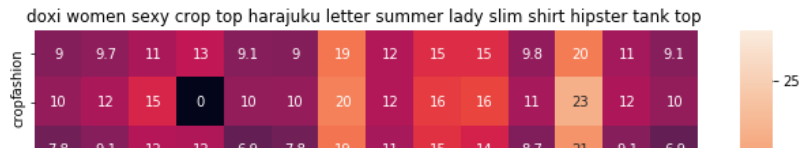
ASIN : B010V3A23U
Brand : Doxi Supermall
euclidean distance from input : 2.124369
==========
==========



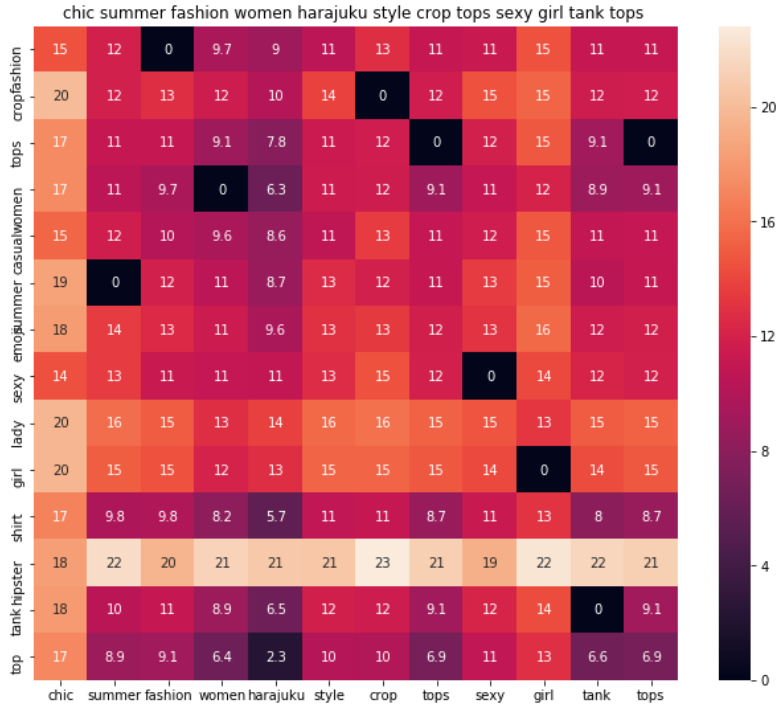
ASIN : B011RCJH58
Brand : Chiclook Cool
euclidean distance from input : 2.1507082
==========
==========



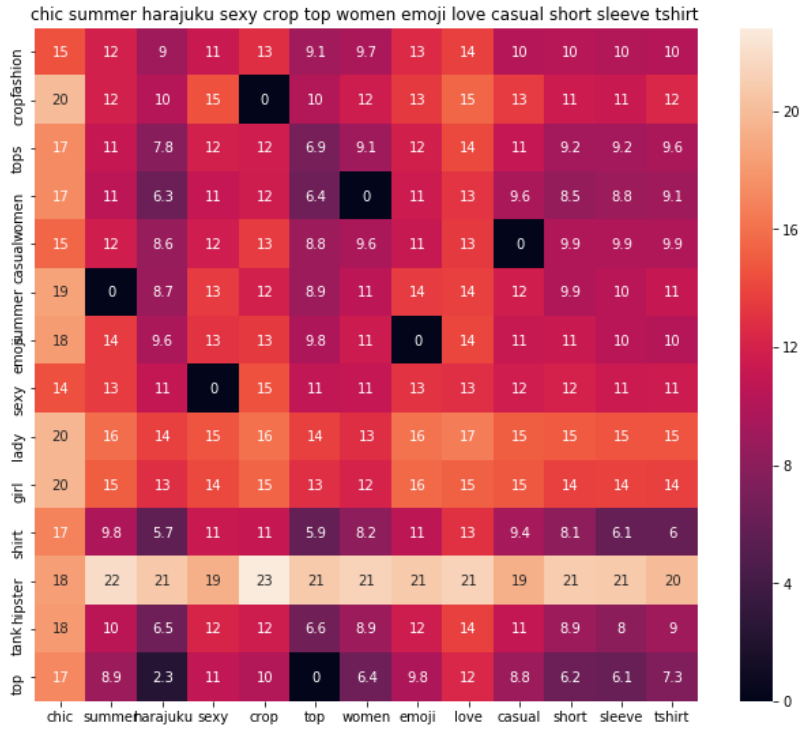
ASIN : B010V380LQ
Brand : Doxi Supermall
euclidean distance from input : 2.1674924
==========
==========



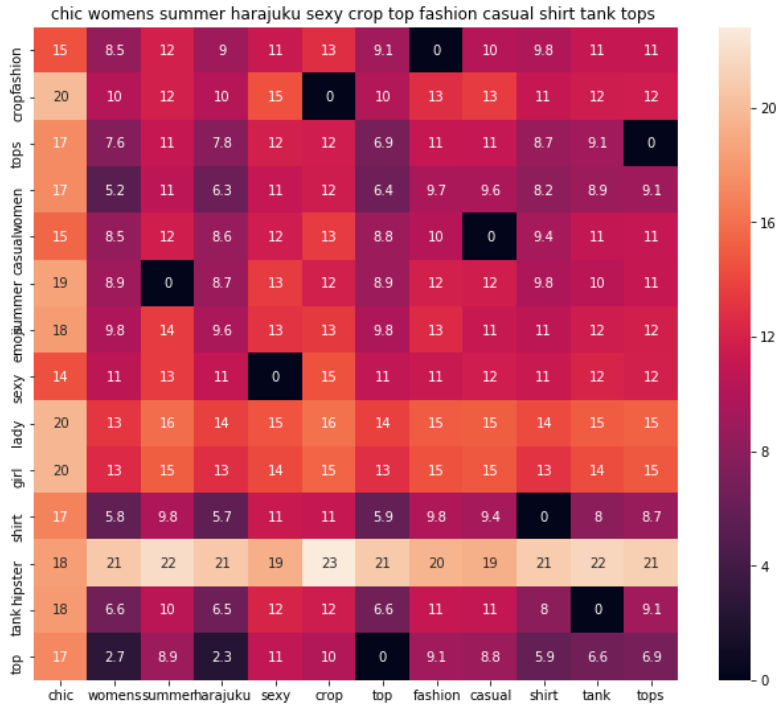
ASIN : B010V39146
Brand : Doxi Supermall
euclidean distance from input : 2.1674929
==========
==========



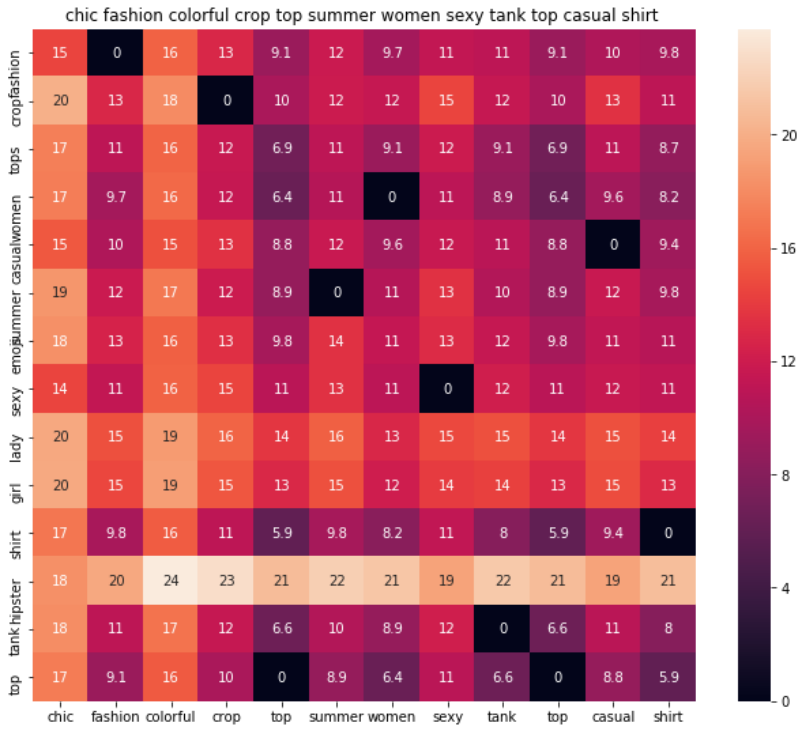
ASIN : B011OU51US
Brand : Chiclook Cool
euclidean distance from input : 2.2320182
==========
==========



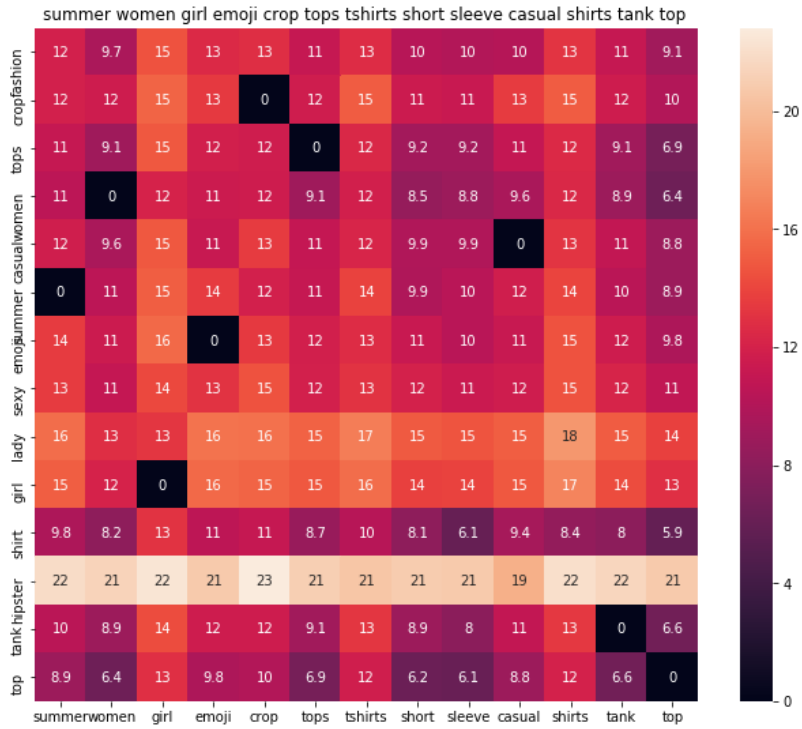
ASIN : B011UEVF4O
Brand : Chiclook Cool
euclidean distance from input : 2.313454
==========
==========



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



ASIN : B011RCJ6UE
Brand : Chiclook Cool
euclidean distance from input : 2.515184
==========
==========



```
ASIN : B0124ECIU4
Brand : Doxi Supermall
euclidean distance from input : 2.5520844
================================================================================
================================================
```

[9.6] Weighted similarity using brand and color.

In [52]:

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

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

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

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

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

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

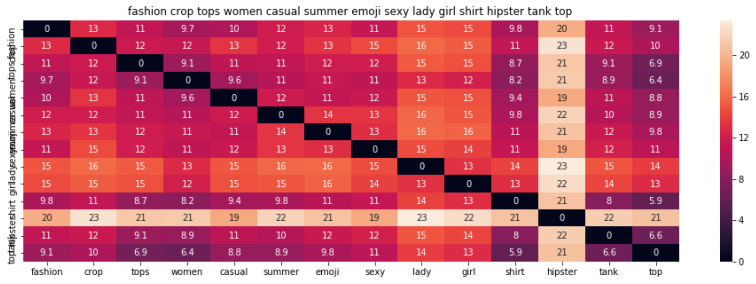
In [53]:

```
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) o
    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) o
    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]],
                [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2], types[doc_id2]]]

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

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

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

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

    # in last 25 * 10:15 grids we display image
    ax2 = plt.subplot(gs[:, 10:16])
    # we dont display grid lins and axis labels to images
    ax2.grid(False)
    ax2.set_xticks([])
    ax2.set_yticks([])

    # pass the url it display it
    display_img(url, ax2, fig)

    plt.show()
```

In [54]:

```
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 appar
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y>
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1
    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]
        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
```

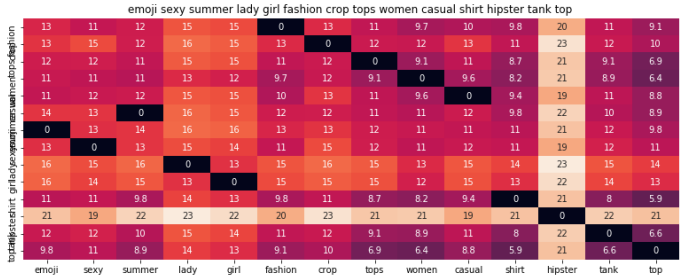
Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3B44G	Doxi-Supermall	White



ASIN : B010V3B44G
Brand : Doxi Supermall
euclidean distance from input : 0.0
==========
==========

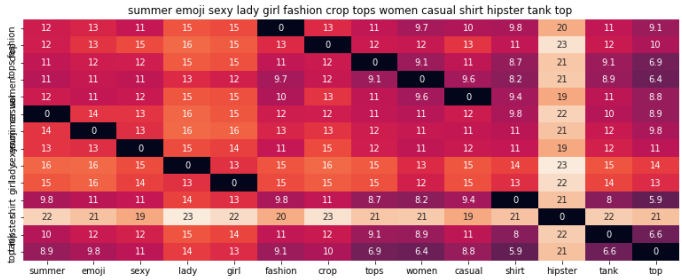
Asin	Brand	Color

B010V3B44G	Doxi-Supermall	White
B010V3BLWQ	Doxi-Supermall	White



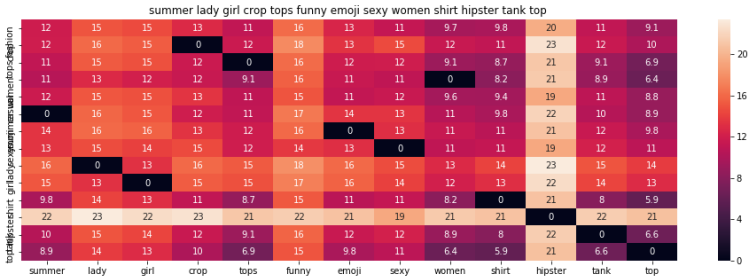
ASIN : B010V3BLWQ
Brand : Doxi Supermall
euclidean distance from input : 0.0
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3BDII	Doxi-Supermall	White



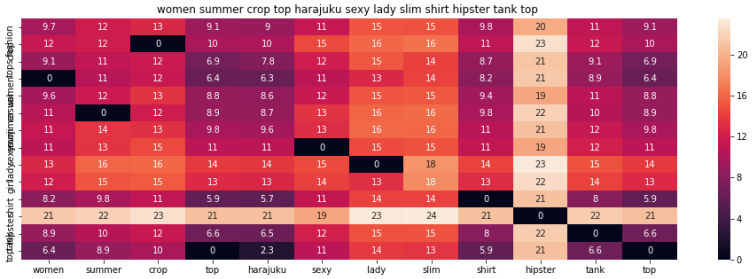
ASIN : B010V3BDII
Brand : Doxi Supermall
euclidean distance from input : 0.0
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3BVMQ	Doxi-Supermall	White



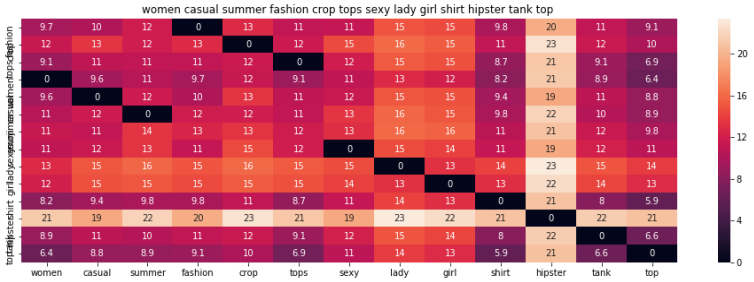
ASIN : B010V3BVMQ
Brand : Doxi Supermall
euclidean distance from input : 0.6765242099761963
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3EDEE	Doxi-Supermall	White



ASIN : B010V3EDEE
Brand : Doxi Supermall
euclidean distance from input : 0.8717080116271972
==========
==========

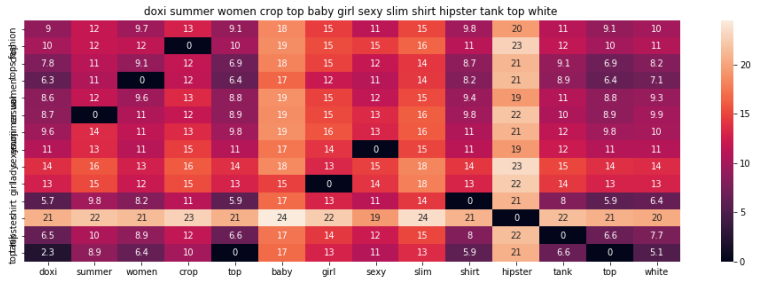
Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3AYSS	Doxi-Supermall	Multicoloured



ASIN : B010V3AYSS
Brand : Doxi Supermall
euclidean distance from input : 1.0552235605139402

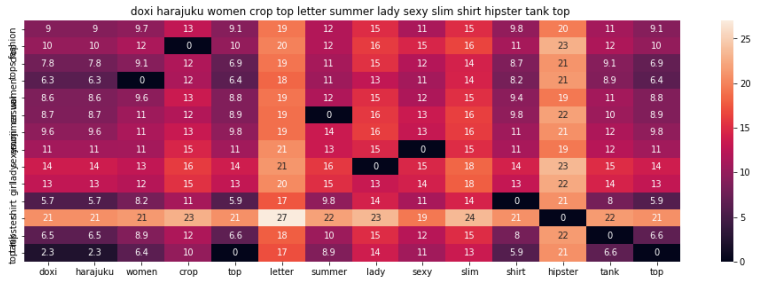
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3A23U	Doxi-Supermall	White



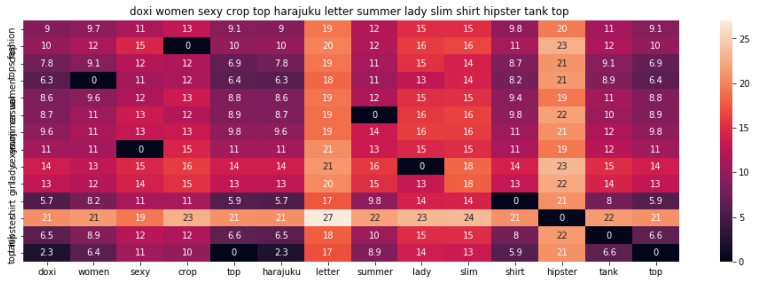
ASIN : B010V3A23U
Brand : Doxi Supermall
euclidean distance from input : 1.0621844291687013
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V380LQ	Doxi-Supermall	White



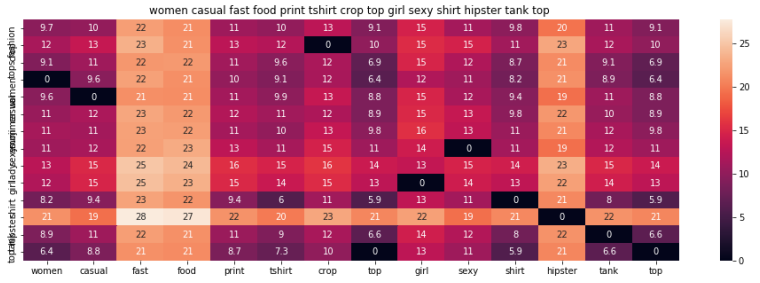
ASIN : B010V380LQ
Brand : Doxi Supermall
euclidean distance from input : 1.0837461471557617
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V39146	Doxi-Supermall	White



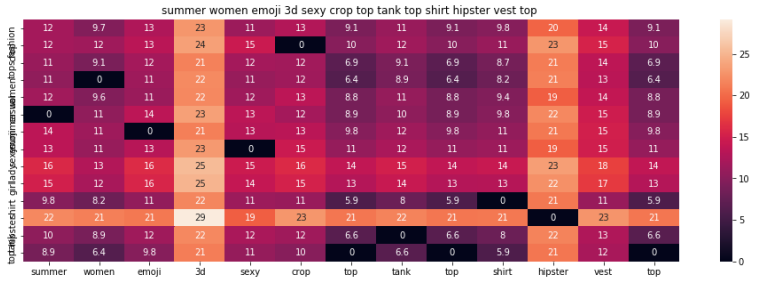
ASIN : B010V39146
Brand : Doxi Supermall
euclidean distance from input : 1.0837464332580566
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3AB5O	Doxi-Supermall	White



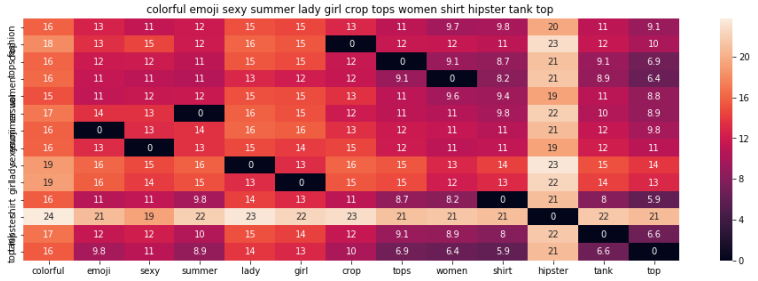
ASIN : B010V3AB5O
Brand : Doxi Supermall
euclidean distance from input : 1.3135900497436523
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3E5EC	Doxi-Supermall	White



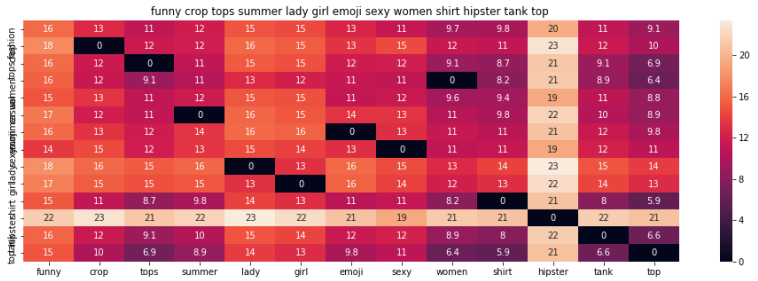
ASIN : B010V3E5EC
Brand : Doxi Supermall
euclidean distance from input : 1.3397351264953614
==
==

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3BQZS	Doxi-Supermall	Multicoloured



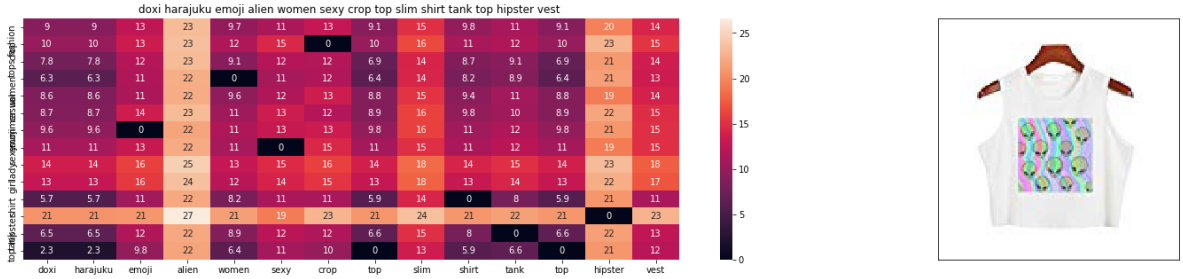
ASIN : B010V3BQZS
Brand : Doxi Supermall
euclidean distance from input : 1.3507432939428952
==
==

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3C116	Doxi-Supermall	Black



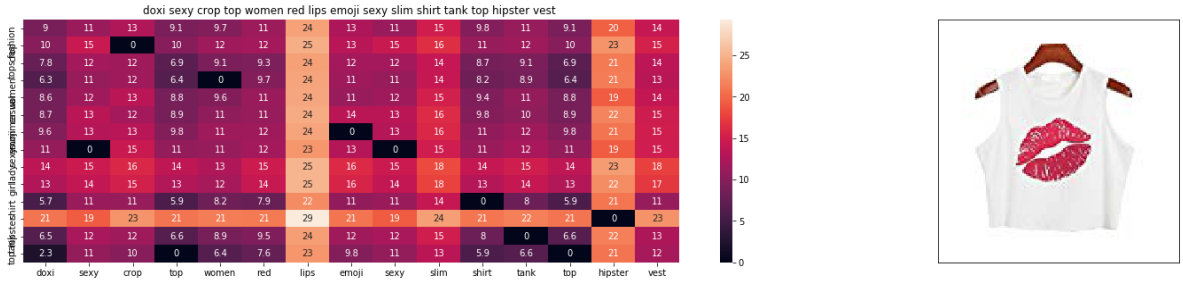
ASIN : B010V3C116
Brand : Doxi Supermall
euclidean distance from input : 1.3836309911627438
==
==

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010TKXAI4	Doxi-Supermall	White



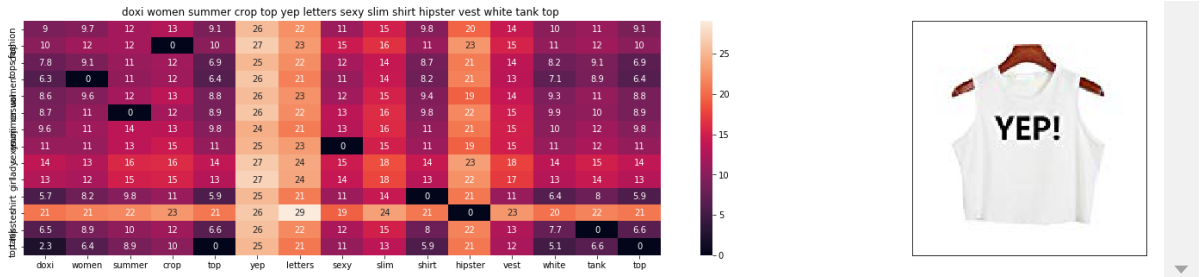
ASIN : B010TKXAI4
Brand : Doxi Supermall
euclidean distance from input : 1.3876009941101075
==
==

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010TKXEHG	Doxi-Supermall	White



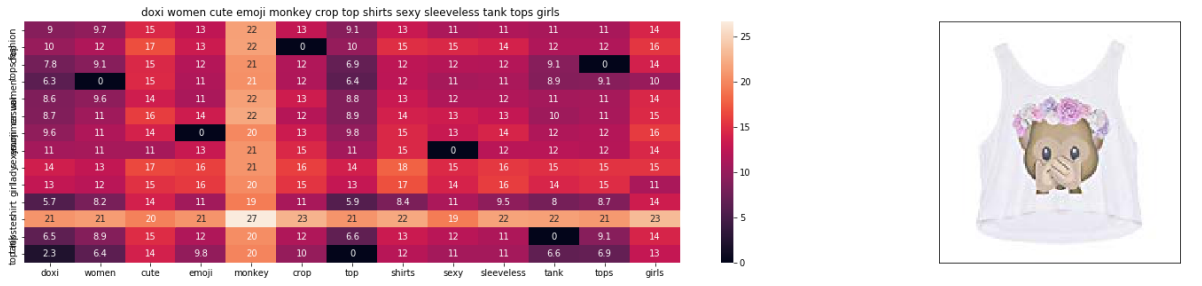
ASIN : B010TKXEHG
Brand : Doxi Supermall
euclidean distance from input : 1.4269559860229493
==
==

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3487Q	Doxi-Supermall	White



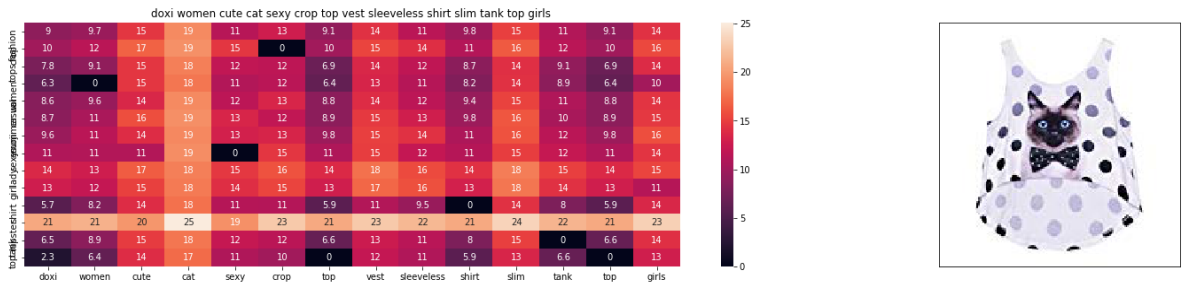
ASIN : B010V3487Q
Brand : Doxi Supermall
euclidean distance from input : 1.468186855316162
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01LF90QTO	Doxi-Supermall	White



ASIN : B01LF90QTO
Brand : Doxi Supermall
euclidean distance from input : 1.4761533737182617
==========
==========

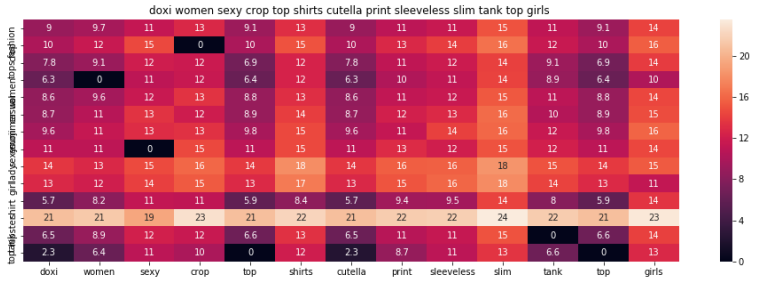
Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01LF90ROI	Doxi-Supermall	White



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

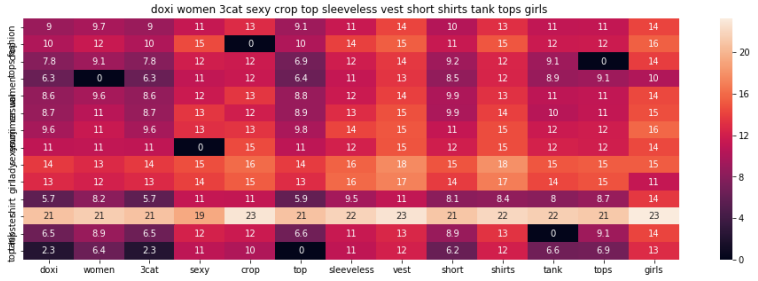
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01LF90RKC	Doxi-Supermall	White



ASIN : B01LF90RKC
Brand : Doxi Supermall
euclidean distance from input : 1.5058938026428224
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01LY4GQY0	Doxi-Supermall	White

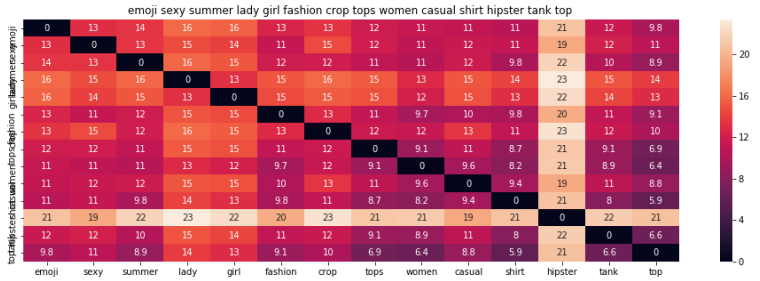


ASIN : B01LY4GQY0
Brand : Doxi Supermall
euclidean distance from input : 1.5098180770874023
==========
==========

In [55]:

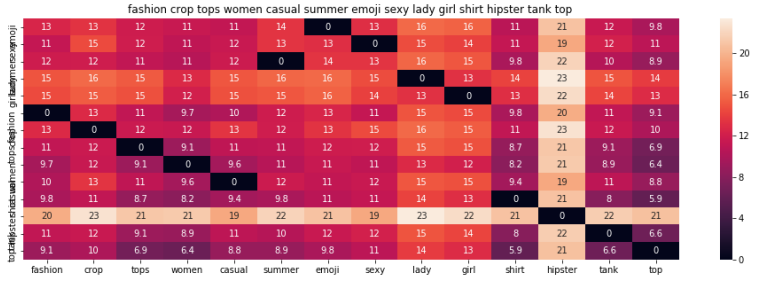
```
# brand and color weight =50
# title vector weight = 5

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

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010V3BLWQ	Doxi-Supermall	White



ASIN : B010V3BLWQ
Brand : Doxi Supermall
euclidean distance from input : 0.0
==========
==========

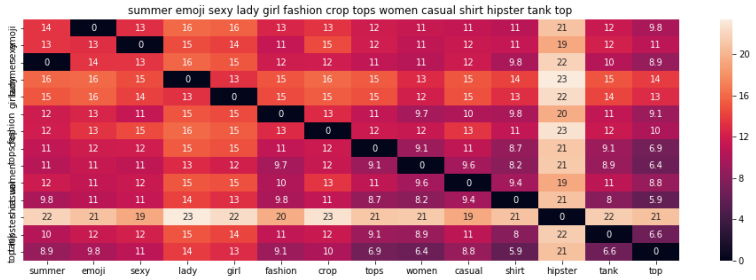
Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010V3B44G	Doxi-Supermall	White



ASIN : B010V3B44G
Brand : Doxi Supermall
euclidean distance from input : 0.0
==========
==========

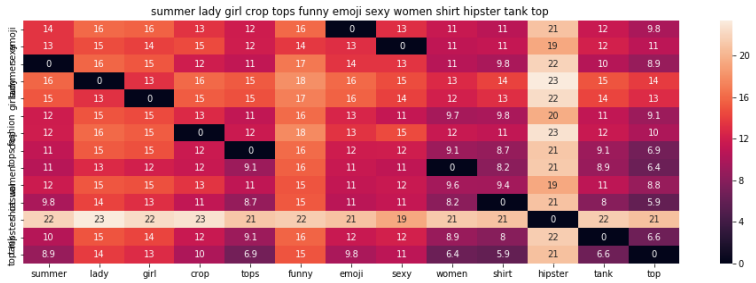
Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White

B010V3BDII	Doxi-Supermall	White



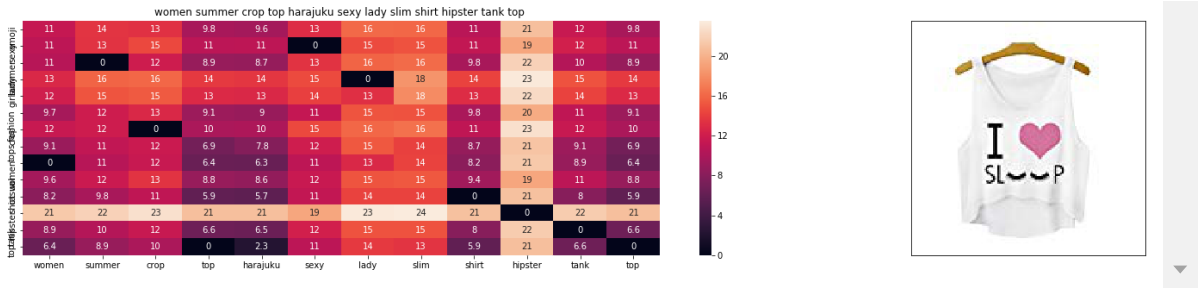
ASIN : B010V3BDII
Brand : Doxi Supermall
euclidean distance from input : 0.0
==========
==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010V3BVMQ	Doxi-Supermall	White



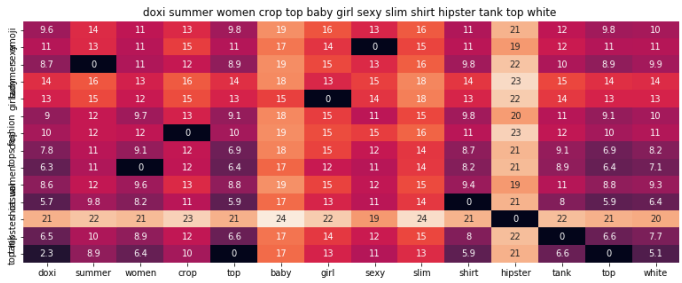
ASIN : B010V3BVMQ
Brand : Doxi Supermall
euclidean distance from input : 0.12300440181385387
==========
==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010V3EDEE	Doxi-Supermall	White



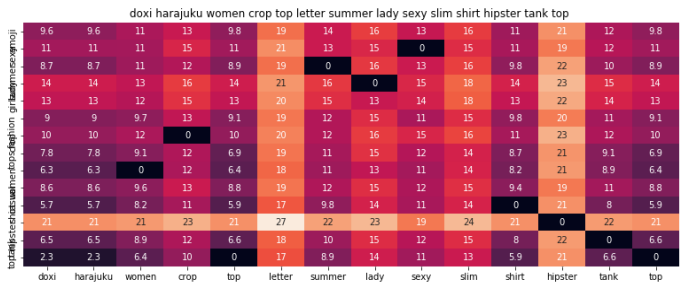
ASIN : B010V3EDEE
Brand : Doxi Supermall
euclidean distance from input : 0.15849236575039952
==========
==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010V3A23U	Doxi-Supermall	White



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

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010V380LQ	Doxi-Supermall	White

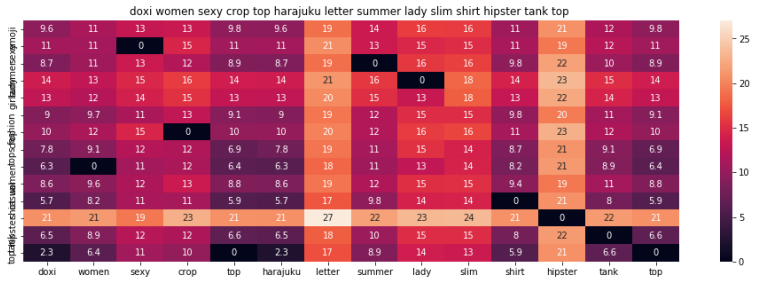


ASIN : B010V380LQ
Brand : Doxi Supermall
euclidean distance from input : 0.1970447540283203

==========

==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010V39146	Doxi-Supermall	White

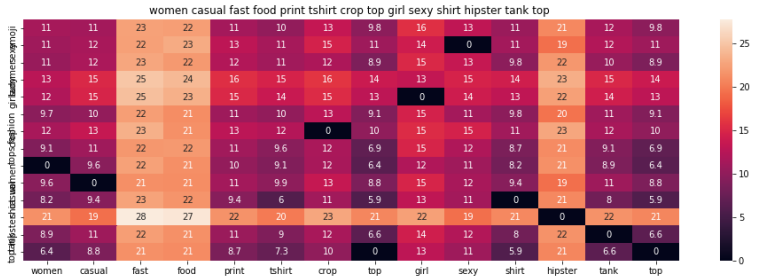


ASIN : B010V39146
Brand : Doxi Supermall
euclidean distance from input : 0.1970448060469194

==========

==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010V3AB5O	Doxi-Supermall	White

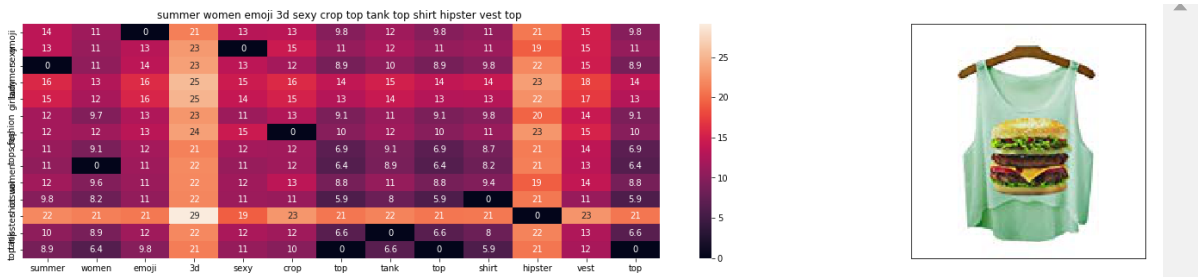


ASIN : B010V3AB5O
Brand : Doxi Supermall
euclidean distance from input : 0.23883455449884589

==========

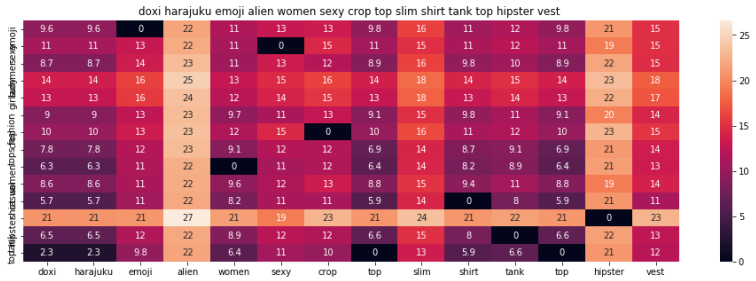
==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010V3E5EC	Doxi-Supermall	White



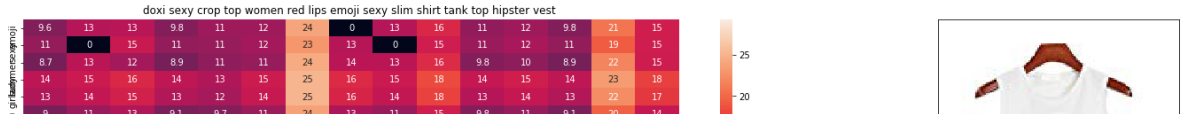
ASIN : B010V3E5EC
Brand : Doxi Supermall
euclidean distance from input : 0.24358820481733842
==========
==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010TKXAI4	Doxi-Supermall	White



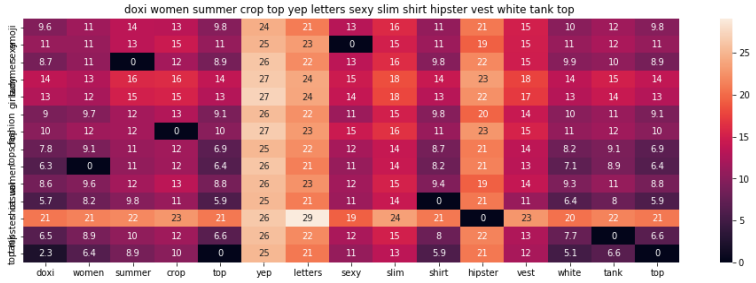
ASIN : B010TKXAI4
Brand : Doxi Supermall
euclidean distance from input : 0.25229108983820137
==========
==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010TKXEHG	Doxi-Supermall	White



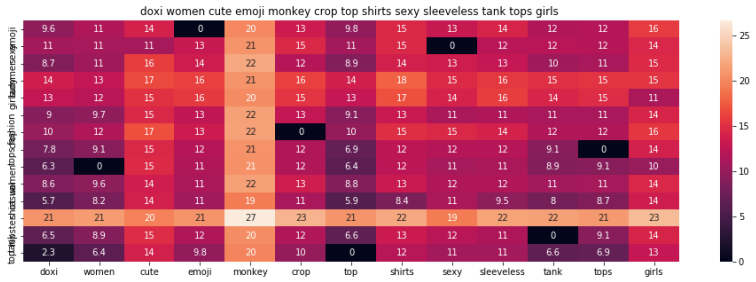
ASIN : B010TKXEHG
Brand : Doxi Supermall
euclidean distance from input : 0.25944654291326347
==========
==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010V3487Q	Doxi-Supermall	White



ASIN : B010V3487Q
Brand : Doxi Supermall
euclidean distance from input : 0.26694306460293854
==========
==========

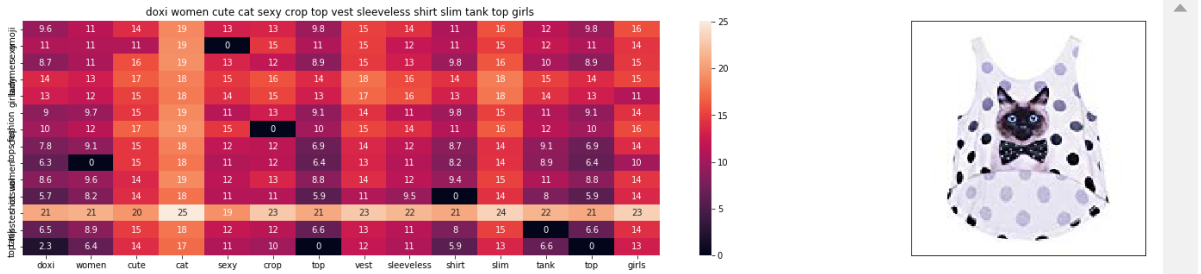
Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B01LF90QTO	Doxi-Supermall	White



ASIN : B01LF90QTO
Brand : Doxi Supermall
euclidean distance from input : 0.26839152249422943
==========
==========

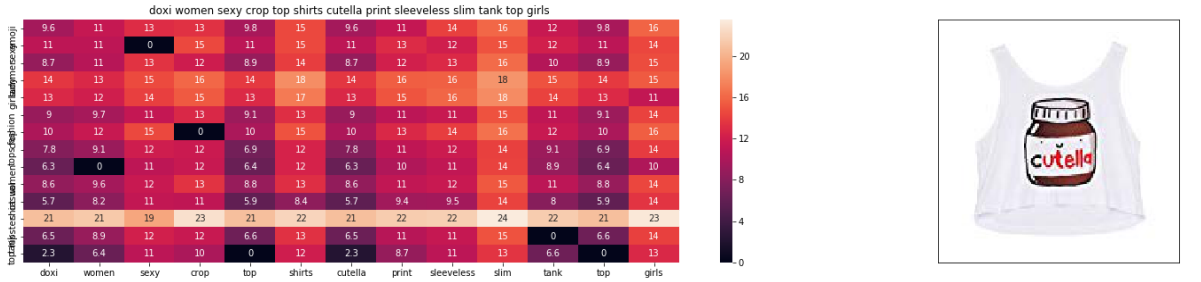
Asin	Brand	Color

B010V3BLWQ	Doxi-Supermall	White
B01LF90ROI	Doxi-Supermall	White



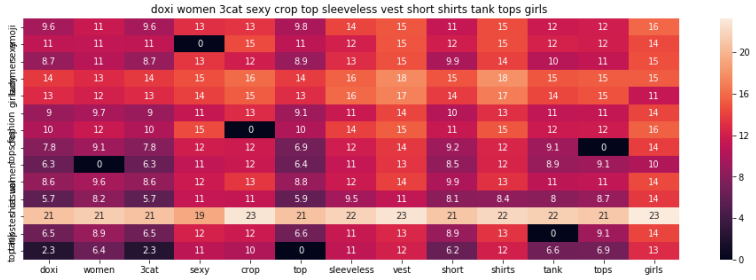
ASIN : B01LF90ROI
Brand : Doxi Supermall
euclidean distance from input : 0.2694863406094638
==========
==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B01LF90RKC	Doxi-Supermall	White



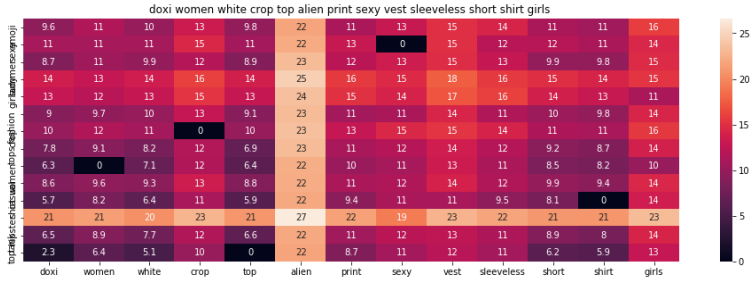
ASIN : B01LF90RKC
Brand : Doxi Supermall
euclidean distance from input : 0.27379887320778584
==========
==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B01LY4GQY0	Doxi-Supermall	White



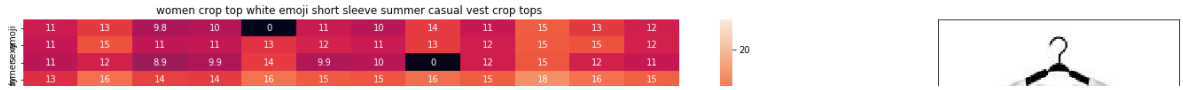
ASIN : B01LY4GQY0
Brand : Doxi Supermall
euclidean distance from input : 0.27451237765225495
==========
==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B01LF90SCY	Doxi-Supermall	White



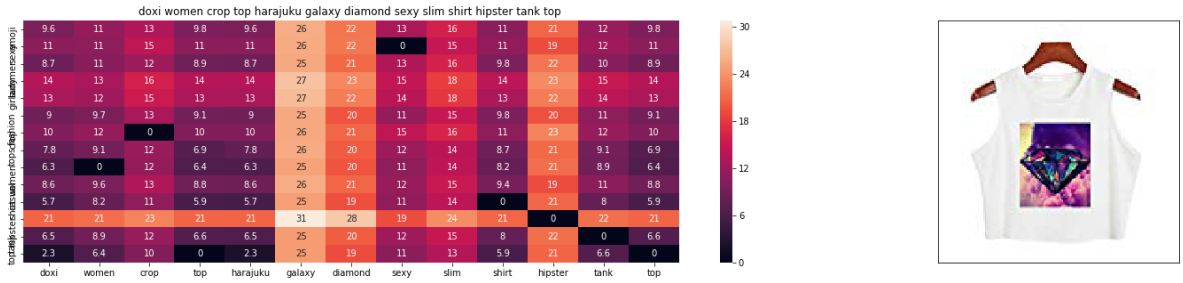
ASIN : B01LF90SCY
Brand : Doxi Supermall
euclidean distance from input : 0.28159148476340556
==========
==========

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B0124E7MHS	Doxi-Supermall	White



ASIN : B0124E7MHS
Brand : Doxi Supermall
euclidean distance from input : 0.28713430924849076
==
==

Asin	Brand	Color
B010V3BLWQ	Doxi-Supermall	White
B010V39EUM	Doxi-Supermall	White



ASIN : B010V39EUM
Brand : Doxi Supermall
euclidean distance from input : 0.3081340443004261
==
==

[10.2] Keras and Tensorflow to extract features

In [56]:

```
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 [57]:

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



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

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

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

# each image is converted into 25088 length dense-vector


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

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


def save_bottlebeck_features():

    #Function to compute VGG-16 CNN for image feature extraction.

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

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

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

    bottleneck_features_train = model.predict_generator(generator, nb_train_samples // batc
    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[57]:

```
"\n# dimensions of our images.\nimg_width, img_height = 224, 224\n\ntop_mode
l_weights_path = 'bottleneck_fc_model.h5'\ntrain_data_dir = 'images2/'\nnb_t
rain_samples = 16042\nepochs = 50\nbatch_size = 1\n\n\ndef save_bottlebeck_f
eatures():\n    \n    #Function to compute VGG-16 CNN for image feature extr
action.\n    \n    asins = []\n    datagen = ImageDataGenerator(rescale=1. /
255)\n    \n    # build the VGG16 network\n    model = applications.VGG16(in
clude_top=False, weights='imagenet')\n    generator = datagen.flow_from_dire
ctory(\n        train_data_dir,\n        target_size=(img_width, img_heigh
t),\n        batch_size=batch_size,\n        class_mode=None,\n        shuff
le=False)\n\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_fe
atures_train.reshape((16042,25088))\n    \n    np.save(open('16k_data_cnn_fe
atures.npy', 'wb'), bottleneck_features_train)\n    np.save(open('16k_data_c
nn_feature_asins.npy', 'wb'), np.array(asins))\n    \n\nsave_bottlebeck_feat
ures()\n\n"
```

[10.3] Visual features based product similarity.

In [58]:

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

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


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


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

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

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

get_similar_products_cnn(12566, 20)
```


```
Product Title:  fashion crop tops women casual summer emoji sexy lady girl s
hirt hipster tank top
Euclidean Distance from input image: 0.054126587
Amazon Url: www.amzon.com/dp/B010V3B44G
```


```
Product Title:  chic women harajuku fashion funny emoji sexy crop tops sleev
eless vest shirt
```

Euclidean Distance from input image: 18.585712
Amazon Url: www.amzon.com/dp/B011RCJNL6

Product Title: women quotes boys print white sleeveless crop top
Euclidean Distance from input image: 22.045673
Amazon Url: www.amzon.com/dp/B0748CKWF3

Product Title: women forever young print white sleeveless crop top
Euclidean Distance from input image: 22.32101
Amazon Url: www.amzon.com/dp/B0748CDWPH

Product Title: kingde self confident stretch vest tshirt suspendersbqn46
Euclidean Distance from input image: 22.724596
Amazon Url: www.amzon.com/dp/B015H2R3SC

Product Title: women three wise monkeys emoji print sleeveless crop top
Euclidean Distance from input image: 24.45252
Amazon Url: www.amzon.com/dp/B074VPC98H

Product Title: women infinity books print white sleeveless crop top
Euclidean Distance from input image: 25.06742
Amazon Url: www.amzon.com/dp/B074TYKHPL

Product Title: ladies crop top emoji women never grow shirt vest style tank tops
Euclidean Distance from input image: 26.78721
Amazon Url: www.amzon.com/dp/B013SPUM7O

Product Title: women fashion giant pink donut printed white sleeveless crop top
Euclidean Distance from input image: 28.364958
Amazon Url: www.amzon.com/dp/B017Q4BOCA

Product Title: women keep swimming print sleeveless crop top
Euclidean Distance from input image: 29.022234
Amazon Url: www.amzon.com/dp/B0749CCCY4

Product Title: women summer crop top harajuku sexy lady slim shirt hipster tank top
Euclidean Distance from input image: 29.029299
Amazon Url: www.amzon.com/dp/B010V3EDEE

Product Title: women fashion wanna hug printed white sleeveless crop top
Euclidean Distance from input image: 29.049618
Amazon Url: www.amzon.com/dp/B017R0CDOQ

Product Title: summer emoji sexy lady girl fashion crop tops women casual s hirt hipster tank top
Euclidean Distance from input image: 29.482252
Amazon Url: www.amzon.com/dp/B010V3BDII

Product Title: women fashion love food printed white crop top
Euclidean Distance from input image: 29.615307
Amazon Url: www.amzon.com/dp/B017K0DQZ8

Product Title: kingde slim starbucks coffee small vest tshirt sling stret chbqn53
Euclidean Distance from input image: 29.689983
Amazon Url: www.amzon.com/dp/B015H2QWY8

Product Title: women fashion cute word printed name barbie sleeveless crop top
Euclidean Distance from input image: 29.862267
Amazon Url: www.amzon.com/dp/B01AK8SC10

Product Title: women fashion best friends printed best pattern one sleevele ss crop top
Euclidean Distance from input image: 29.954777
Amazon Url: www.amzon.com/dp/B01GFJOGUE

Product Title: women yabish print white sleeveless crop top
Euclidean Distance from input image: 30.084452
Amazon Url: www.amzon.com/dp/B0748JNFL9

```
Product Title:  women fashion cool quote printed sleeveless crop top
Euclidean Distance from input image: 30.100662
Amazon Url: www.amzon.com/dp/B018RVK1FM
```

```
Product Title:  women pattern 8 cute baby alien print sleeveless crop top
Euclidean Distance from input image: 30.180511
Amazon Url: www.amzon.com/dp/B074BNJM8S
```

In [59]:

```
data.head()
```

Out[59]:

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

assignment

In [60]:

```
bottleneck_features_train = np.load('16k_data_cnn_features.npy')
```

In [61]:

```
bft=bottleneck_features_train[0:16042]
wtw=w2v_title_weight[0:16042]
bf=brand_features[0:16042]
cf=color_features[0:16042]
```

In [62]:

```
assignment_features = hstack((wtw,bf,cf,bft)).tocsr()
```

In [63]:

```
assignment_features.shape
```

Out[63]:

```
(16042, 31118)
```

In [64]:

```
def heat_map_w2v_brand2(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) o
    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) o
    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]],
                [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2], types[doc_id2]]]

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

    # 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

    # 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)



    plt.show()
```

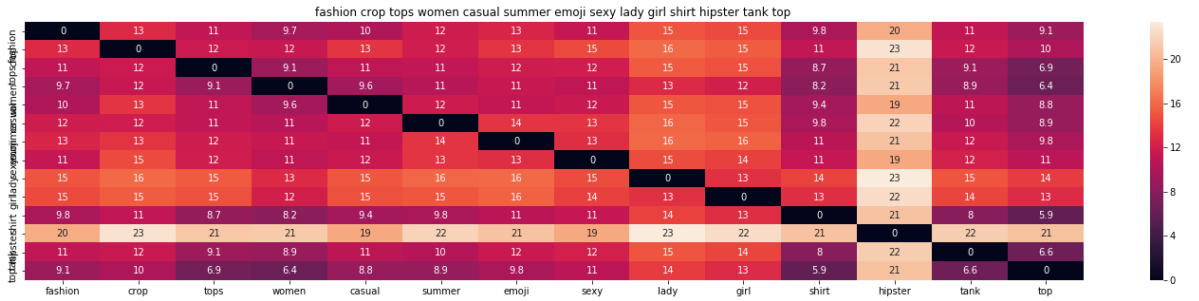
In [65]:

```
def idf_w2v_brand2(doc_id, w12, w22, w32, w42, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand and color feature   wtw,bf,cf,bfts

    # pairwise_dist will store the distance from given input apparel to all remaining appar
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y>
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    idf_w2v_dist2  = pairwise_distances(wtw,wtw[doc_id].reshape(1,-1))
    ex_feat_dist_brand2 = pairwise_distances(bf, bf[doc_id])
    ex_feat_dist_color_2 = pairwise_distances(cf,cf[doc_id])
    ex_feat_dist_bootleneck_2 = pairwise_distances(bft, bft[doc_id].reshape(1,-1))

    pairwise_dist2   = (w12*idf_w2v_dist2 +  w22*ex_feat_dist_brand2 +  w32*ex_feat_dist_cc

    # np.argsort will return indices of 9 smallest distances
    indices = np.argsort(pairwise_dist2.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist2.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)):

        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))

        heat_map_w2v_brand2(data['title'].loc[df_indices[0]],data['title'].loc[df_indices[i
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print('Brand :',data['brand'].loc[df_indices[i]])
        print('euclidean distance from input :', pdists[i])
        print('='*125)
```

In [67]:

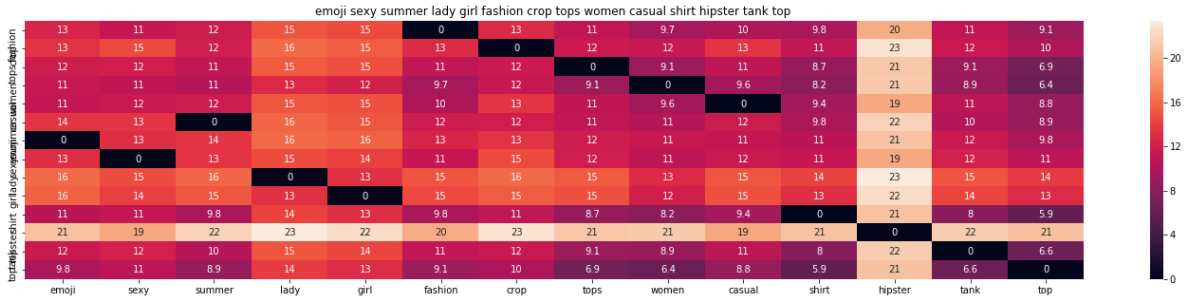
```
idf_w2v_brand2(12566, 50, 5, 5, 5 ,10)
```

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3B44G	Doxi-Supermall	White



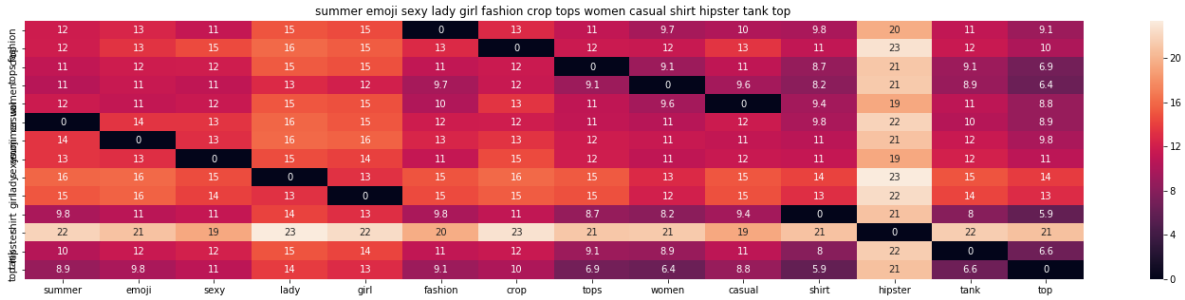

```
ASIN : B010V3B44G
Brand : Doxi Supermall
euclidean distance from input : 0.0033995518317589393
================================================================================
================================================
```

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3BLWQ	Doxi-Supermall	White



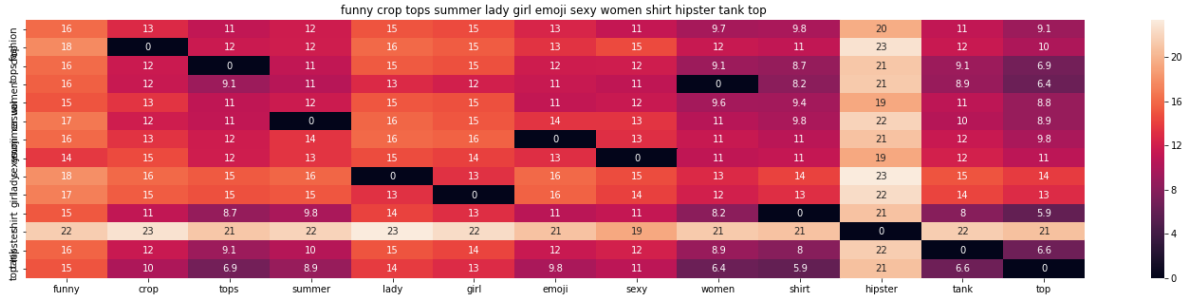
ASIN : B010V3BLWQ
Brand : Doxi Supermall
euclidean distance from input : 4.360835618239182
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3BDII	Doxi-Supermall	White



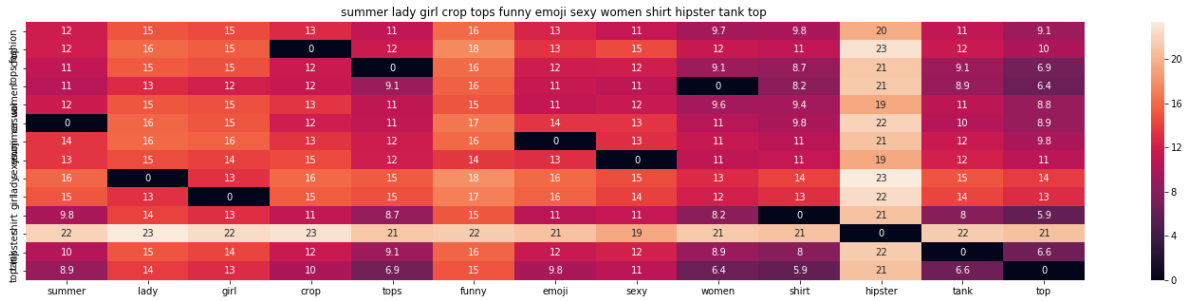
ASIN : B010V3BDII
Brand : Doxi Supermall
euclidean distance from input : 4.927261117788461
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3C116	Doxi-Supermall	Black



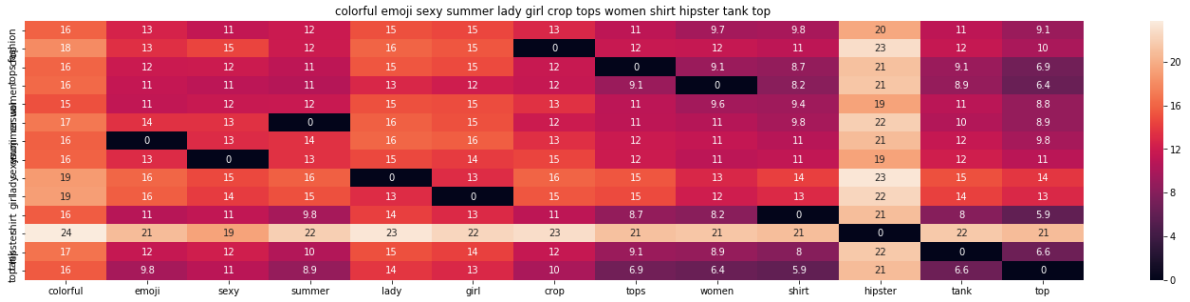
ASIN : B010V3C116
Brand : Doxi Supermall
euclidean distance from input : 5.086525198157306
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3BVMQ	Doxi-Supermall	White



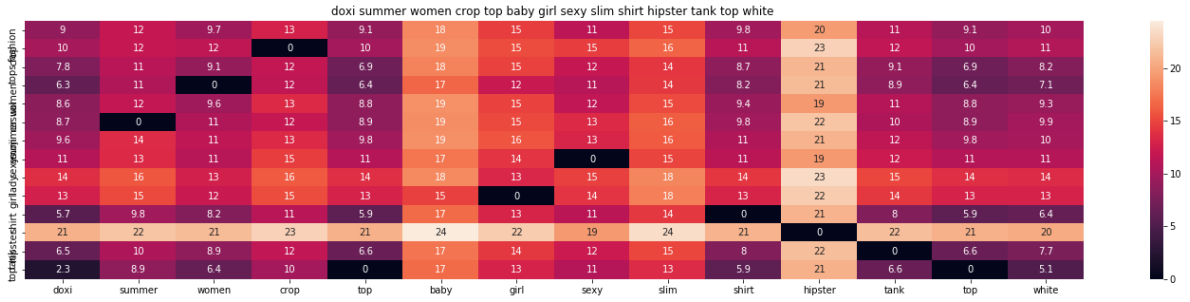
ASIN : B010V3BVMQ
Brand : Doxi Supermall
euclidean distance from input : 5.103096595177283
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3BQZS	Doxi-Supermall	Multicoloured



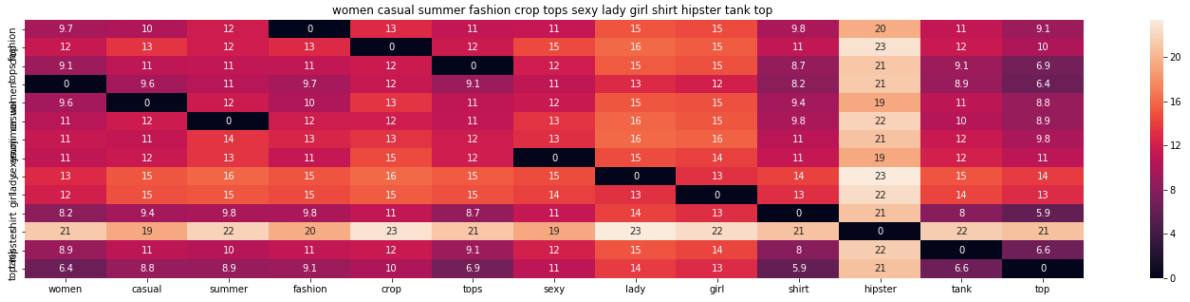
ASIN : B010V3BQZS
Brand : Doxi Supermall
euclidean distance from input : 5.135792893657606
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3A23U	Doxi-Supermall	White



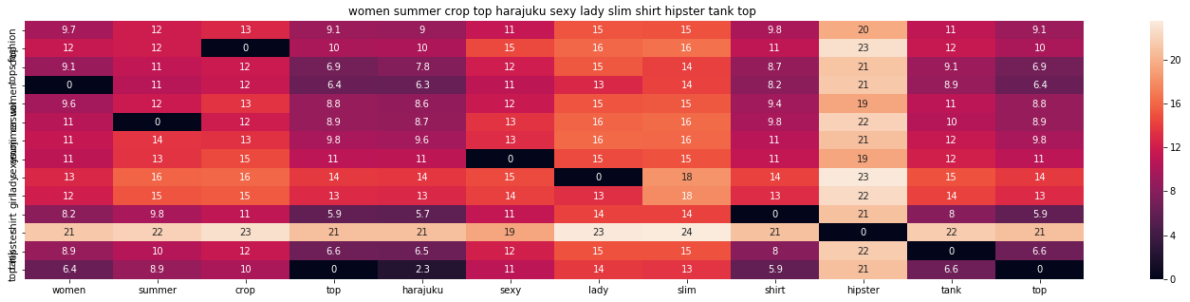
ASIN : B010V3A23U
Brand : Doxi Supermall
euclidean distance from input : 5.568003258338341
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3AYSS	Doxi-Supermall	Multicoloured



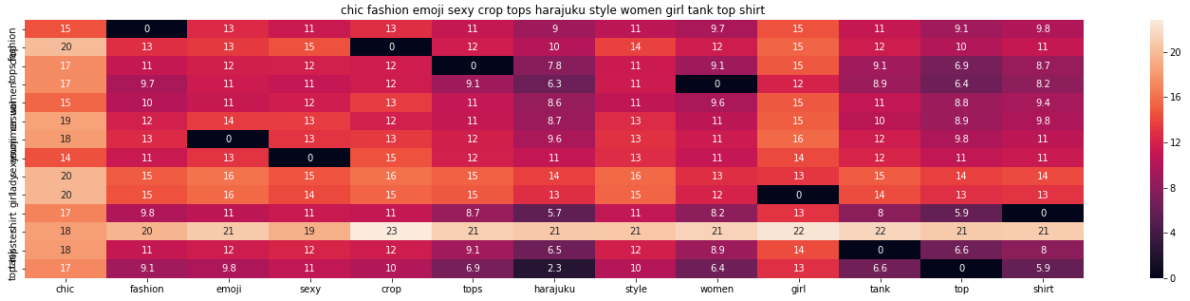
ASIN : B010V3AYSS
Brand : Doxi Supermall
euclidean distance from input : 5.569250429621398
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3EDEE	Doxi-Supermall	White



ASIN : B010V3EDEE
Brand : Doxi Supermall
euclidean distance from input : 5.588982567420373
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B011RCJPR8	Chiclook-Cool	White

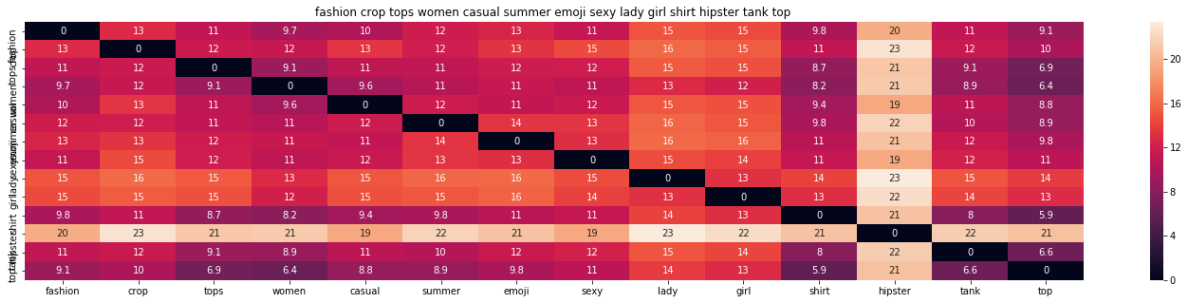


ASIN : B011RCJPR8
Brand : Chiclook Cool
euclidean distance from input : 5.775457176795372
==========
==========

In [68]:

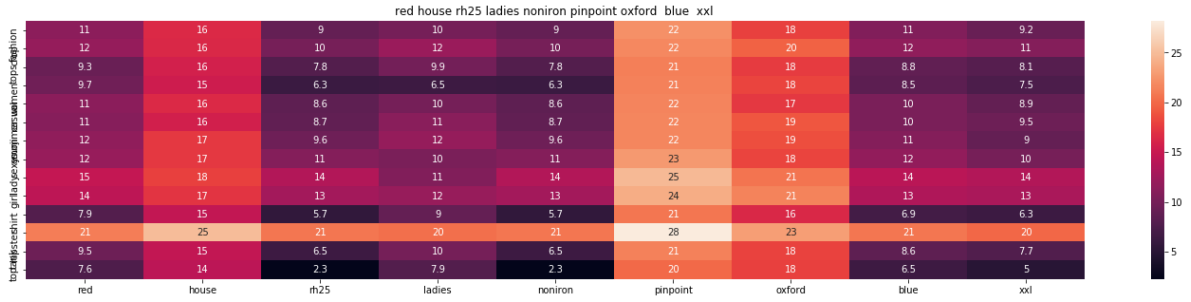
```
idf_w2v_brand2(12566, 10, 50, 10,10 ,10)
```

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3B44G	Doxi-Supermall	White



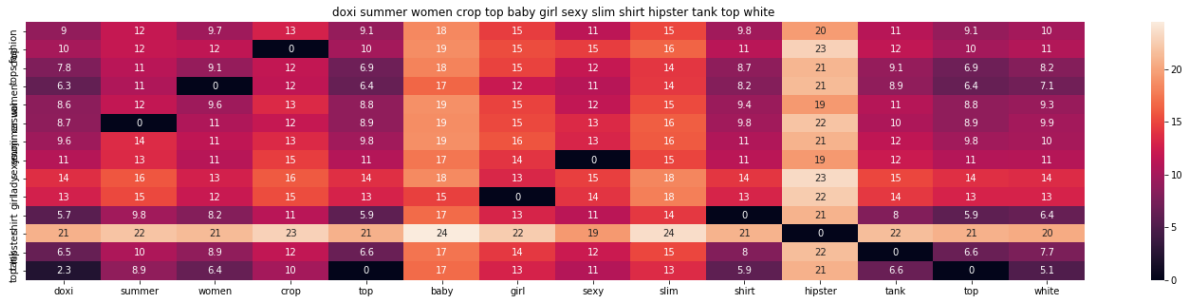
ASIN : B010V3B44G
Brand : Doxi Supermall
euclidean distance from input : 0.005524271726608276
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B00884HIEG	Red-House	Blue



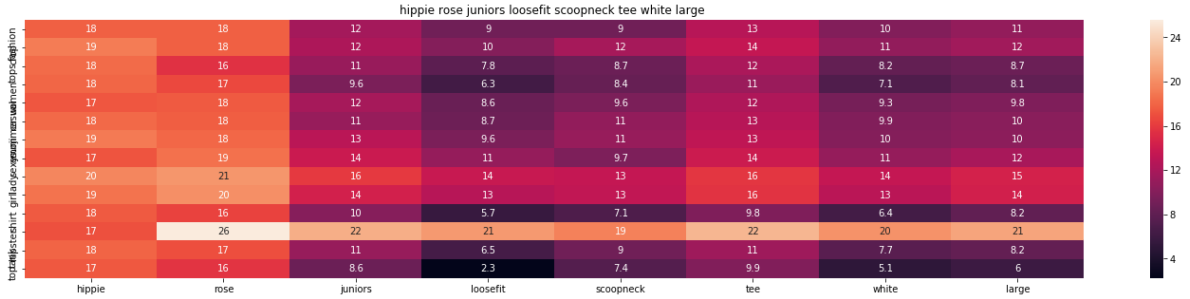
ASIN : B00884HIEG
Brand : Red House
euclidean distance from input : 6.625225400969855
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3A23U	Doxi-Supermall	White



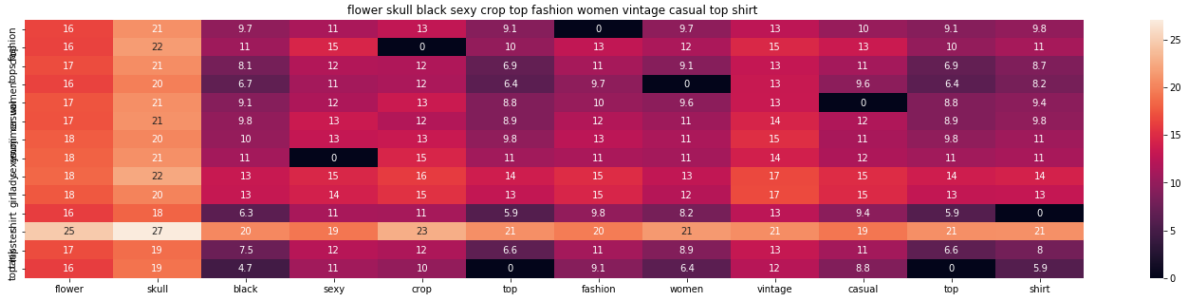
ASIN : B010V3A23U
Brand : Doxi Supermall
euclidean distance from input : 6.658090281486511
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01JNY5C2Y	Hippie-Rose	White



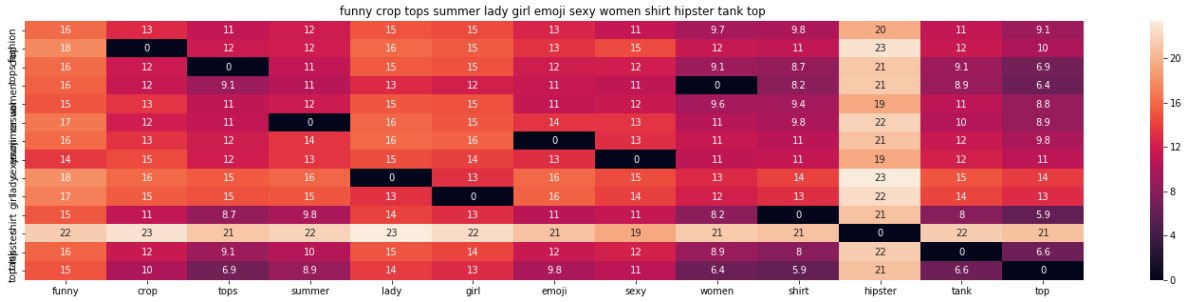
ASIN : B01JNY5C2Y
Brand : Hippie Rose
euclidean distance from input : 6.700562906265259
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B0124E8862	Doxi-Supermall	Black



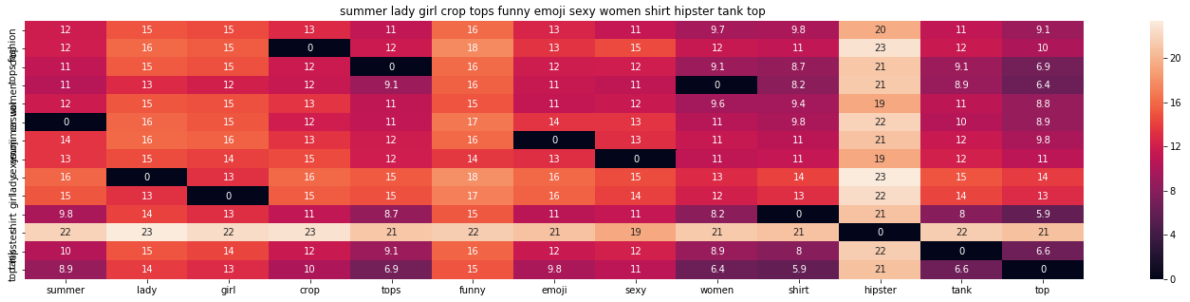
ASIN : B0124E8862
Brand : Doxi Supermall
euclidean distance from input : 6.730817914054266
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3C116	Doxi-Supermall	Black



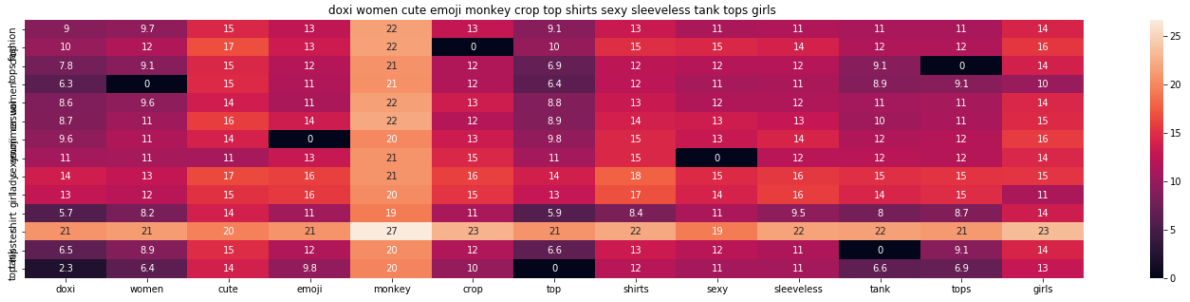
ASIN : B010V3C116
Brand : Doxi Supermall
euclidean distance from input : 6.743423998401037
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3BVMQ	Doxi-Supermall	White



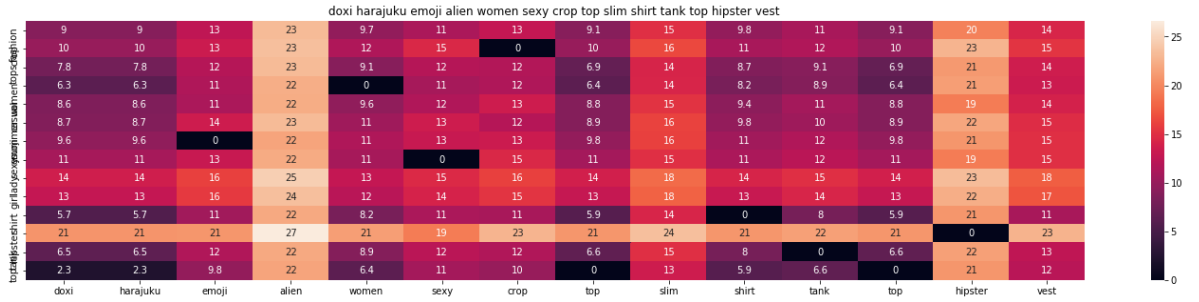
ASIN : B010V3BVMQ
Brand : Doxi Supermall
euclidean distance from input : 6.770352518558502
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01LF90QTO	Doxi-Supermall	White



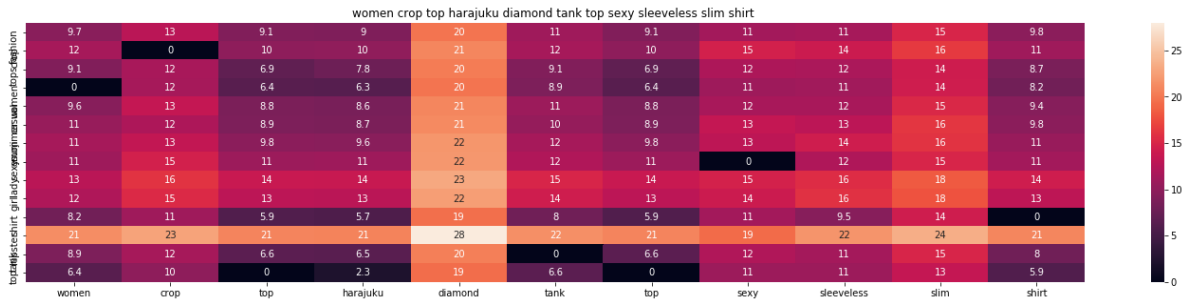
ASIN : B01LF90QTO
Brand : Doxi Supermall
euclidean distance from input : 6.791046094894409
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010TKXAI4	Doxi-Supermall	White



ASIN : B010TKXAI4
Brand : Doxi Supermall
euclidean distance from input : 6.836821818351746
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V34452	Doxi-Supermall	White



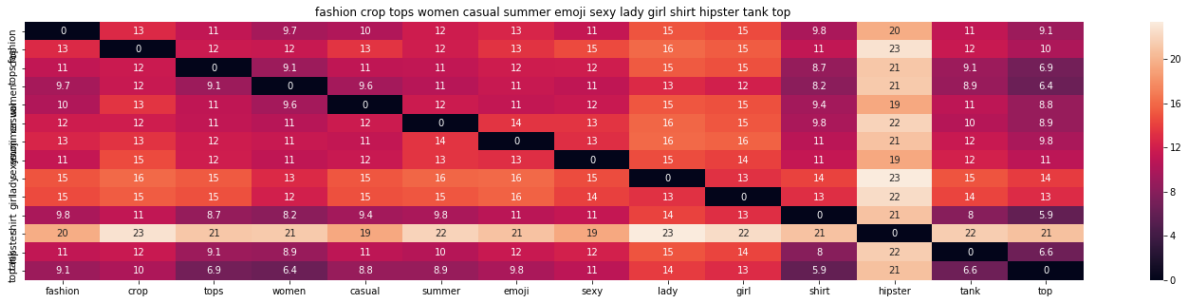
ASIN : B010V34452
Brand : Doxi Supermall
euclidean distance from input : 6.849838829040527

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

In [69]:

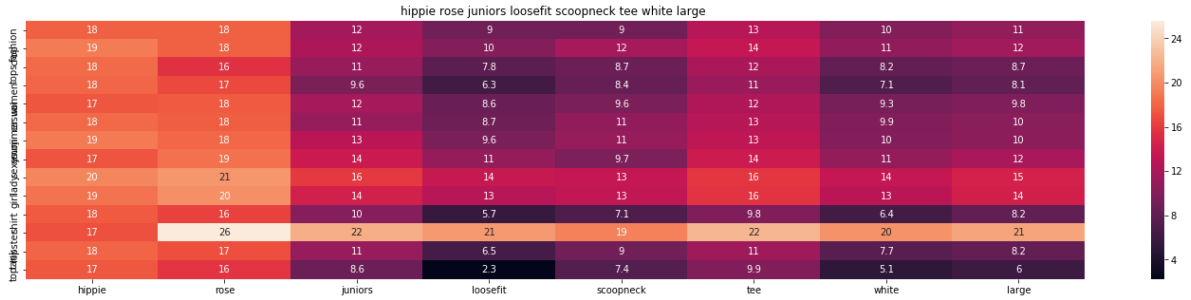
```
idf_w2v_brand2(12566, 10, 10, 50,10 ,10)
```

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3B44G	Doxi-Supermall	White



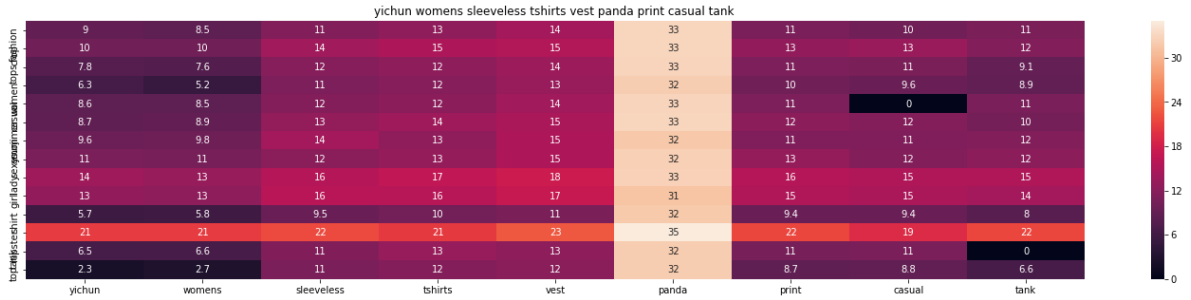
ASIN : B010V3B44G
Brand : Doxi Supermall
euclidean distance from input : 0.005524271726608276
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01JNY5C2Y	Hippie-Rose	White



ASIN : B01JNY5C2Y
Brand : Hippie Rose
euclidean distance from input : 5.700562906265259
==========
==========

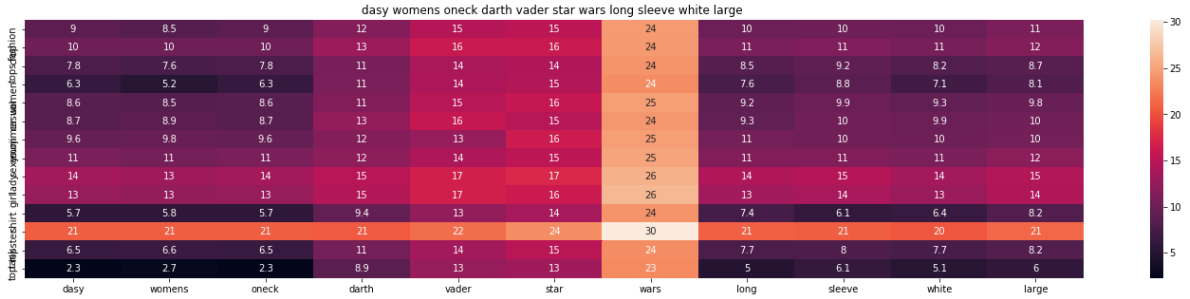
Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B00VYJXDMI	YICHUN	White



ASIN : B00VYJXDMI
Brand : YICHUN
euclidean distance from input : 6.061461604547183
==========
==========

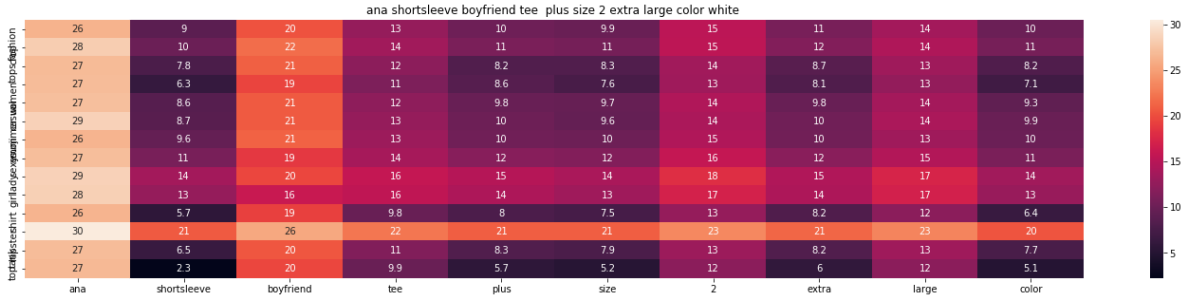
Asin	Brand	Color

B010V3B44G	Doxi-Supermall	White
B017YUFGQG	Dasy	White



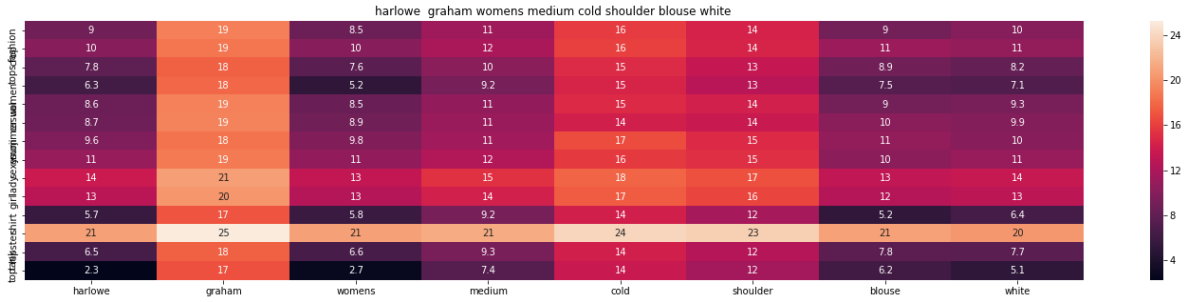
ASIN : B017YUFGQG
Brand : Dasy
euclidean distance from input : 6.071174968194645
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01MSN4CTE	A.N.A-Shirt	White



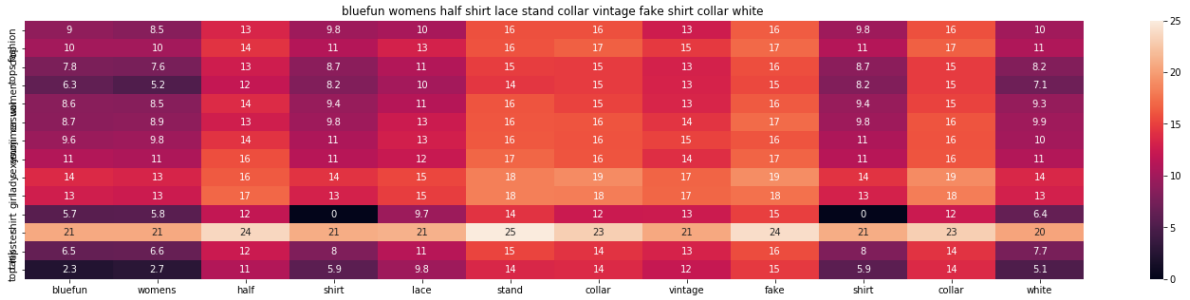
ASIN : B01MSN4CTE
Brand : A.N.A Shirt
euclidean distance from input : 6.132980311822575
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B07354CFVZ	Harlowe-&-Graham	White



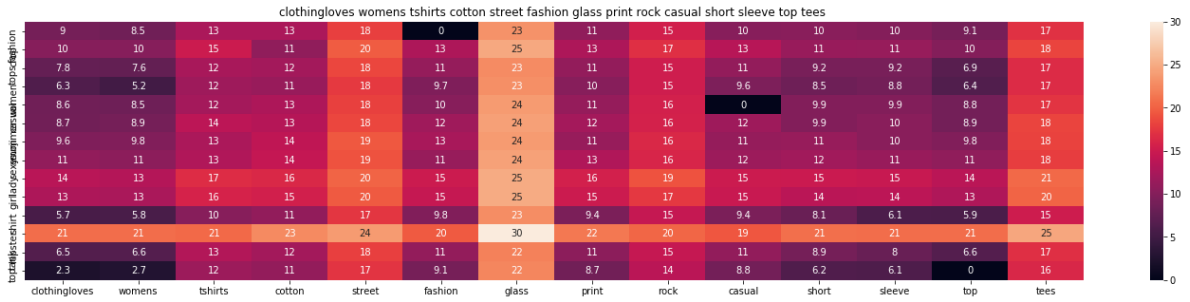
ASIN : B07354CFVZ
Brand : Harlowe & Graham
euclidean distance from input : 6.134935808181763
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01KLCOVZQ	Bluefun	White



ASIN : B01KLCOVZQ
Brand : Bluefun
euclidean distance from input : 6.24695593209235
==
==

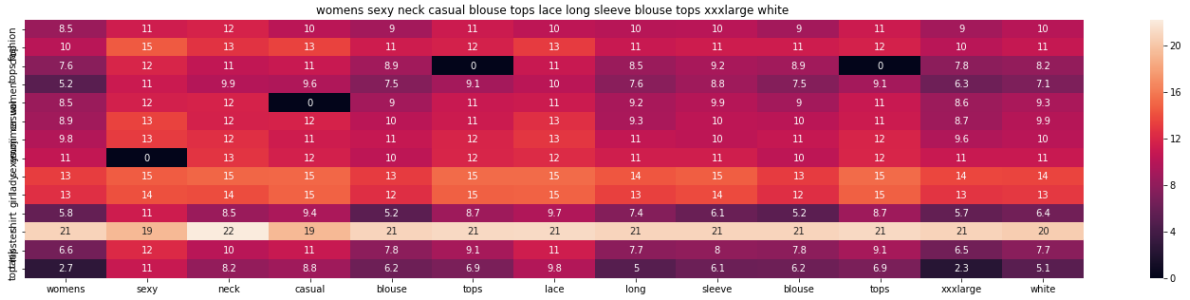
Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B074KD6ZCP	ClothingLoves	White



ASIN : B074KD6ZCP
Brand : ClothingLoves
euclidean distance from input : 6.288050807428044
==
==

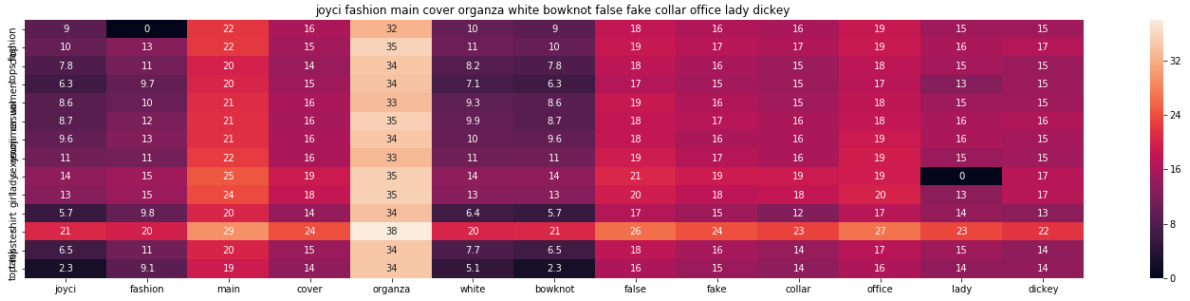
Asin	Brand	Color

B010V3B44G	Doxi-Supermall	White
B06XSZH7SY	Kedera	White



ASIN : B06XSZH7SY
Brand : Kedera
euclidean distance from input : 6.289846242379825
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B019TP6P14	Joyci	White

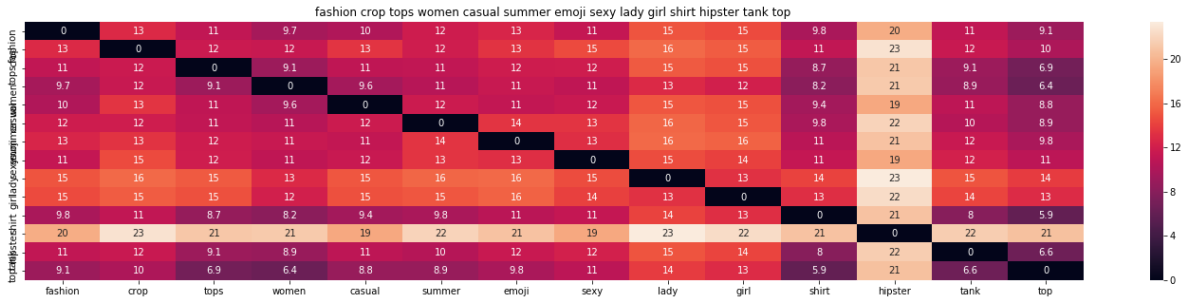


ASIN : B019TP6P14
Brand : Joyci
euclidean distance from input : 6.324158872079532
==========
==========

In [70]:

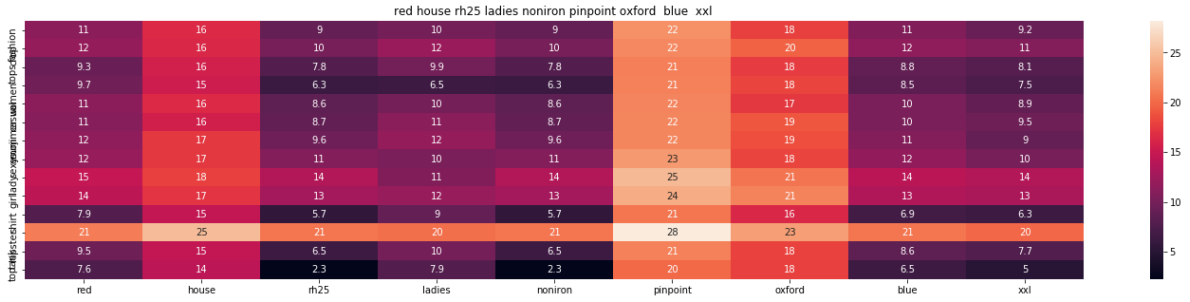
```
idf_w2v_brand2(12566, 10, 10, 10,50 ,10)
```

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B010V3B44G	Doxi-Supermall	White



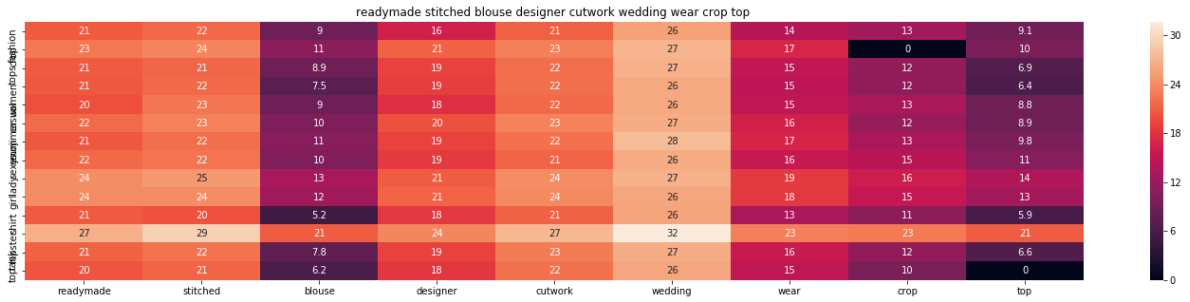

```
ASIN : B010V3B44G
Brand : Doxi Supermall
euclidean distance from input : 0.027621358633041382
================================================================================
================================================
```

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B00884HIEG	Red-House	Blue



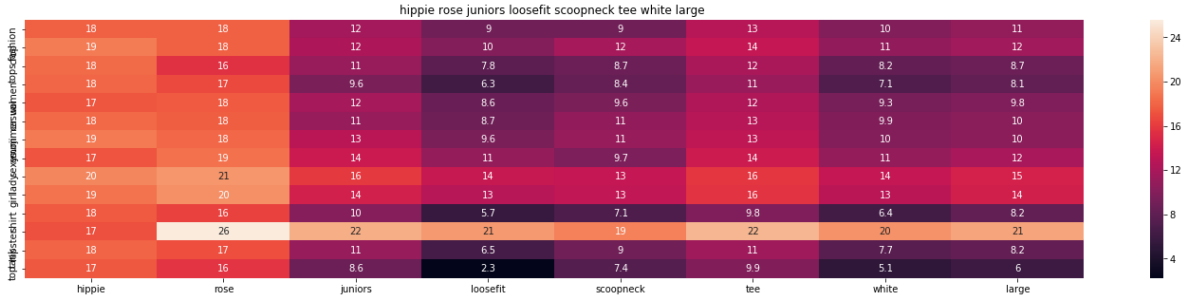
ASIN : B00884HIEG
Brand : Red House
euclidean distance from input : 24.219125318572395
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01908VPR4	Odishabazaar	Gold



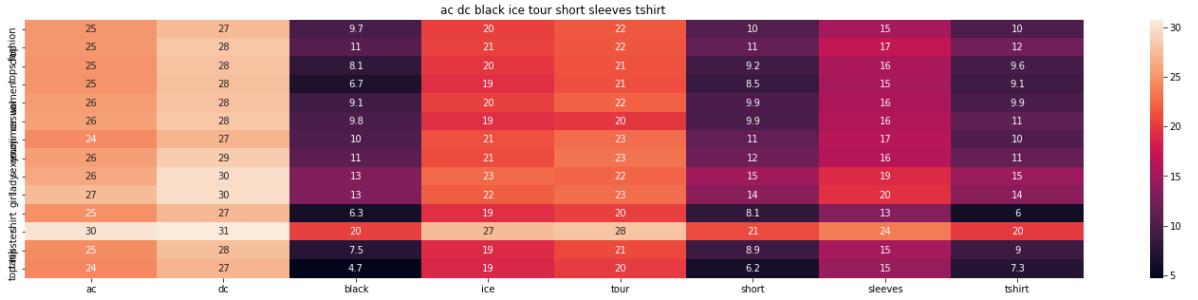
ASIN : B01908VPR4
Brand : Odishabazaar
euclidean distance from input : 25.23986002301887
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01JNY5C2Y	Hippie-Rose	White



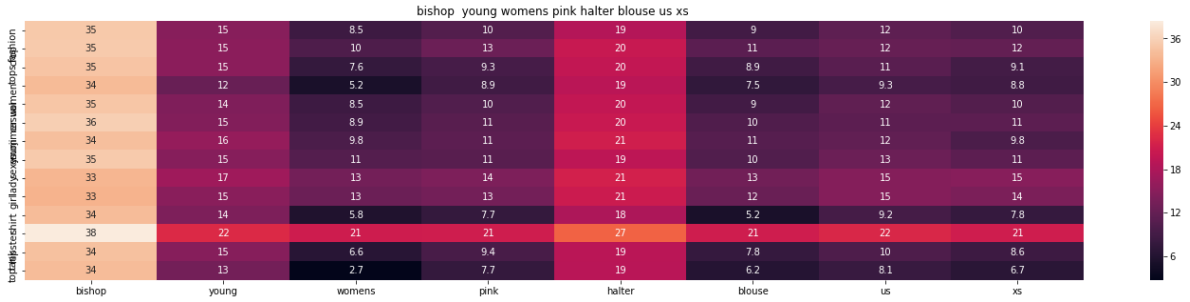
ASIN : B01JNY5C2Y
Brand : Hippie Rose
euclidean distance from input : 25.495816659927367
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01GR2TVWM	Naviiro	Black



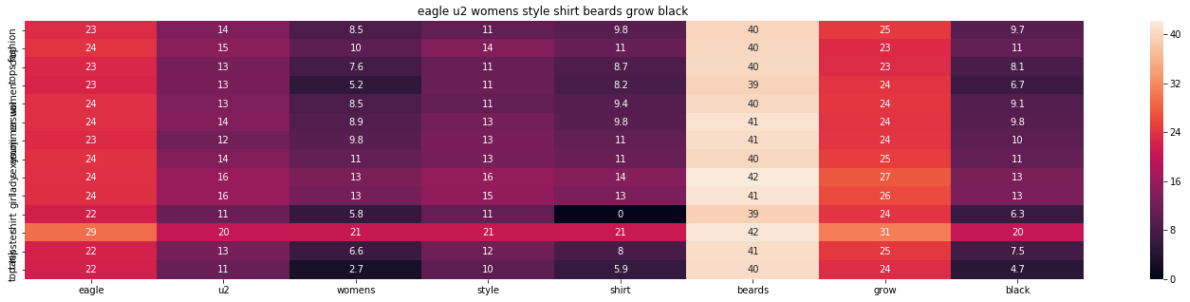
ASIN : B01GR2TVWM
Brand : Naviiro
euclidean distance from input : 25.921908057686842
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B074Q6V49F	Bishop-&-Young	Pink



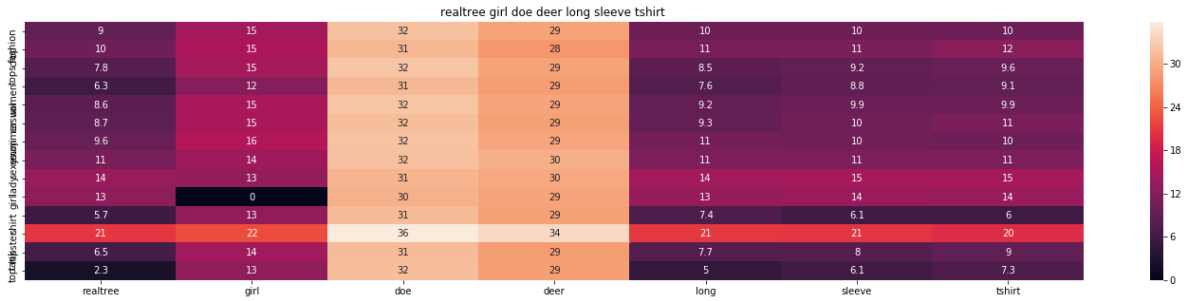
ASIN : B074Q6V49F
Brand : Bishop & Young
euclidean distance from input : 25.95233192448365
==
==

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B01N8S01JW	Eagle	Black



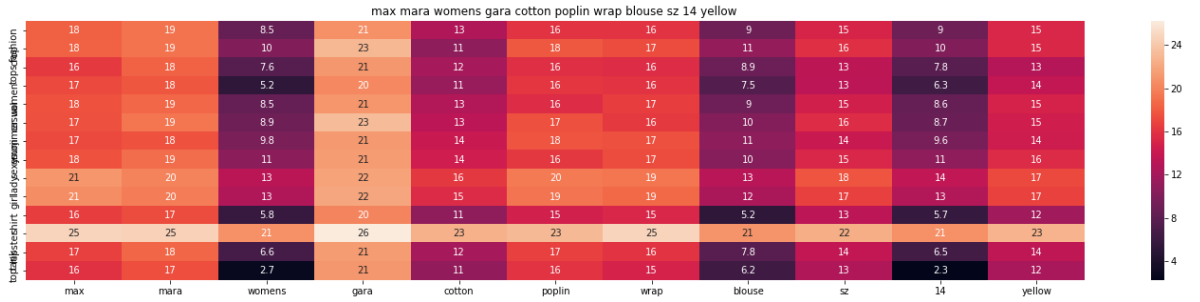
ASIN : B01N8S01JW
Brand : Eagle
euclidean distance from input : 25.998792422768627
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B0735SDJRX	Realtree-Girl	Black



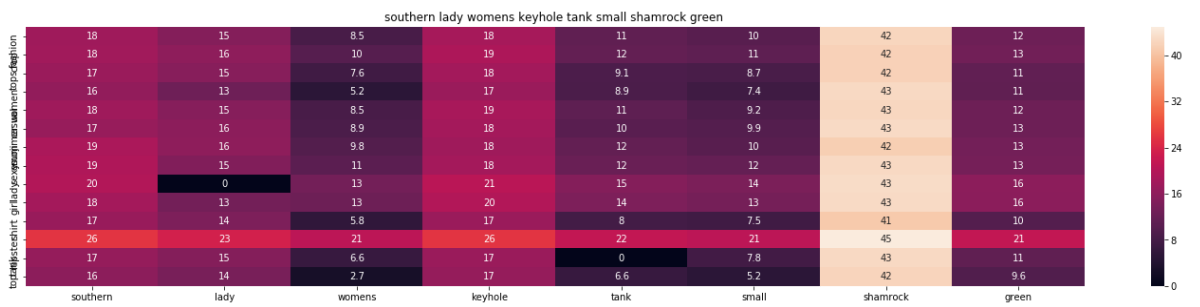
ASIN : B0735SDJRX
Brand : Realtree Girl
euclidean distance from input : 26.040102386519784
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B06XG3632R	MaxMara	Yellow



ASIN : B06XG3632R
Brand : MaxMara
euclidean distance from input : 26.545855248925243
==========
==========

Asin	Brand	Color
B010V3B44G	Doxi-Supermall	White
B005BFCIWW	Southern-Lady	Shamrock-Gree



ASIN : B005BFCIWW
Brand : Southern Lady
euclidean distance from input : 26.96576343865363

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

CONCLUTION:-

1:Steps Followed:-

1:We did some EDA on data to unserstand.

2:We did some data pre-processing like removing duplicates fix null value issue.

3:We did some text pre-processing on title like stemming via porter stemmer and stopword removal.

4:Text based product similarity on following using cosine similarity:-

a:TF-IDF based product similarity.

b:IDF based product similarity.

c:Text Semantics based product similarity.

d:Average Word2Vec product similarity.

e:IDF weighted Word2Vec for product similarity.

f:Visual features based product similarity.

5:Weighted similarity using brand and color.

6:Weighted similarity using title(IDF-WT),brand(BOW),color(BOW) and visual features(via C.N.N).

2: OBSERVATIONS:-

1: We observe there are different results when we are using different vectorizatio
n techniques.

2: So we calculate similarity of products with merging or concatinating different
 vector with weights to get better
   results.

3: We also see how good our C.N.N model is working to obtain the features and givi
ng us good results.

4: We can also optimize our results with concatiation of C.N.N features + normal f
eatures(like idf-w2v.bow etc)

5: We also do some experment to give high weight to one of vector in concatinating
vector and we observe expected results as our weights increases bias of t
hat vector in results also reflected which is looking awesome.

6: We can also adjust weight of vector in our production model if we want to give
 high or low preference or weight to any
   of our vector.

In [ ]: