

Amazon Fashion Discovery Engine

[4.2] Data and Code:

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

[4.3] Overview of the data

```
In [2]: #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 [33]: # 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 [34]: 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 [35]: # 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', 'color',
       'editorial_reivew', 'editorial_review', 'formatted_price',
       'large_image_url', 'manufacturer', 'medium_image_url', 'model',
       'product_type_name', 'publisher', 'reviews', 'sku', 'small_image_url',
       'title'],
      dtype='object')
```

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

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

6. title (title of the product.)
7. formatted_price (price of the product)

```
In [36]: data = data[['asin', 'brand', 'color', 'medium_image_url', 'product_type_name', 'title', 'formatted_price']]
```

```
In [37]: 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	formatted_price
0	B016I2TS4W	FNC7C	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Minions Como Superheroes Ironman Long Sleeve R...	None
1	B01N49AI08	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG Clothing Womens Izo Tunic	None
2	B01JDPCOHO	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG Clothing Womens Won Top	None
3	B01N19U5H5	Focal18	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Focal18 Sailor Collar Bubble Sleeve Blouse Shi...	None
4	B004GSI2OS	FeatherLite	Onyx Black/Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	Featherlite Ladies' Long Sleeve Stain Resistan...	\$26.26

[5.1] Missing data for various features.

Basic stats for the feature: product_type_name

```
In [38]: # 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 [39]: # 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 [40]: # 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 [41]: # 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 [42]: 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 [43]: 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 [44]: 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 [45]: 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 [46]: 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 [47]: 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 [48]: 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 [49]: # 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 [50]: # consider products which have color information
# data['color'].isnull() => gives the information about the dataframe row's which have null values pri
ce == None/NULL
data = data.loc[~data['color'].isnull()]
print('Number of data points After eliminating color=NULL :', data.shape[0])
```

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

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

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

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

```
In [51]: data.to_pickle('pickels/28k_apparel_data')
```

```
In [52]: # 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 index, row in images.iterrows():
():\n    url = row['large_image_url']\n    response = requests.get(url)\n    img = Image.o
pen(BytesIO(response.content))\n    img.save('workshop/images/28k_images/'+row['asin']+'.jpeg')\n\n"
```

[5.2] Remove near duplicate items

[5.2.1] Understand about duplicates.

```
In [97]: # 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



:B00G278GZ6

:B00G278W6O



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

In [103]: `data.head()`

Out[103]:

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

In [104]: `# 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 [105]: `# Sort the whole data based on title (alphabetical order of title)
data_sorted.sort_values('title', inplace=True, ascending=False)
data_sorted.head()`

Out[105]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
61973	B06Y1KZ2WB	Éclair	Black/Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	Éclair Women's Printed Thin Strap Blouse Black...	\$24.99
133820	B010RV33VE	xiaoming	Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Womens Sleeveless Loose Long T-shirts...	\$18.19
81461	B01DDSDLNS	xiaoming	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Women's White Long Sleeve Single Brea...	\$21.58
75995	B00X5LYO9Y	xiaoming	Red Anchors	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Stripes Tank Patch/Bear Sleeve Anchor...	\$15.91
151570	B00WPJG35K	xiaoming	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Sleeve Sheer Loose Tassel Kimono Woma...	\$14.32

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

Titles 1:

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

Title 2:

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

Title 3:

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

In [106]:

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

```
In [107]: import itertools
stage1_dedupe_asins = []
i = 0
j = 0
num_data_points = data_sorted.shape[0]
while i < num_data_points and j < num_data_points:

    previous_i = i

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

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

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

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

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

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

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

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

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

```
In [108]: data = data.loc[data['asin'].isin(stage1_dedupe_asins)]
```

We removed the duplicates which differ only at the end.

```
In [109]: print('Number of data points : ', data.shape[0])
Number of data points : 17593
```

```
In [110]: data.to_pickle('pickels/17k_aperal_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 [65]: data = pd.read_pickle('pickels/17k_apperal_data')
```

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

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

stage2_deduplicate_asins = []  
while len(indices) != 0:  
    i = indices.pop()  
    stage2_deduplicate_asins.append(data['asin'].loc[i])  
    # consider the first apparel's title  
    a = data['title'].loc[i].split()  
    # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of', 'Diamond'  
    s', 'Women's', 'Shirt', 'X-Large']  
    for j in indices:  
  
        b = data['title'].loc[j].split()  
        # store the list of words of jth string in b, ex: b = ['tokidoki', 'The', 'Queen', 'of', 'Diamond'  
        s', 'Women's', 'Shirt', 'X-Large']  
  
        length = max(len(a), len(b))  
  
        # count is used to store the number of words that are matched in both strings  
        count = 0  
  
        # itertools.zip_longest(a,b): will map the corresponding words in both strings, it will append  
        d None in case of unequal strings  
        # example: a = ['a', 'b', 'c', 'd']  
        # b = ['a', 'b', 'd']  
        # itertools.zip_longest(a,b): will give [(a, a), (b, b), (c, d), (d, None)]  
        for k in itertools.zip_longest(a, b):  
            if (k[0] == k[1]):  
                count += 1  
  
            # if the number of words in which both strings differ are < 3 , we are considering it as those  
            two apperals are same, hence we are ignoring them  
            if (length - count) < 3:  
                indices.remove(j)
```

```
In [71]: # from whole previous products we will consider only  
# the products that are found in previous cell  
data = data.loc[data['asin'].isin(stage2_deduplicate_asins)]
```

```
In [74]: print('Number of data points after stage two of dedupe: ', data.shape[0])  
# from 17k apperals we reduced to 16k apperals
```

Number of data points after stage two of dedupe: 16042

```
In [75]: 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 [3]: 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 [4]: # 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: {'such', 'and', 'hers', 'up', 'she', 'd', 'further', 'all', 'than', 'under', 'is', 'off', 'both', 'most', 'few', 'should', 're', 'very', 'just', 'then', 'didn', 'myself', 'in', 'too', 's', 'shouldn', 'herself', 'because', 'how', 'itself', 'what', 'shan', 'weren', 'doing', 'them', 'coul dn', 'their', 'so', 'ain', 'haven', 'yourself', 'now', 'll', 'isn', 'about', 'over', 'into', 'before', 'during', 'on', 'as', 'aren', 'against', 'above', 'down', 'they', 'below', 'me', 'again', 'for', 'wh y', 'been', 'yourselves', 'more', 'her', 'that', 'can', 'am', 'was', 'themselves', 'mighthn', 'does', 'those', 'only', 'hasn', 'any', 'ma', 'are', 'nor', 'out', 'you', 'ourselves', 'the', 'an', 'has', 'wh ere', 'i', 'while', 'ours', 'its', 'your', 'had', 'were', 'being', 'no', 'or', 'needn', 've', 'y', 'a', 'each', 'have', 'through', 'when', 'mustn', 'by', 'won', 'from', 'own', 'will', 'there', 't', 'hi m', 'these', 'doesn', 'theirs', 'my', 'did', 'of', 'who', 'until', 'wouldn', 'we', 'do', 'having', 'yo urs', 'other', 'wasn', 'it', 'with', 'once', 'here', 'don', 'o', 'whom', 'this', 'if', 'but', 'hadn', 'our', 'some', 'm', 'not', 'between', 'himself', 'same', 'at', 'be', 'he', 'after', 'which', 'to', 'hi s'}
```

```
In [5]: 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")
```

```
3.572722000000006 seconds
```

```
In [6]: data.head()
```

Out[6]:

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

```
In [8]: data.to_pickle('pickels/16k_appral_data_preprocessed')
```

Stemming

```
In [7]: from nltk.stem.porter import *
stemmer = PorterStemmer()
print(stemmer.stem('arguing'))
print(stemmer.stem('fishing'))

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

argu
fish

[8] Text based product similarity

```
In [10]: data = pd.read_pickle('pickels/16k_apperal_data_preprocessed')
data.head()
```

Out[10]:

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

In [11]: # Utility Functions which we will use through the rest of the workshop.

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

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

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

    # 1st, plotting heat map that represents the count of commonly occurred words in title2
    ax = plt.subplot(gs[0])
    # it displays a cell in white color if the word is intersection(lis of words of title1 and lis
    # of words of title2), in black if not
    ax = sns.heatmap(np.array([values]), annot=np.array([labels]))
    ax.set_xticklabels(keys) # set that axis labels as the words of title
    ax.set_title(text) # apparel title

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

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

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

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

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

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

    # we set the values of non intersecting words to zero, this is just to show the difference in heat
    # map
    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): values(i)=coun
    # t of that word in title2 else values(i)=0
    values = [vec2[x] for x in vec2.keys()]

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

```

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

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

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

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

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

    # vector2 = dict{word1:#count, word2:#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 [17]:

```

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

# What is a sparse vector?

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

```

Out[17]: (16042, 12609)

```
In [22]: def bag_of_words_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

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

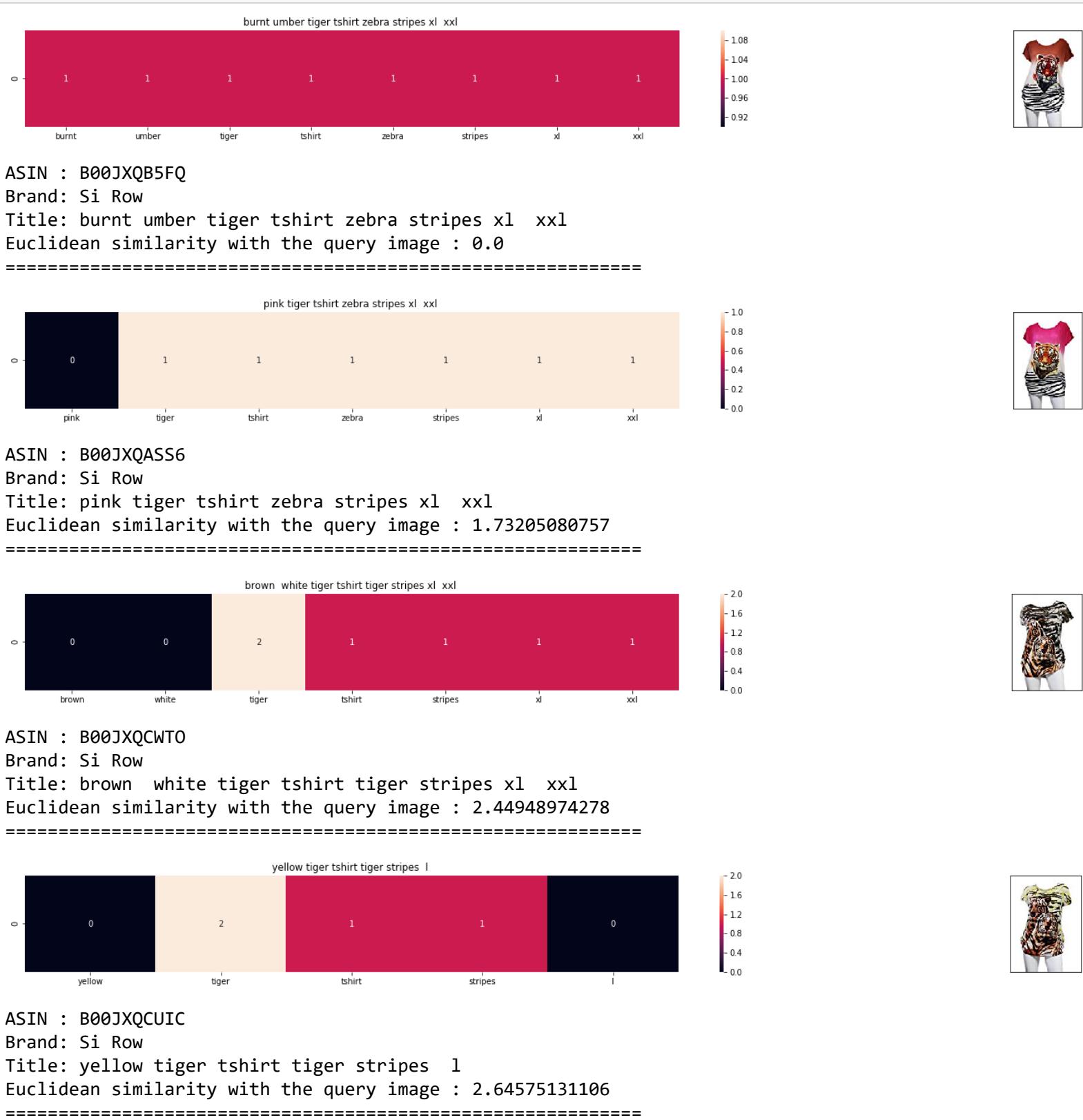
    # np.argsort will return indices of the smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the smallest distances
    pdists = np.sort(pairwise_dist.flatten())[0:num_results]

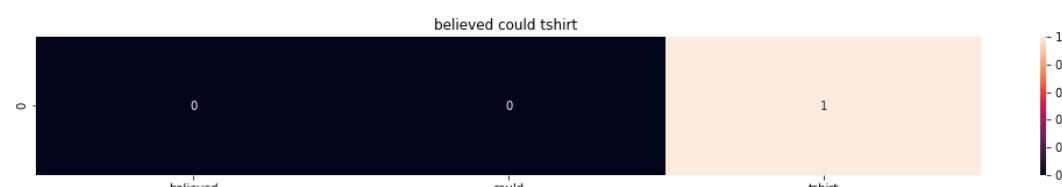
    #data frame indices of the 9 smallest distace's
    df_indices = list(data.index[indices])

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

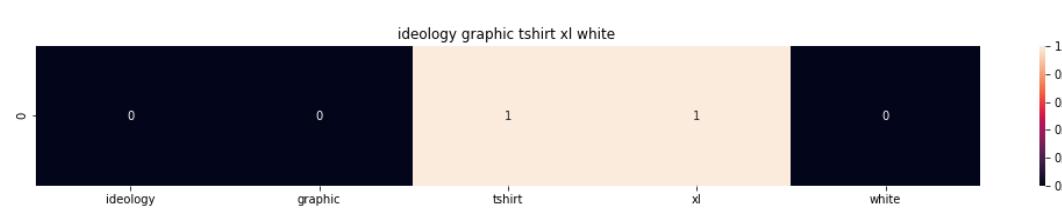
#call the bag-of-words model for a product to get similar products.
bag_of_words_model(12566, 20) # change the index if you want to.
# In the output heat map each value represents the count value
# of the label word, the color represents the intersection
# with inputs title.

#try 12566
#try 931
```

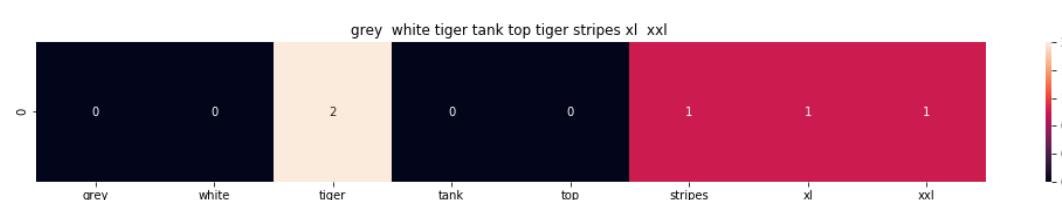




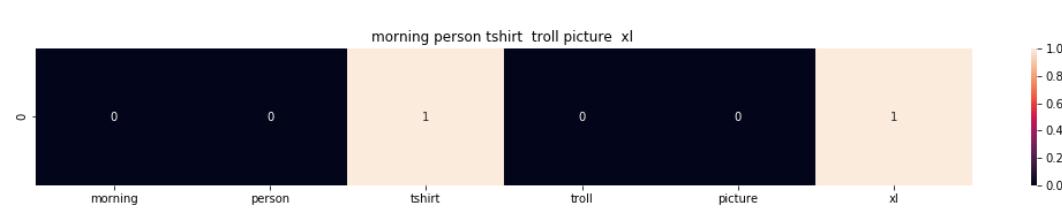
ASIN : B07568NZX4
 Brand: Rustic Grace
 Title: believed could tshirt
 Euclidean similarity with the query image : 3.0



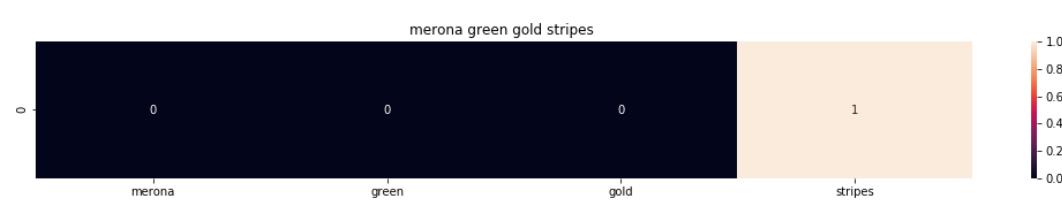
ASIN : B01NB0NKRO
 Brand: Ideology
 Title: ideology graphic tshirt xl white
 Euclidean similarity with the query image : 3.0



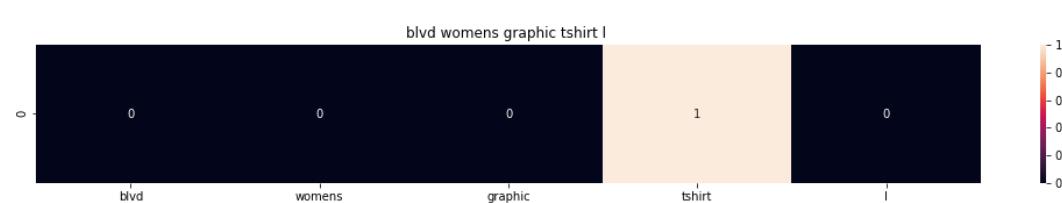
ASIN : B00JXQAFZ2
 Brand: Si Row
 Title: grey white tiger tank top tiger stripes xl xxl
 Euclidean similarity with the query image : 3.0



ASIN : B01CLS8LMW
 Brand: Awake
 Title: morning person tshirt troll picture xl
 Euclidean similarity with the query image : 3.16227766017



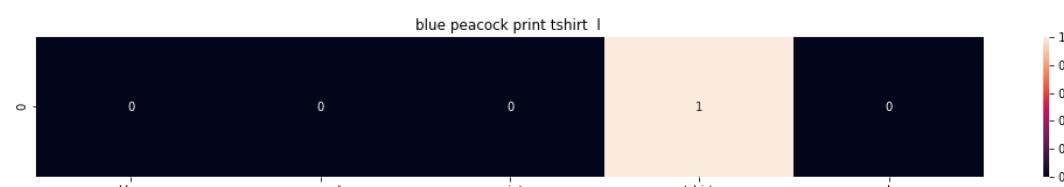
ASIN : B01KVZUB6G
 Brand: Merona
 Title: merona green gold stripes
 Euclidean similarity with the query image : 3.16227766017



ASIN : B0733R2CJK
 Brand: BLVD
 Title: blvd womens graphic tshirt l
 Euclidean similarity with the query image : 3.16227766017



ASIN : B012VQLT6Y
 Brand: KM T-shirt
 Title: km tiger printed sleeveless vest tshirt
 Euclidean similarity with the query image : 3.16227766017



ASIN : B00JXQC8L6

Brand: Si Row

Title: blue peacock print tshirt l

Euclidean similarity with the query image : 3.16227766017



ASIN : B06XC3CF6

Brand: Fjallraven

Title: fjallraven womens ovik tshirt plum xxl

Euclidean similarity with the query image : 3.16227766017

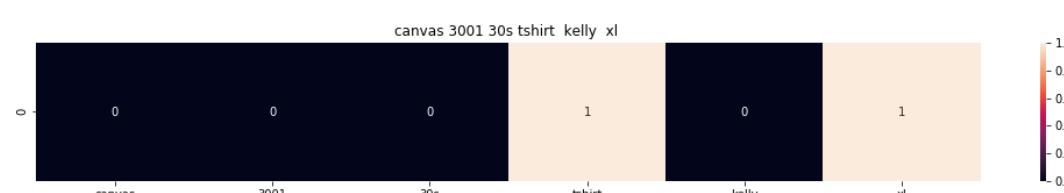


ASIN : B005IT80BA

Brand: Hetalia

Title: hetalia us girl tshirt

Euclidean similarity with the query image : 3.16227766017

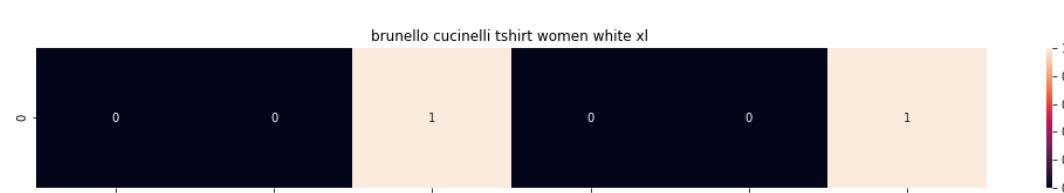


ASIN : B0088PN0LA

Brand: Red House

Title: canvas 3001 30s tshirt kelly xl

Euclidean similarity with the query image : 3.16227766017



ASIN : B06X99V6WC

Brand: Brunello Cucinelli

Title: brunello cucinelli tshirt women white xl

Euclidean similarity with the query image : 3.16227766017



ASIN : B06Y1JPW1Q

Brand: Xhilaration

Title: xhilaration womens lace tshirt salmon xxl

Euclidean similarity with the query image : 3.16227766017

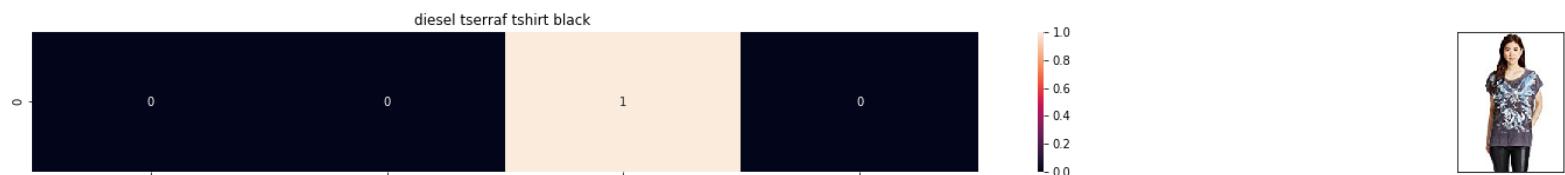


ASIN : B06X6GX6WG

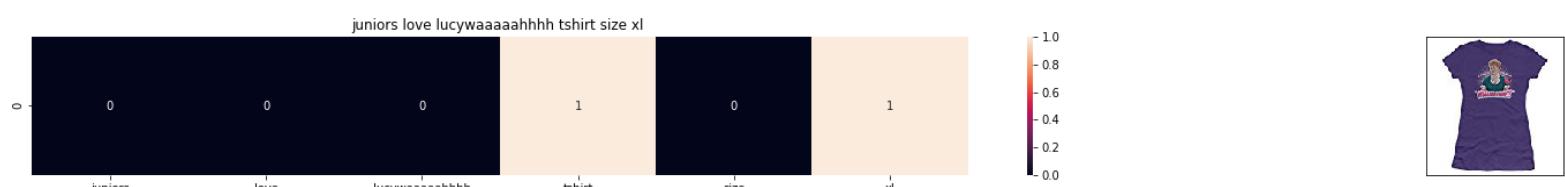
Brand: Animal

Title: animal oceania tshirt yellow

Euclidean similarity with the query image : 3.16227766017



ASIN : B017X8PW9U
Brand: Diesel
Title: diesel tserraf tshirt black
Euclidean similarity with the query image : 3.16227766017



ASIN : B00IAA4JIQ
Brand: I Love Lucy
Title: juniors love lucywaaaaahhhh tshirt size xl
Euclidean similarity with the query image : 3.16227766017
=====

[8.5] TF-IDF based product similarity

```
In [23]: tfidf_title_vectorizer = TfidfVectorizer(min_df = 0)
tfidf_title_features = tfidf_title_vectorizer.fit_transform(data['title'])
# tfidf_title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(courpus) returns the a sparase matrix of dimensions #data_points * #
words_in_corpus
# tfidf title features[doc id, index of word in corpus] = tfidf values of the word in given doc
```

```
In [38]: def tfidf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

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

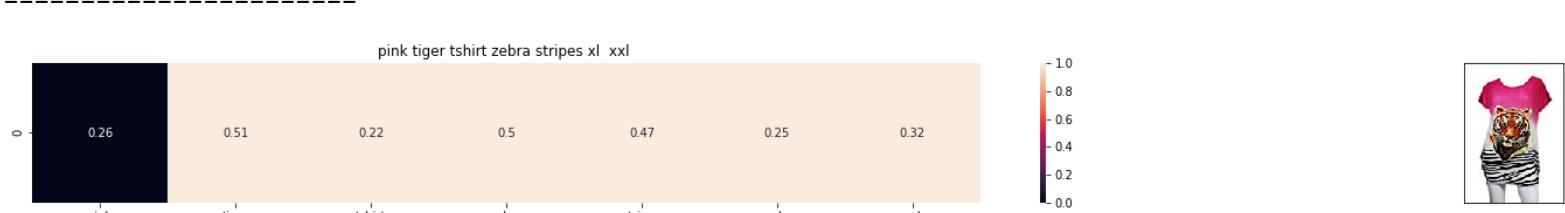
        # np.argsort will return indices of 9 smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists = np.sort(pairwise_dist.flatten())[0:num_results]

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

    for i in range(0,len(indices)):
        # we will pass 1. doc_id, 2. title1, 3. title2, url, model
        get_result(indices[i], data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]], 'tfidf')
            print('ASIN :',data['asin'].loc[df_indices[i]])
            print('BRAND :',data['brand'].loc[df_indices[i]])
            print ('Eucliden distance from the given image :', pdists[i])
            print('*'*125)
tfidf_model(12566, 20)
# in the output heat map each value represents the tfidf values of the label word, the color represents the intersection with inputs title
```



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



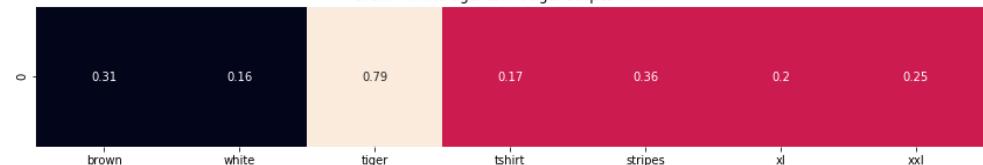
ASIN : B00JXQASS6

BRAND : Si Row

Euclidean distance from the given image : 0.753633191245

=====

brown white tiger tshirt tiger stripes xl xxl



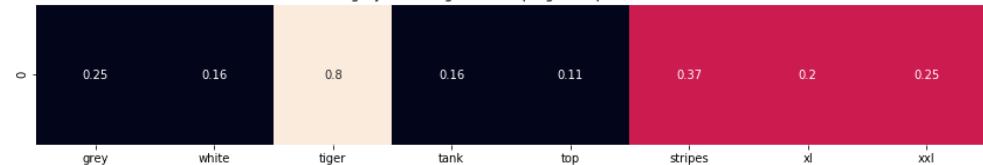
ASIN : B00JXQCWT0

BRAND : Si Row

Euclidean distance from the given image : 0.935764394377

=====

grey white tiger tank top tiger stripes xl xxl



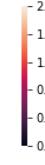
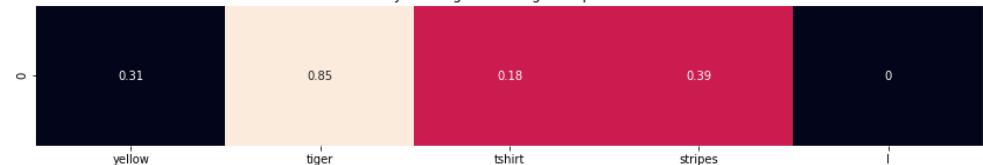
ASIN : B00JXQAFZ2

BRAND : Si Row

Euclidean distance from the given image : 0.95861535242

=====

yellow tiger tshirt tiger stripes l



ASIN : B00JXQCUIC

BRAND : Si Row

Euclidean distance from the given image : 1.00007496145

=====

black white tiger tank top tiger stripes l



ASIN : B00JXQA094

BRAND : Si Row

Euclidean distance from the given image : 1.02321555246

=====

yellow tiger tank top tiger stripes l



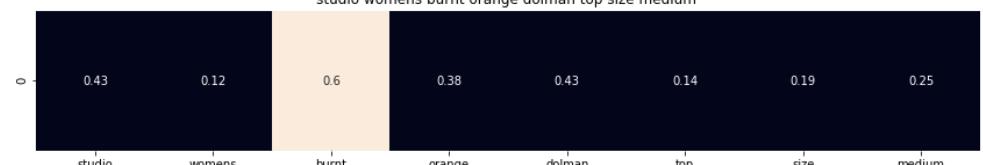
ASIN : B00JXQAUWA

BRAND : Si Row

Euclidean distance from the given image : 1.0319918463

=====

studio womens burnt orange dolman top size medium



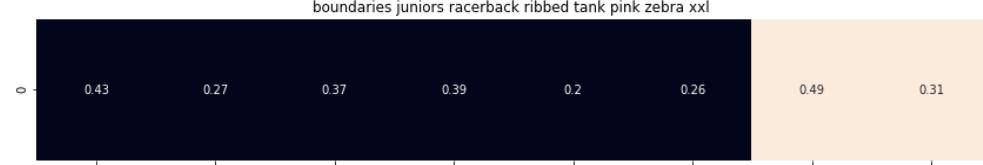
ASIN : B06XSCVFT5

BRAND : Studio M

Euclidean distance from the given image : 1.21068436704

=====

boundaries juniors racerback ribbed tank pink zebra xxl

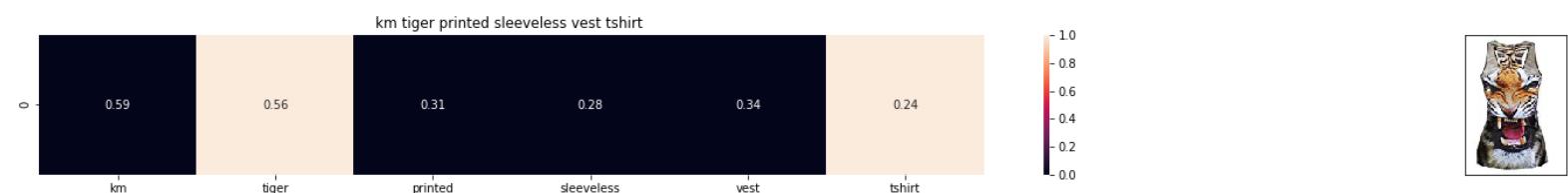


ASIN : B06Y2GTYPM

BRAND : No Boundaries

Euclidean distance from the given image : 1.21216838107

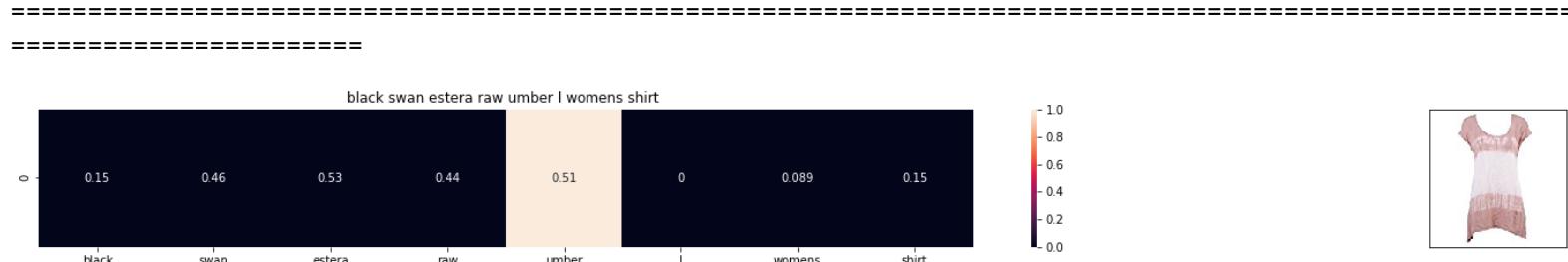
=====



ASIN : B012VQLT6Y

BRAND : KM T-shirt

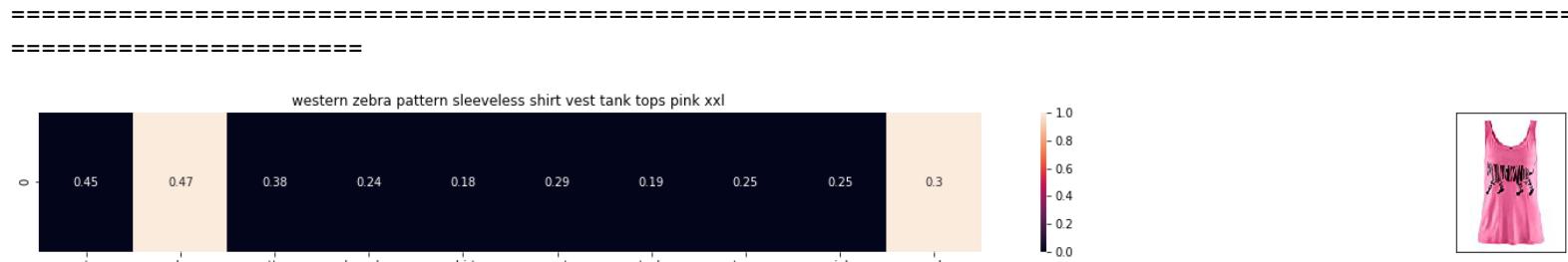
Euclidean distance from the given image : 1.21979064028



ASIN : B06Y1VN8WQ

BRAND : Black Swan

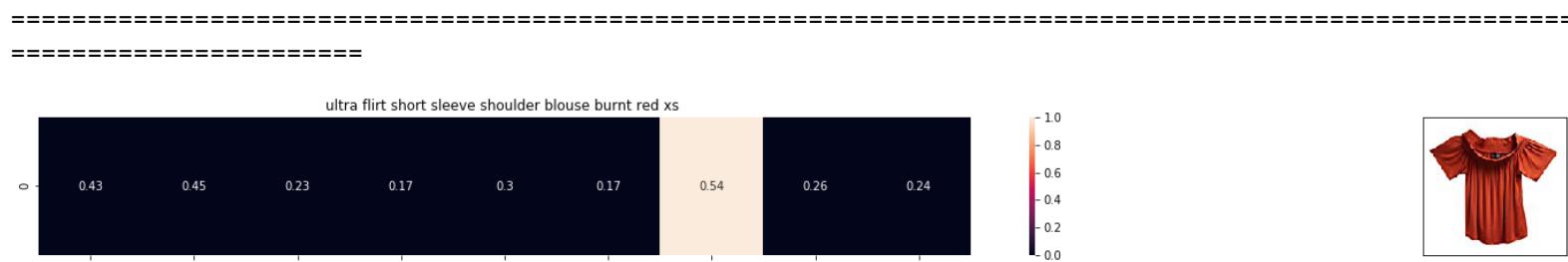
Euclidean distance from the given image : 1.220684966



ASIN : B00Z6HEXWI

BRAND : Black Temptation

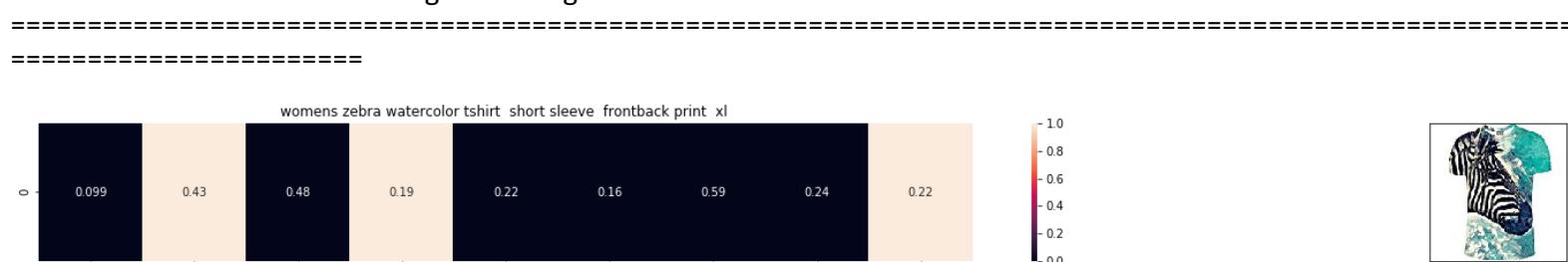
Euclidean distance from the given image : 1.22128139212



ASIN : B074TR12BH

BRAND : Ultra Flirt

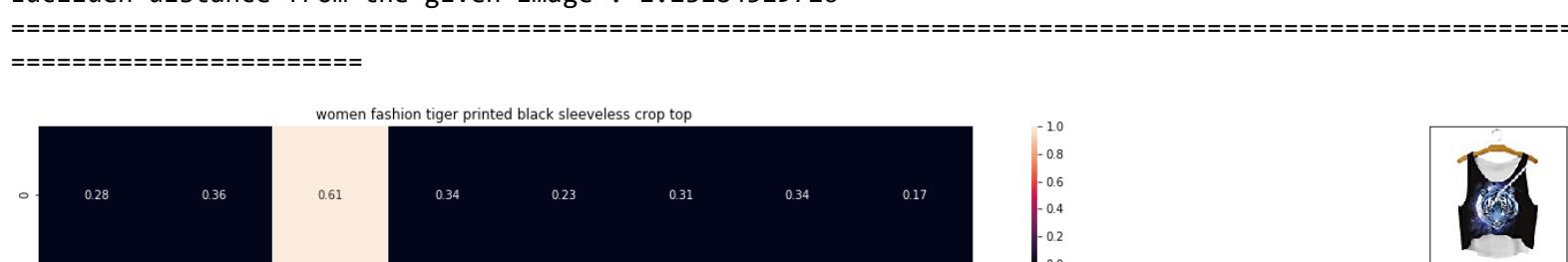
Euclidean distance from the given image : 1.23133640946



ASIN : B072R2JXKW

BRAND : WHAT ON EARTH

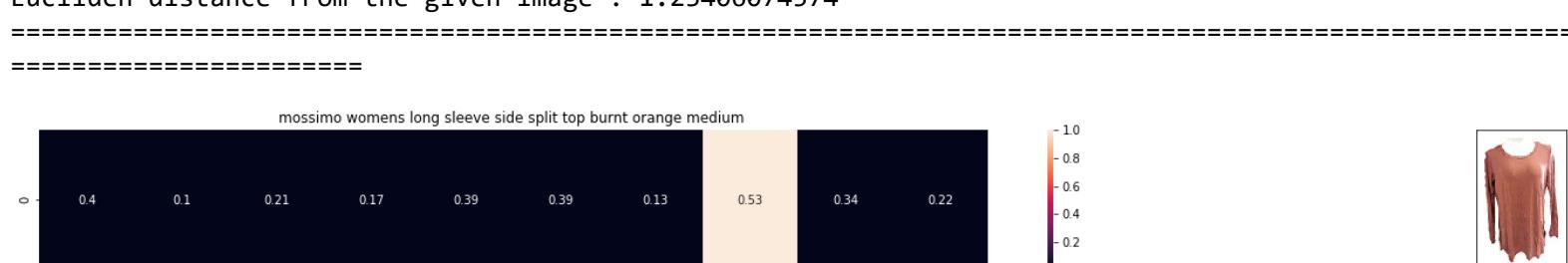
Euclidean distance from the given image : 1.23184519726



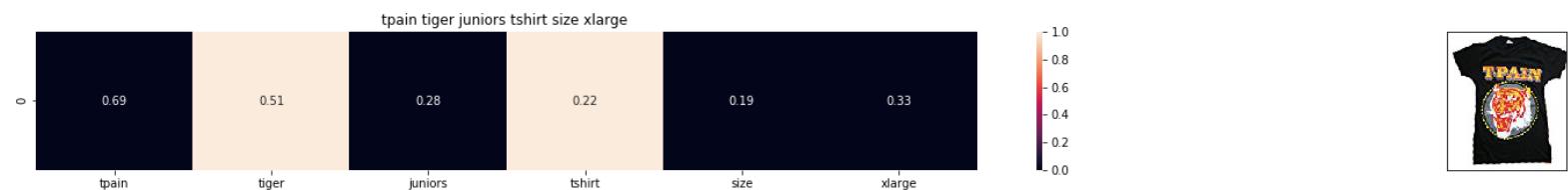
ASIN : B074T8ZYGX

BRAND : MKP Crop Top

Euclidean distance from the given image : 1.23406074574



ASIN : B071ZDF6T2
BRAND : Mossimo
Euclidean distance from the given image : 1.23527855777



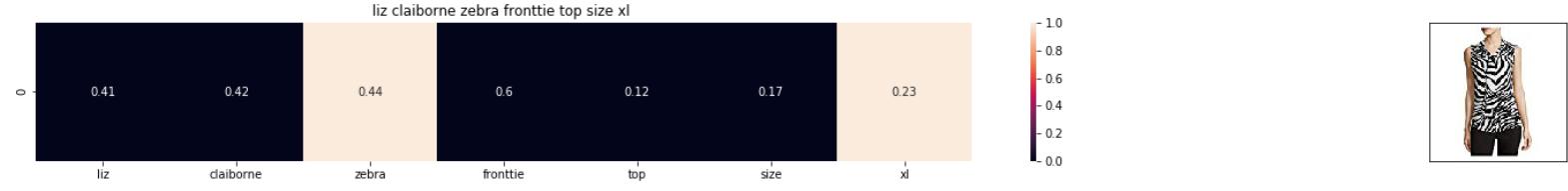
ASIN : B01K0H020G
BRAND : Tultex
Euclidean distance from the given image : 1.23645729881



ASIN : B00H8A6ZLI
BRAND : Vivian's Fashions
Euclidean distance from the given image : 1.24996155053



ASIN : B010NN9RX0
BRAND : YICHUN
Euclidean distance from the given image : 1.25354614209



ASIN : B06XBY5QXL
BRAND : Liz Claiborne
Euclidean distance from the given image : 1.25388329384

[8.5] IDF based product similarity

```
In [33]: idf_title_vectorizer = CountVectorizer()
idf_title_features = idf_title_vectorizer.fit_transform(data['title'])

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

```
In [34]: def nContaining(word):
    # return the number of documents which had the given word
    return sum(1 for blob in data['title'] if word in blob.split())

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

```
In [35]: # we need to convert the values into float
idf_title_features = idf_title_features.astype(np.float)

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

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

```
In [39]: def idf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

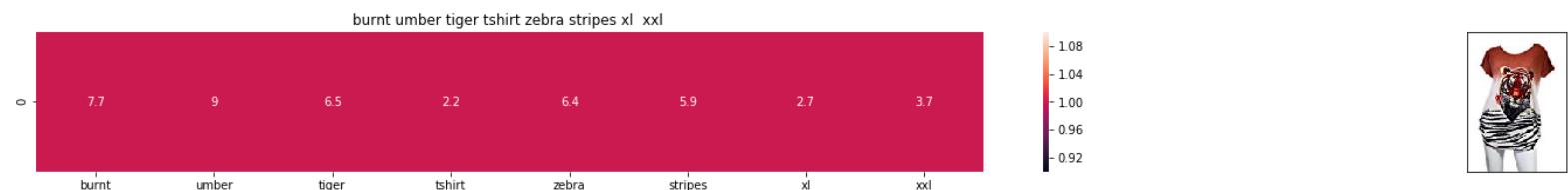
    # pairwise_dist will store the distance from given input apparel to all remaining apparels
    # the metric we used here is cosine, the coside distance is mesured as  $K(X, Y) = \langle X, Y \rangle / (\|X\| * \|Y\|)$ 
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist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

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

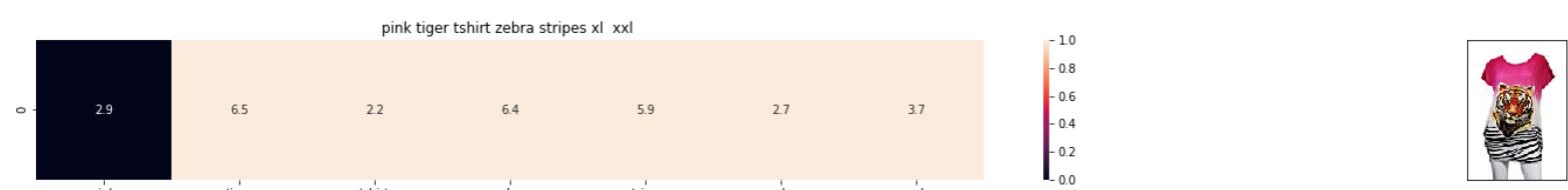
    #data frame indices of the 9 smallest distace's
    df_indices = list(data.index[indices])

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

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



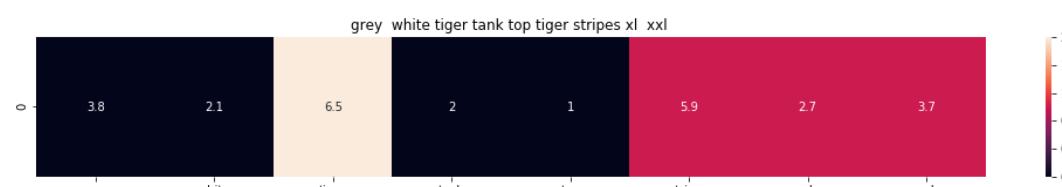
ASIN : B00JXQB5FQ
 Brand : Si Row
 euclidean distance from the given image : 0.0
 ======



ASIN : B00JXQASS6
 Brand : Si Row
 euclidean distance from the given image : 12.2050713112
 ======



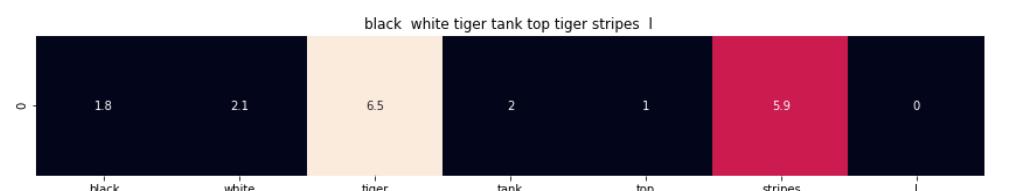
ASIN : B00JXQCWT0
 Brand : Si Row
 euclidean distance from the given image : 14.4683626856
 ======



ASIN : B00JXQAFZ2

Brand : Si Row

euclidean distance from the given image : 14.4868329248



ASIN : B00JXQA094

Brand : Si Row

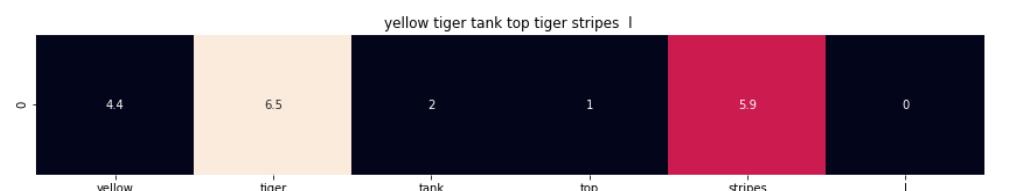
euclidean distance from the given image : 14.8333929667



ASIN : B00JXQCUIC

Brand : Si Row

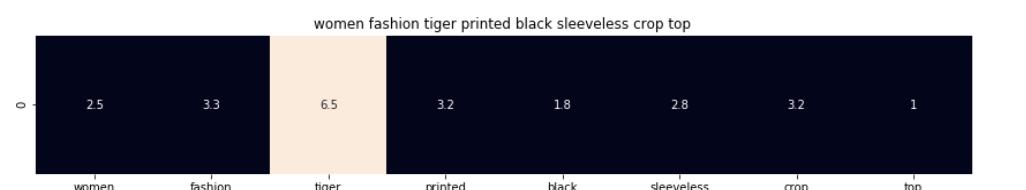
euclidean distance from the given image : 14.8987445167



ASIN : B00JXQAUWA

Brand : Si Row

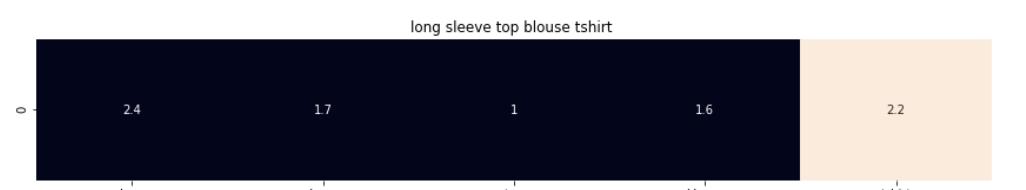
euclidean distance from the given image : 15.2244582873



ASIN : B074T8ZYGX

Brand : MKP Crop Top

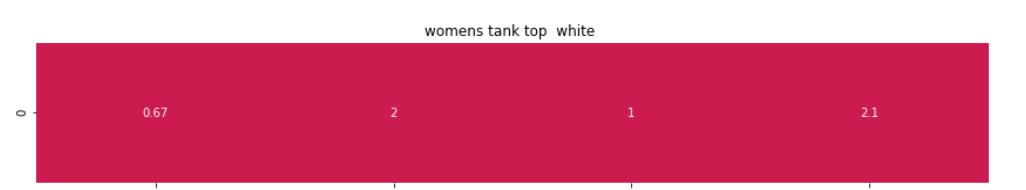
euclidean distance from the given image : 17.0808129556



ASIN : B00KF2N5PU

Brand : Vietsbay

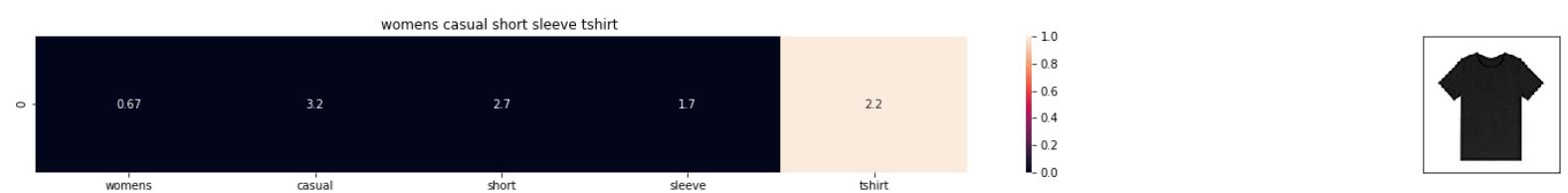
euclidean distance from the given image : 17.0901681256



ASIN : B00JPOZ9GM

Brand : Sofra

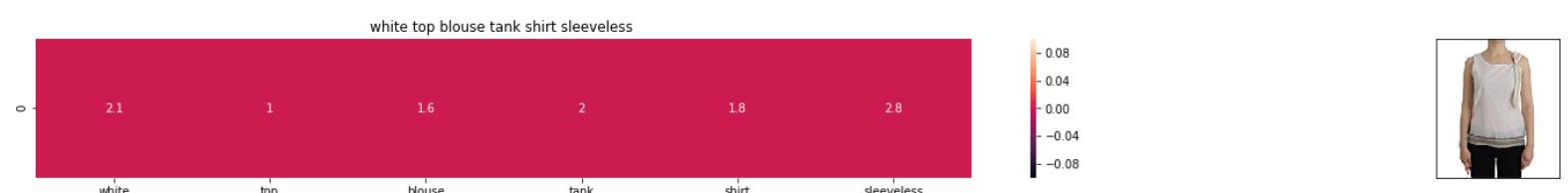
euclidean distance from the given image : 17.1532153376



ASIN : B074T9KG9Q
Brand : Rain
euclidean distance from the given image : 17.3367152387



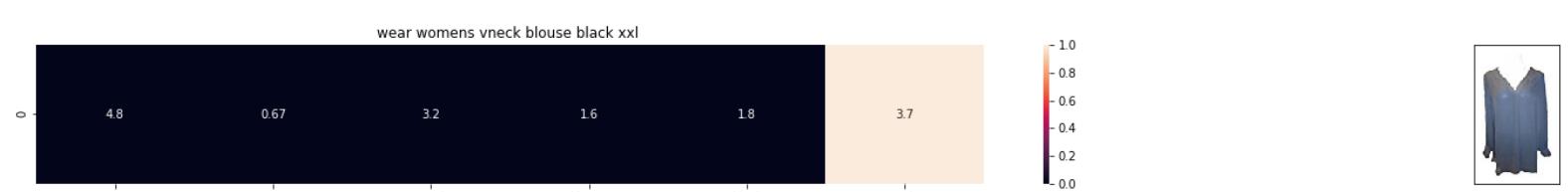
ASIN : B00H8A6ZLI
Brand : Vivian's Fashions
euclidean distance from the given image : 17.410075941



ASIN : B074G5G5RK
Brand : ERMANNO SCERVINO
euclidean distance from the given image : 17.5399213355



ASIN : B06XSCVFT5
Brand : Studio M
euclidean distance from the given image : 17.6127585437



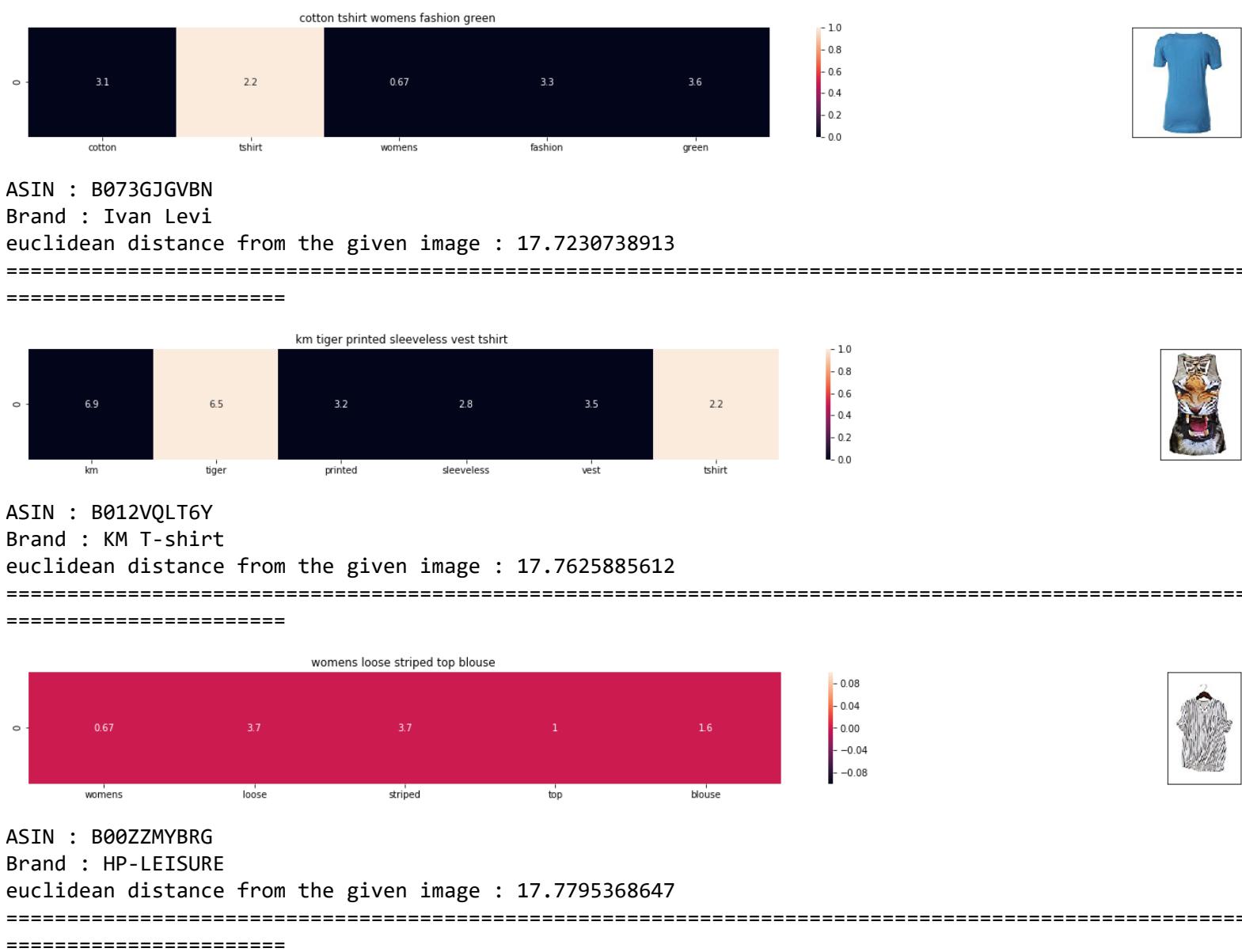
ASIN : B06Y6FH453
Brand : Who What Wear
euclidean distance from the given image : 17.6237452825



ASIN : B074V45DCX
Brand : Rain
euclidean distance from the given image : 17.6343424968



ASIN : B07583CQFT
Brand : Very J
euclidean distance from the given image : 17.6375371274



[9] Text Semantics based product similarity

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

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

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

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

...
```

```
In [27]: from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

# in this project we are using a pretrained model by google
# its 3.3G file, once you load this into your memory
# it occupies ~9Gb, so please do this step only if you have >12G of ram
# we will provide a pickle file which contains a dict ,
# and it contains all our corpus words as keys and model[word] as values
# To use this code-snippet, download "GoogleNews-vectors-negative300.bin"
# from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTLSS21pQmM/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 [28]: # Utility functions

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

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

    final_dist = []
    # for each vector in vec1 we calculate 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):
    # sentence1 : title1, input apparel
    # sentence2 : title2, recommended apparel
    # url: apparel image url
    # doc_id1: document id of input apparel
    # doc_id2: document id of recommended apparel
    # model: it can have two values, 1. avg 2. weighted

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

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

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

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

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

    plt.show()
```

```
In [29]: # vocab = stores all the words that are there in google w2v model
# vocab = model.wv.vocab.keys() # if you are using Google word2Vec

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

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

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

[9.2] Average Word2Vec product similarity.

```
In [30]: 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 corpus * 300), each row corresponds to a doc
w2v_title = np.array(w2v_title)
```

```
In [31]: def avg_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

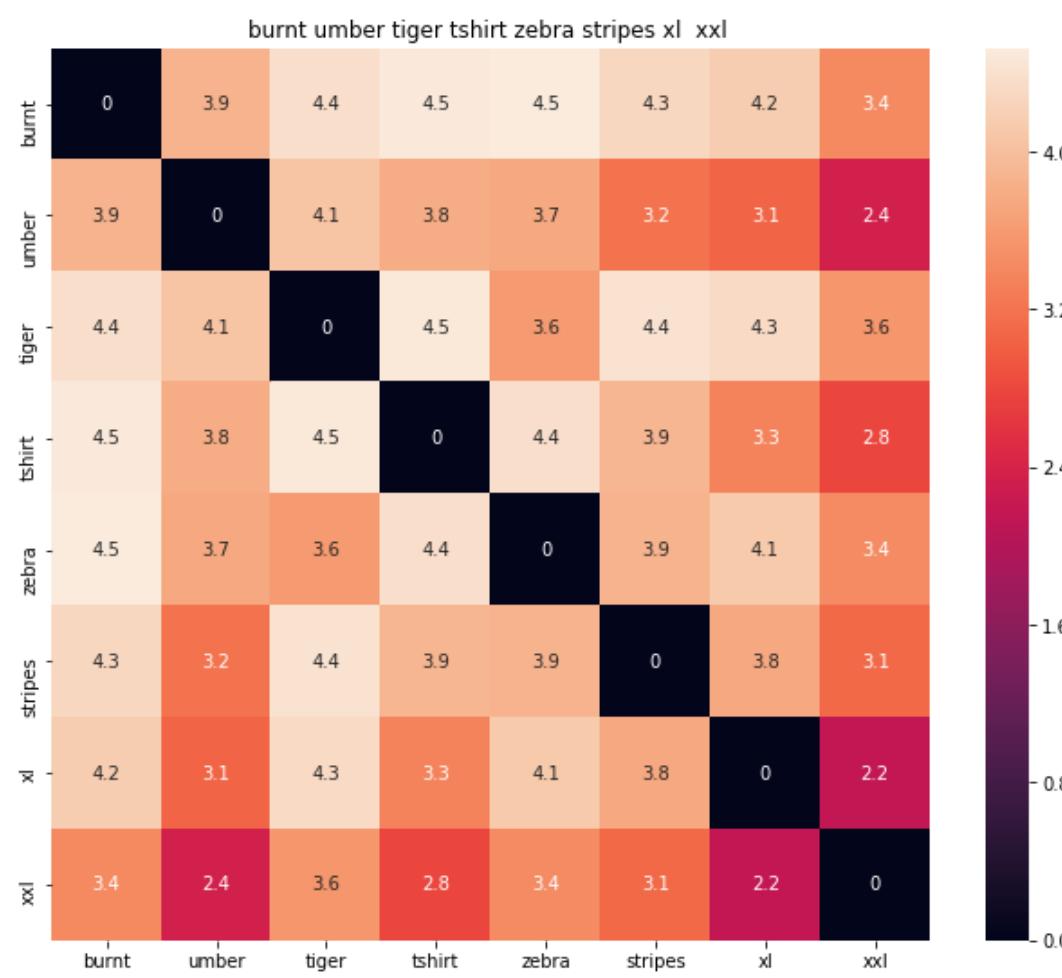
    # dist(x, y) = sqrt(dot(x, x) - 2 * dot(x, y) + dot(y, y))
    pairwise_dist = pairwise_distances(w2v_title, w2v_title[doc_id].reshape(1,-1))

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

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

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

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



ASIN : B00JXQB5FQ

BRAND : Si Row

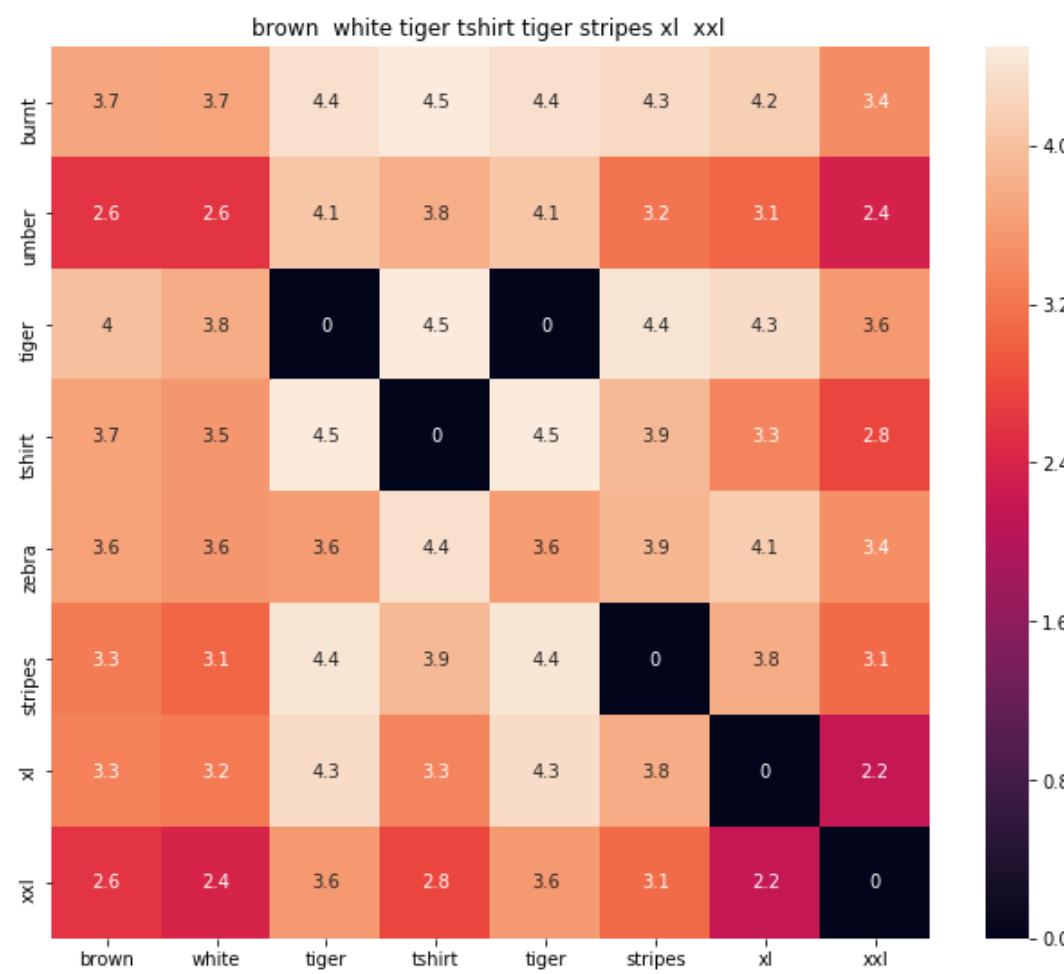
euclidean distance from given input image : 0.000690534



ASIN : B00JXQASS6

BRAND : Si Row

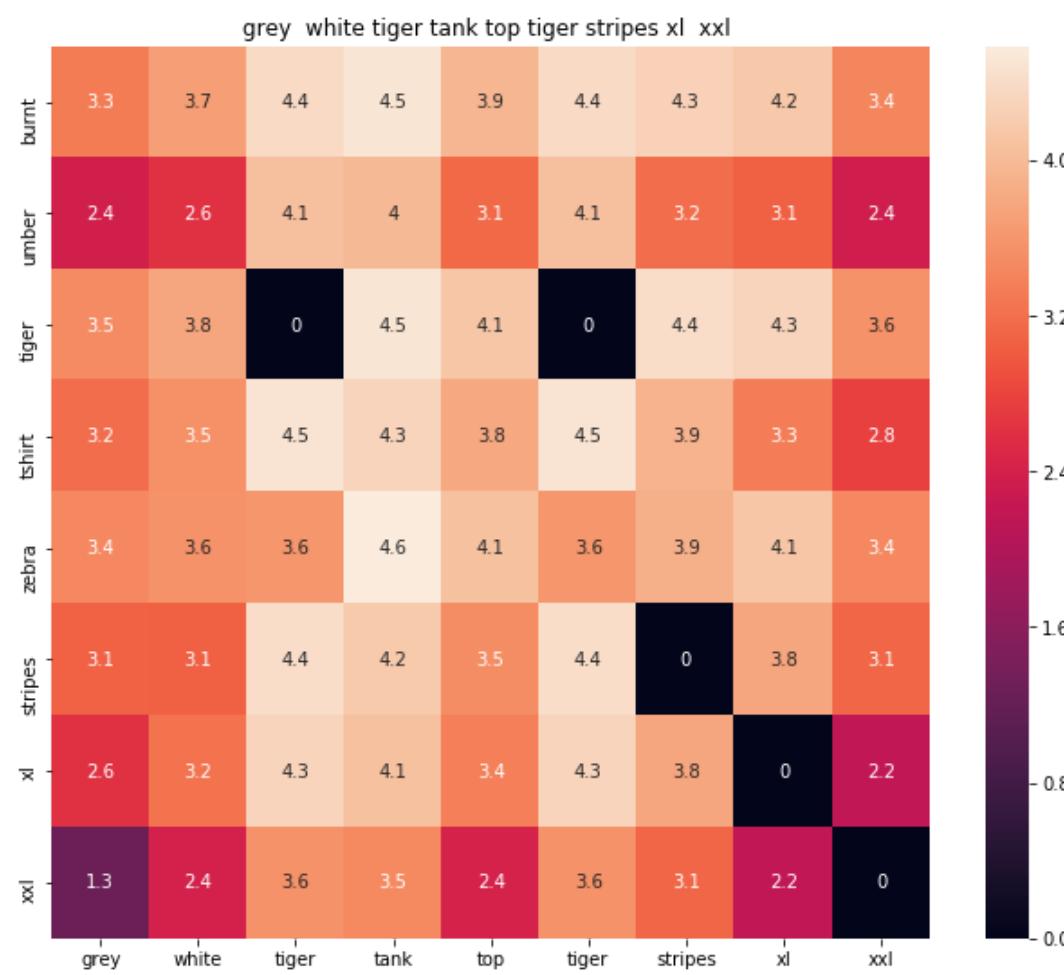
euclidean distance from given input image : 0.589193



ASIN : B00JXQCWTO

BRAND : Si Row

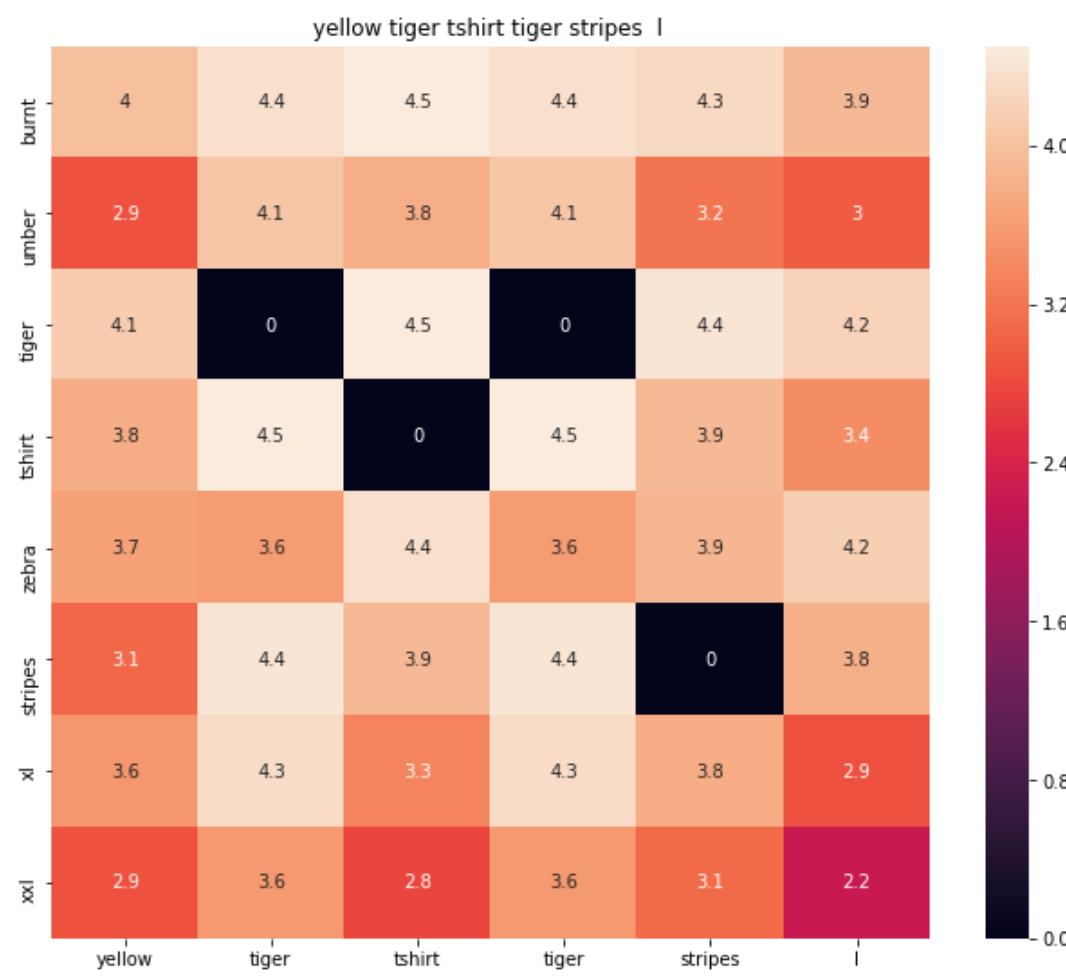
euclidean distance from given input image : 0.700344



ASIN : B00JXQAFZ2

BRAND : Si Row

euclidean distance from given input image : 0.89284



ASIN : B00JXQCUIC

BRAND : Si Row

euclidean distance from given input image : 0.956013



ASIN : B073R5Q8HD

BRAND : Colosseum

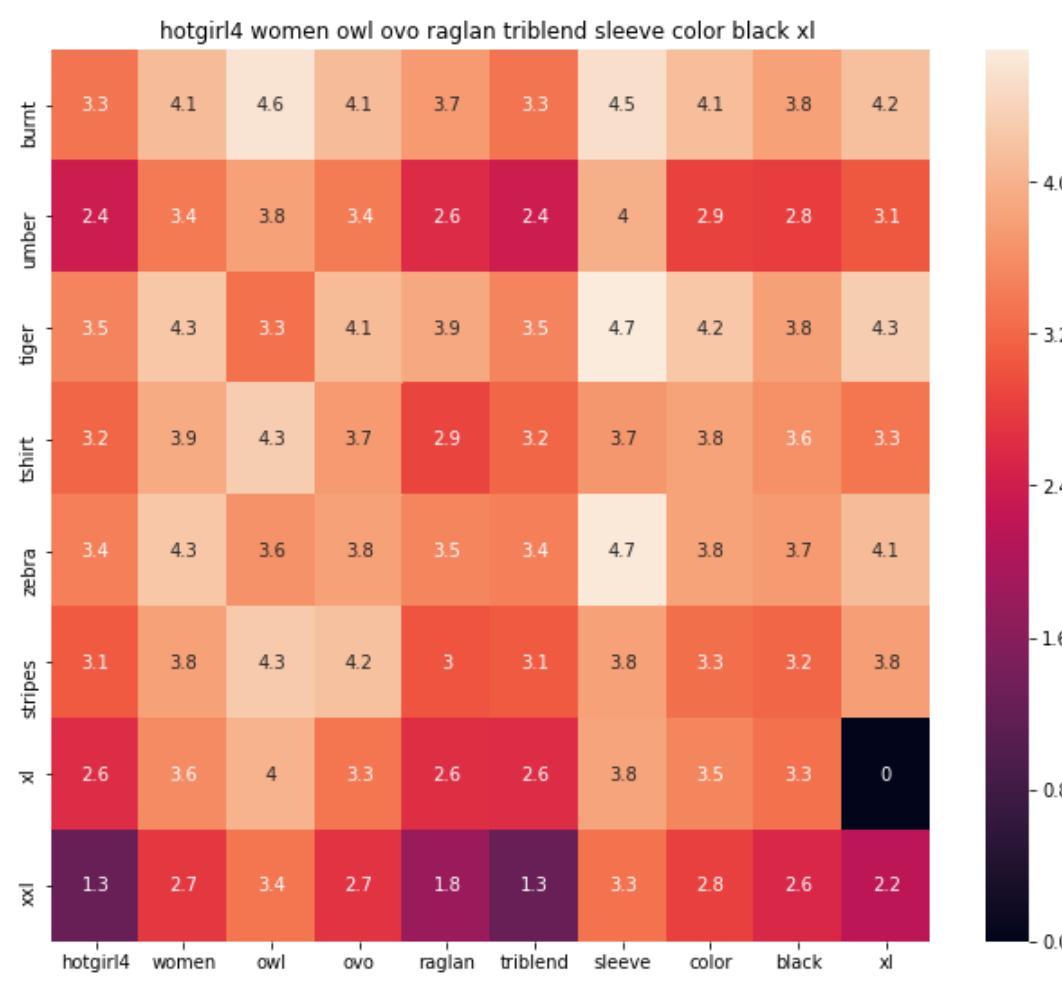
euclidean distance from given input image : 1.02297



ASIN : B06XBY5QXL

BRAND : Liz Claiborne

euclidean distance from given input image : 1.06693



ASIN : B01L8L73M2

BRAND : Hotgirl4 Raglan Design

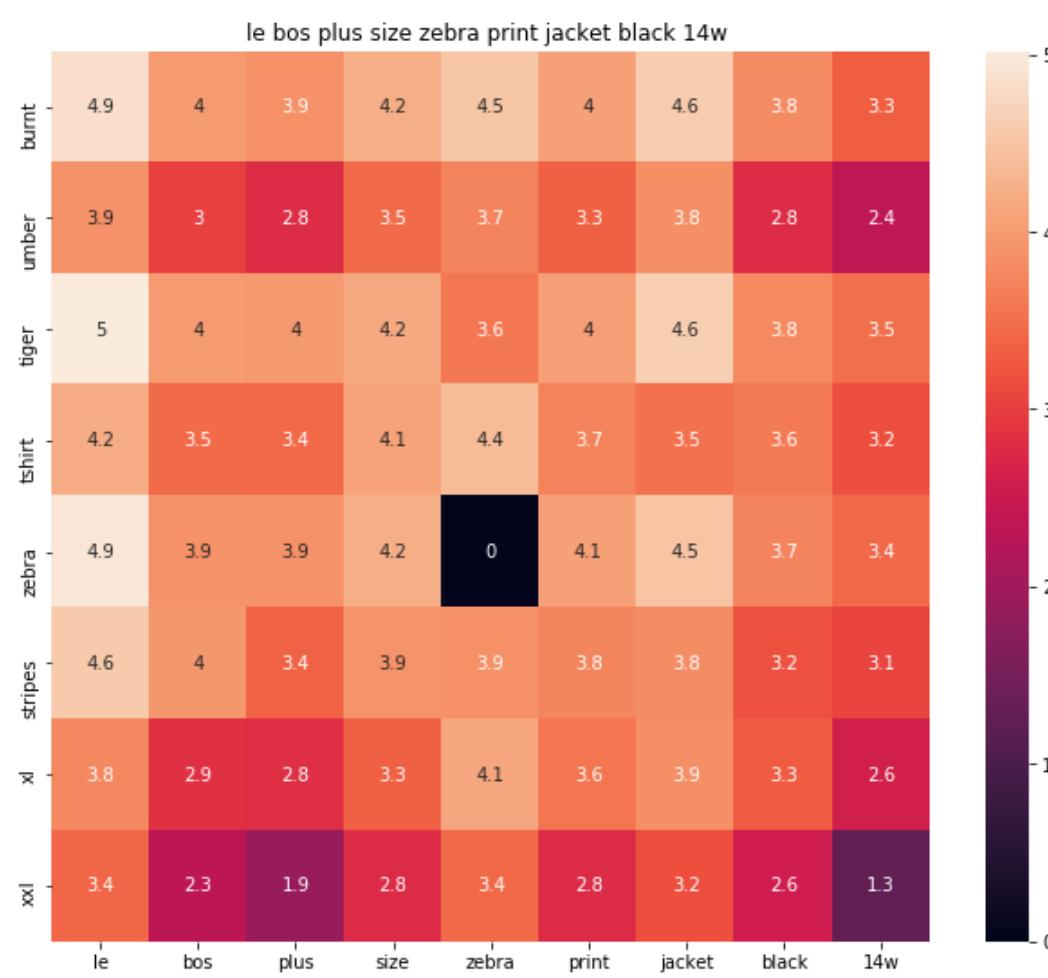
euclidean distance from given input image : 1.07314



ASIN : B01EJS5H06

BRAND : Vansty

euclidean distance from given input image : 1.07572



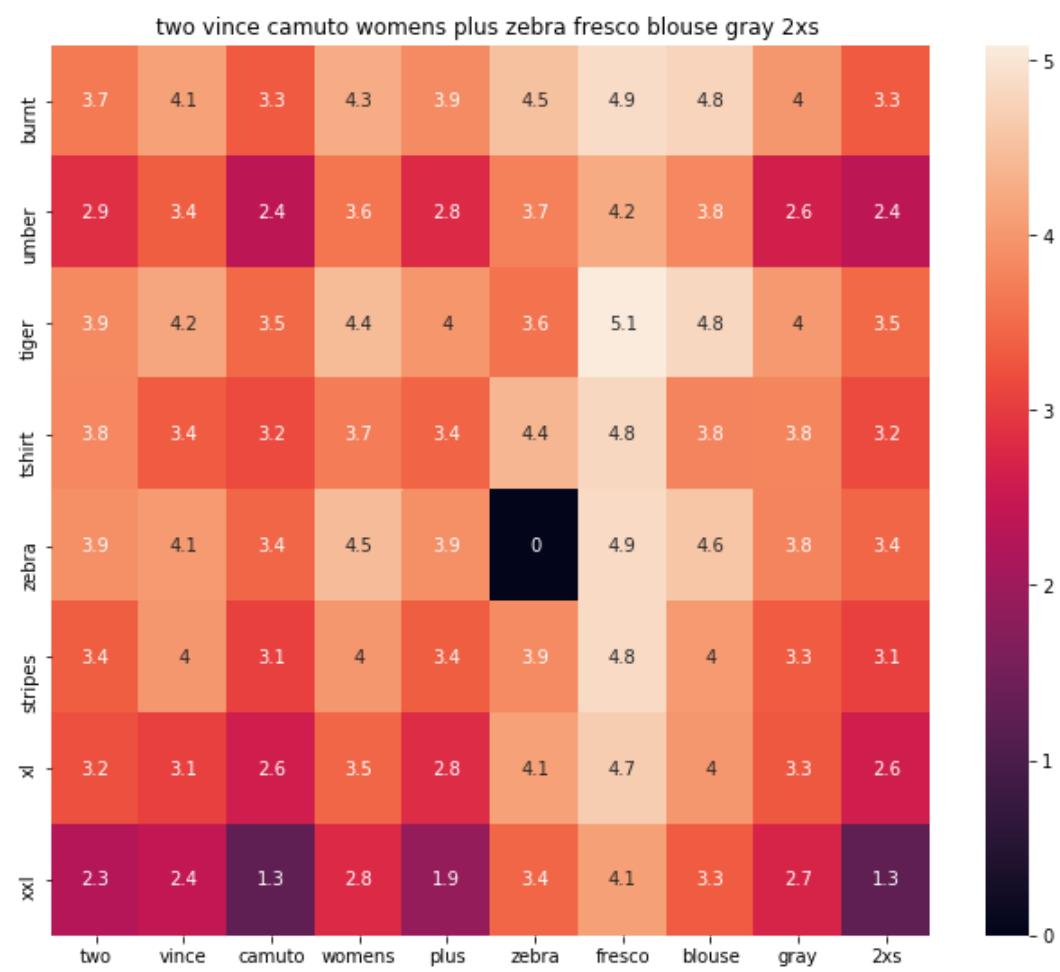
ASIN : B01B01XRK8

BRAND : Le Bos

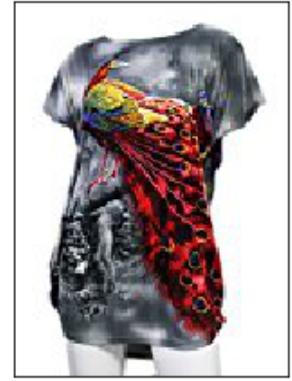
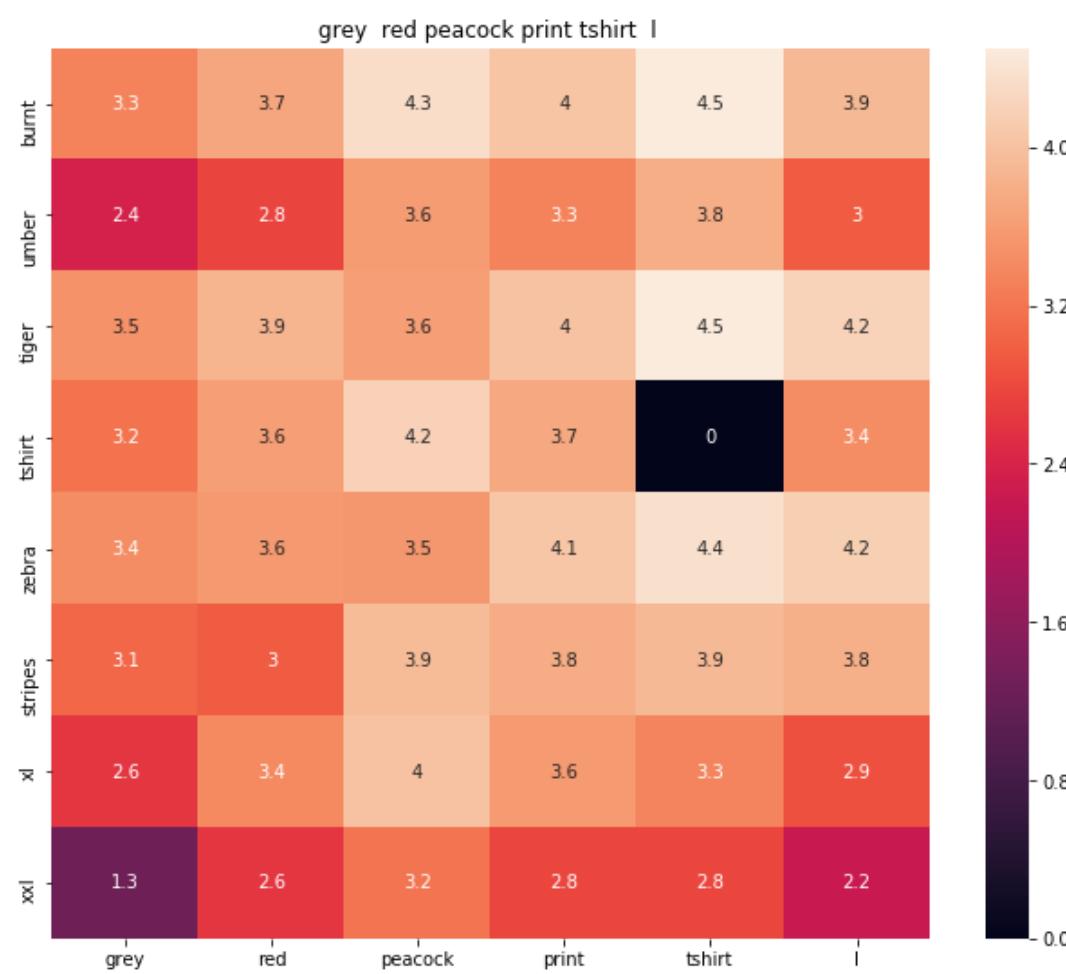
euclidean distance from given input image : 1.084



ASIN : B072R2JXKW
 BRAND : WHAT ON EARTH
 euclidean distance from given input image : 1.08422



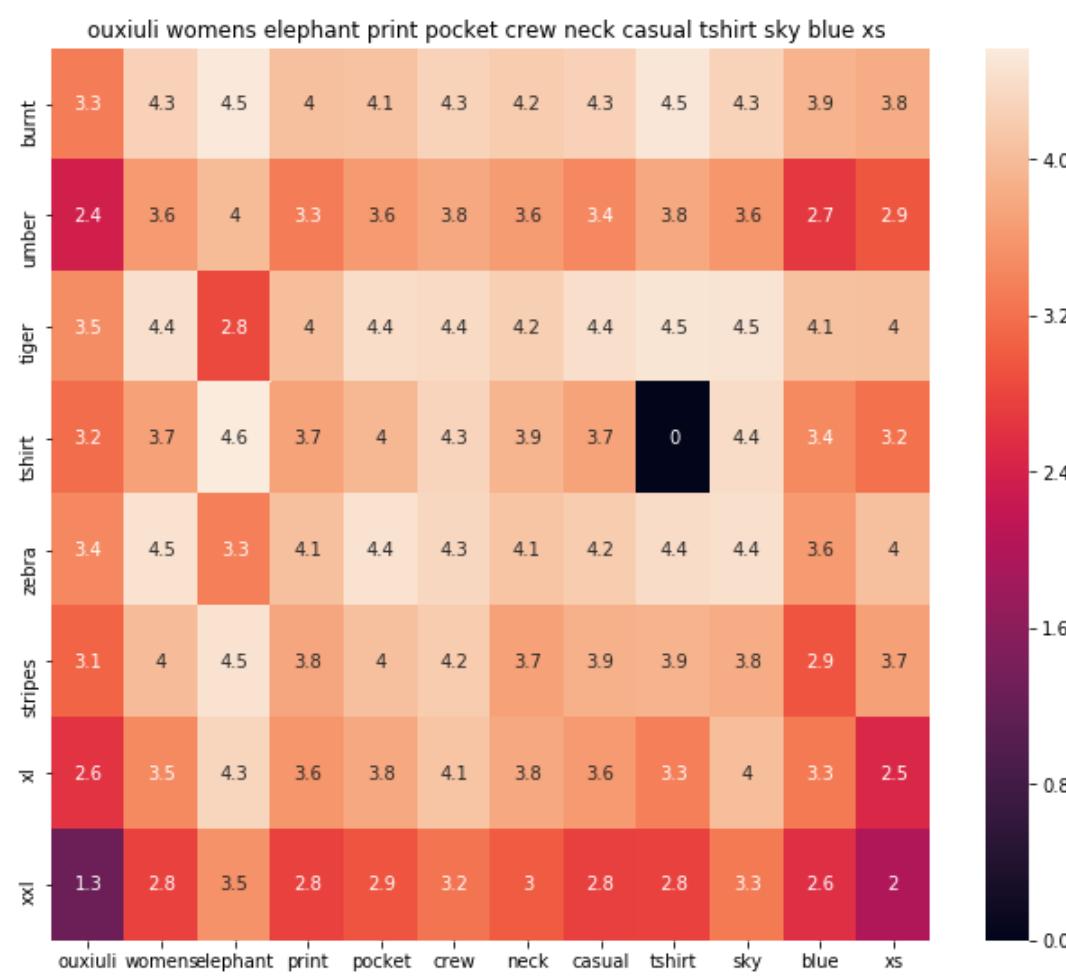
ASIN : B074MJRGW6
 BRAND : Two by Vince Camuto
 euclidean distance from given input image : 1.0895



ASIN : B00JXQCFRS

BRAND : Si Row

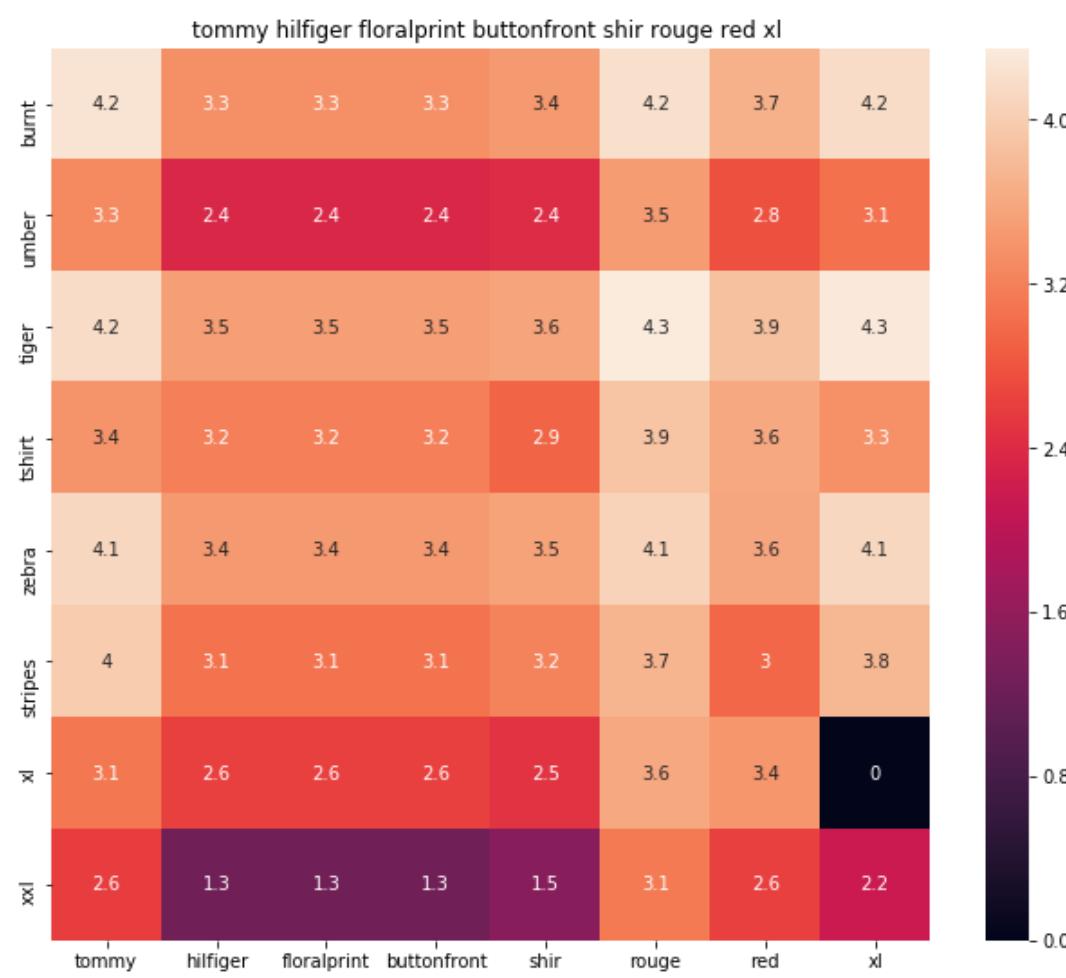
euclidean distance from given input image : 1.09006



ASIN : B01I53HU6K

BRAND : ouxili

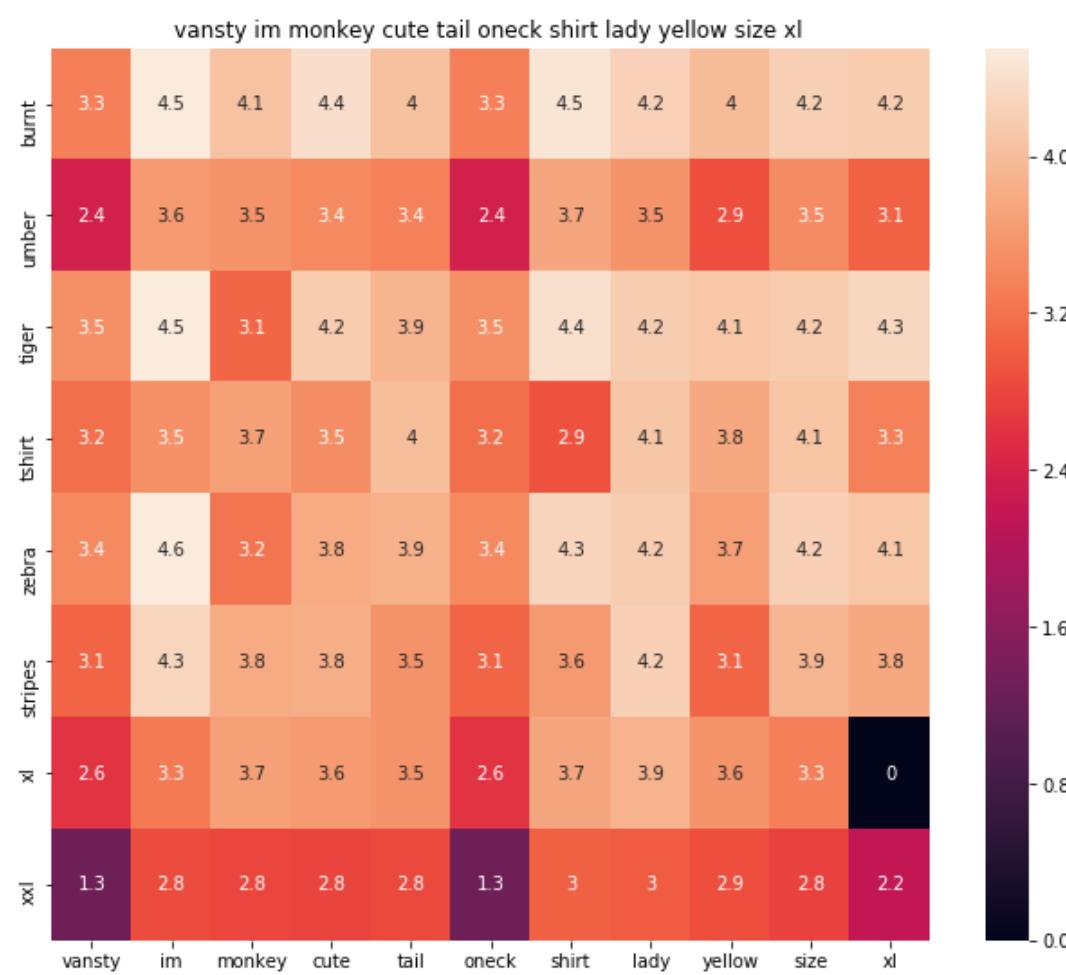
euclidean distance from given input image : 1.09201



ASIN : B0711NGTQM

BRAND : THILFIGER RTW

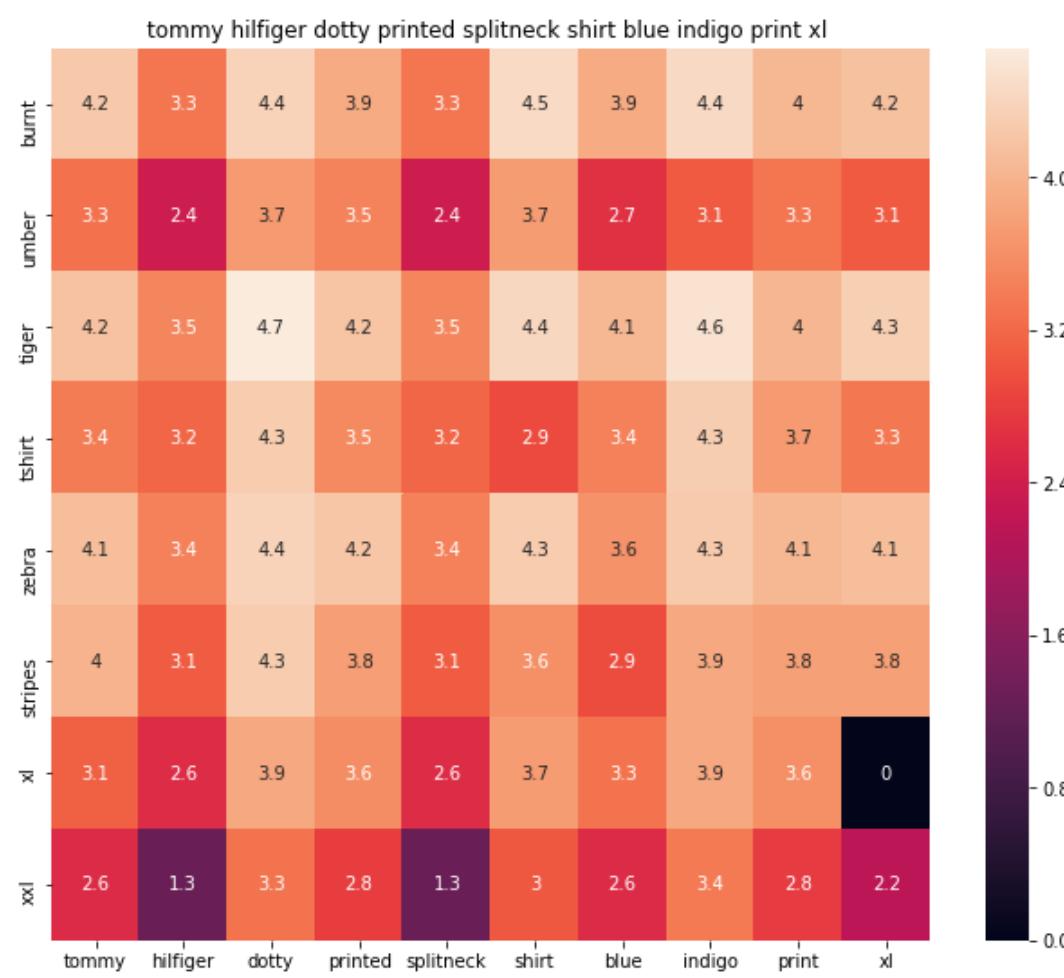
euclidean distance from given input image : 1.09234



ASIN : B01EFSLO8Y

BRAND : Vansty

euclidean distance from given input image : 1.0934



ASIN : B0716TVWQ4

BRAND : THILFIGER RTW

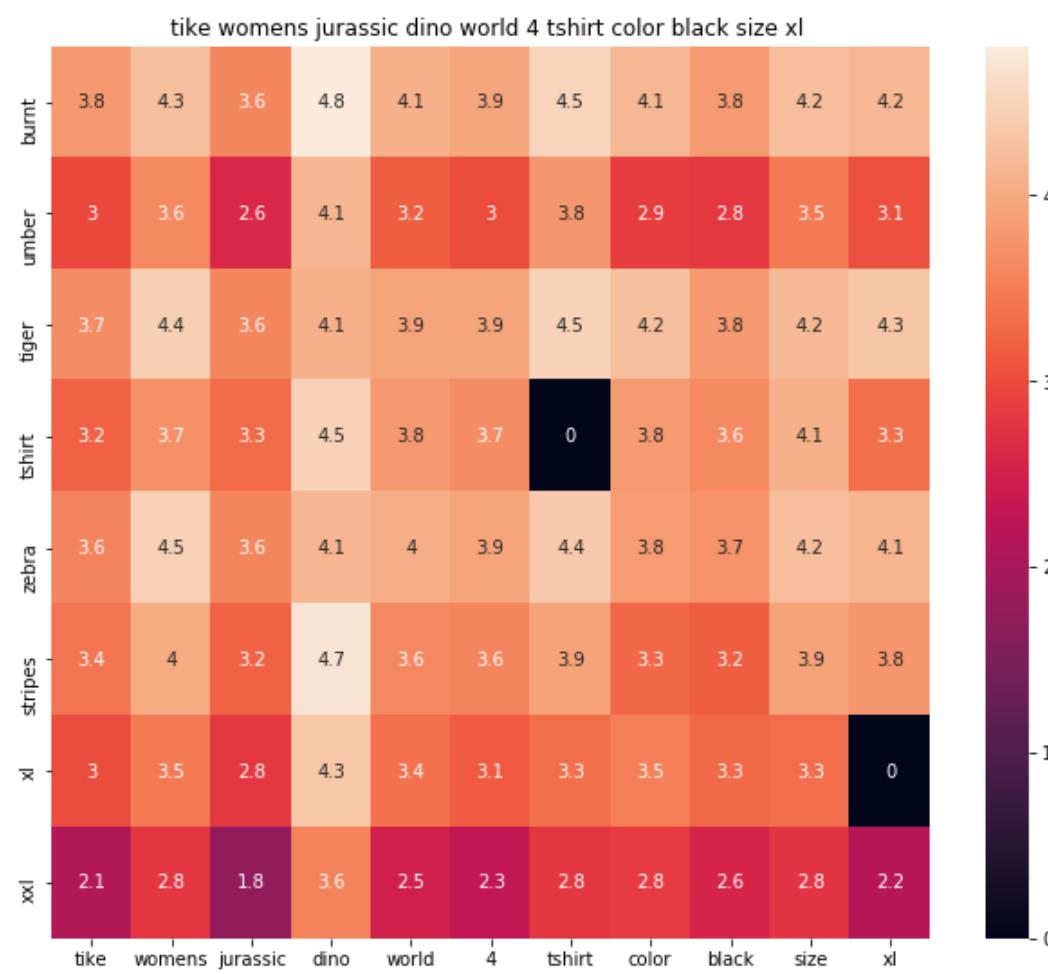
euclidean distance from given input image : 1.0942



ASIN : B0716MVPGV

BRAND : V.Secret

euclidean distance from given input image : 1.09483



ASIN : B0160PN40I

BRAND : TIKE Fashions

euclidean distance from given input image : 1.09513



ASIN : B018WDJCUA

BRAND : INC - International Concepts Woman

euclidean distance from given input image : 1.09669

[9.4] IDF weighted Word2Vec for product similarity

```
In [36]: 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 [37]: def weighted_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

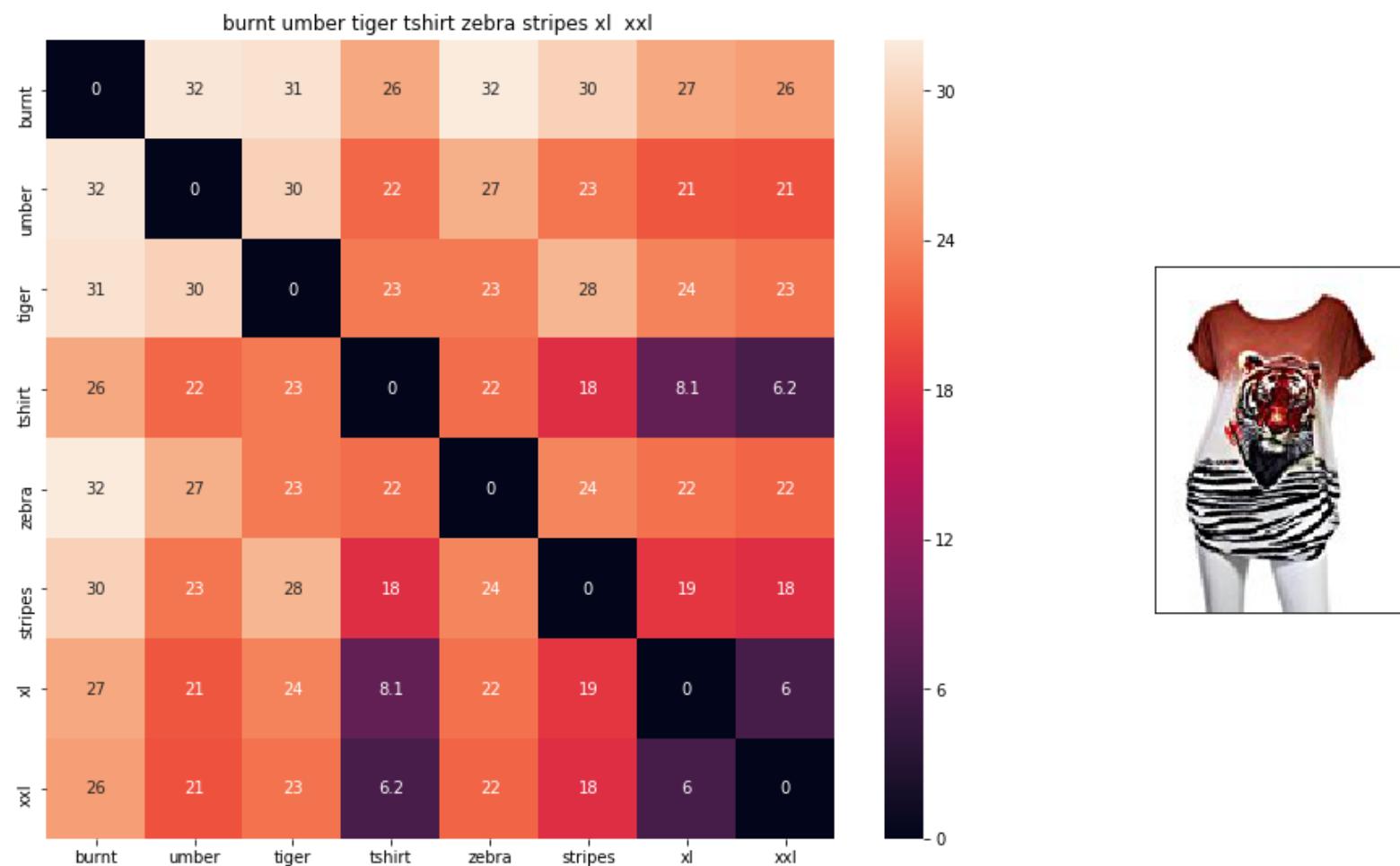
    # pairwise_dist will store the distance from given input apparel to all remaining apparels
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y> / (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = pairwise_distances(w2v_title_weight,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

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

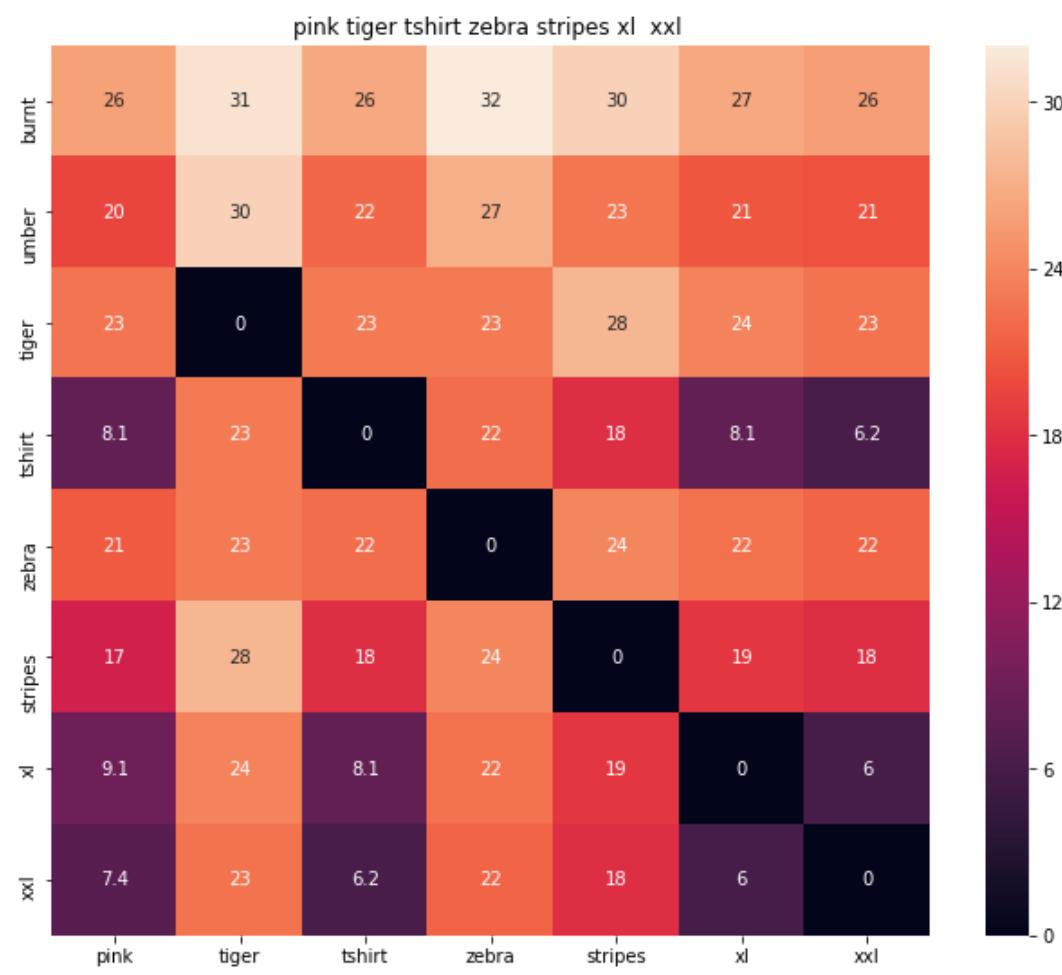
    #data frame indices of the 9 smallest distace's
    df_indices = list(data.index[indices])

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

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



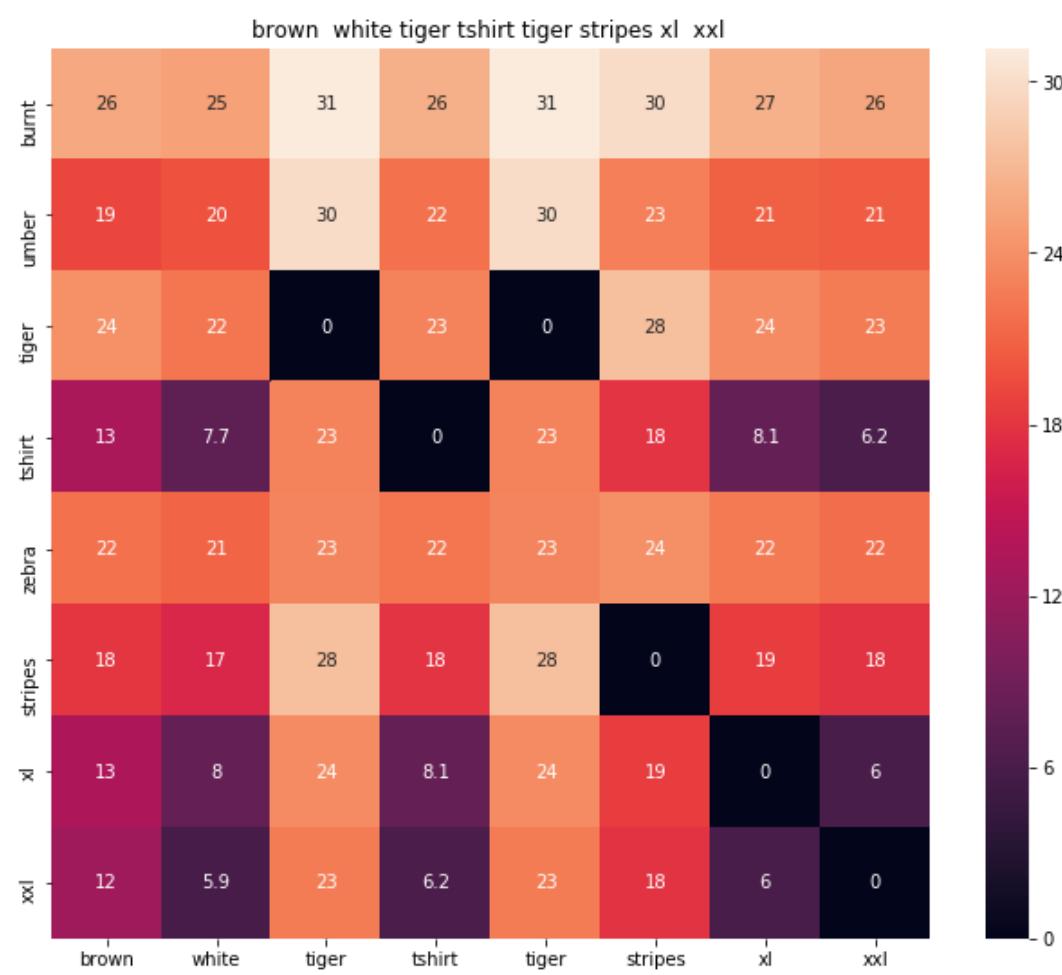
```
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.00390625
=====
=====
```



ASIN : B00JXQASS6

Brand : Si Row

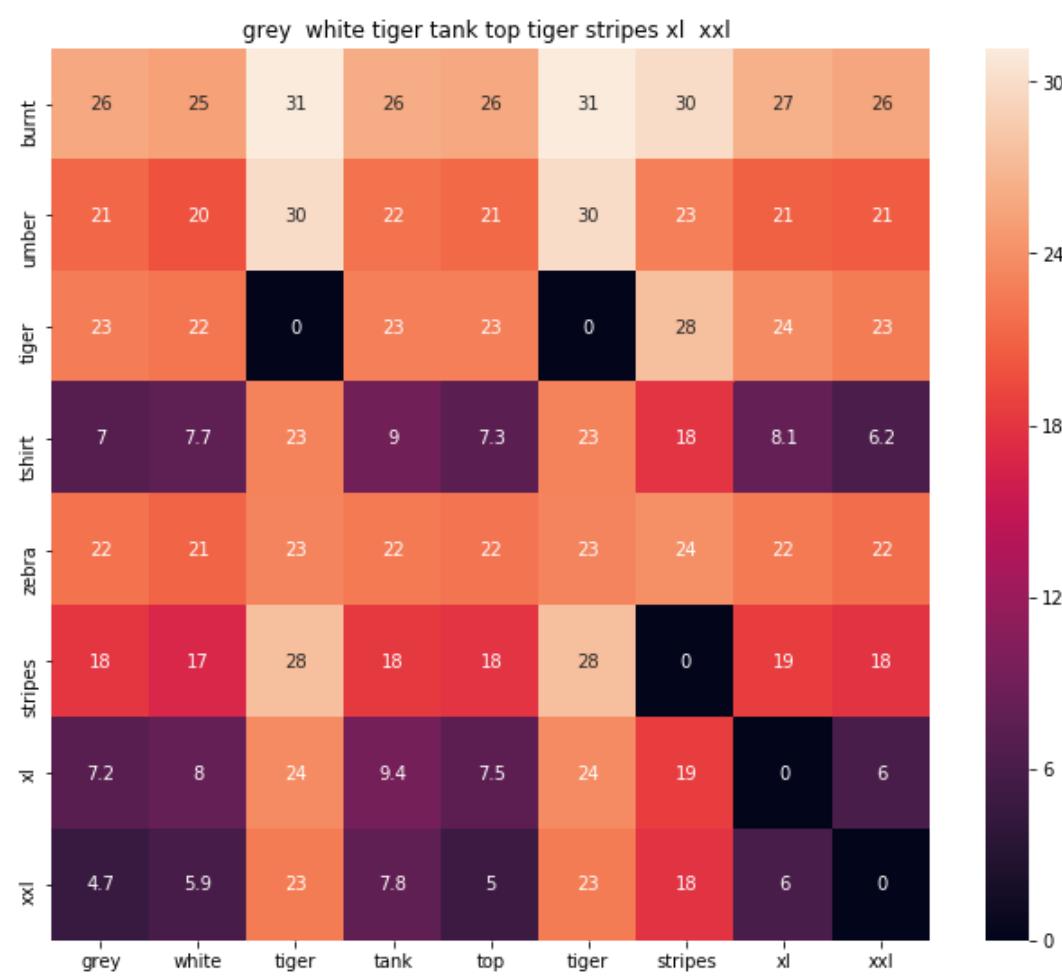
euclidean distance from input : 4.06389



ASIN : B00JXQCWT0

Brand : Si Row

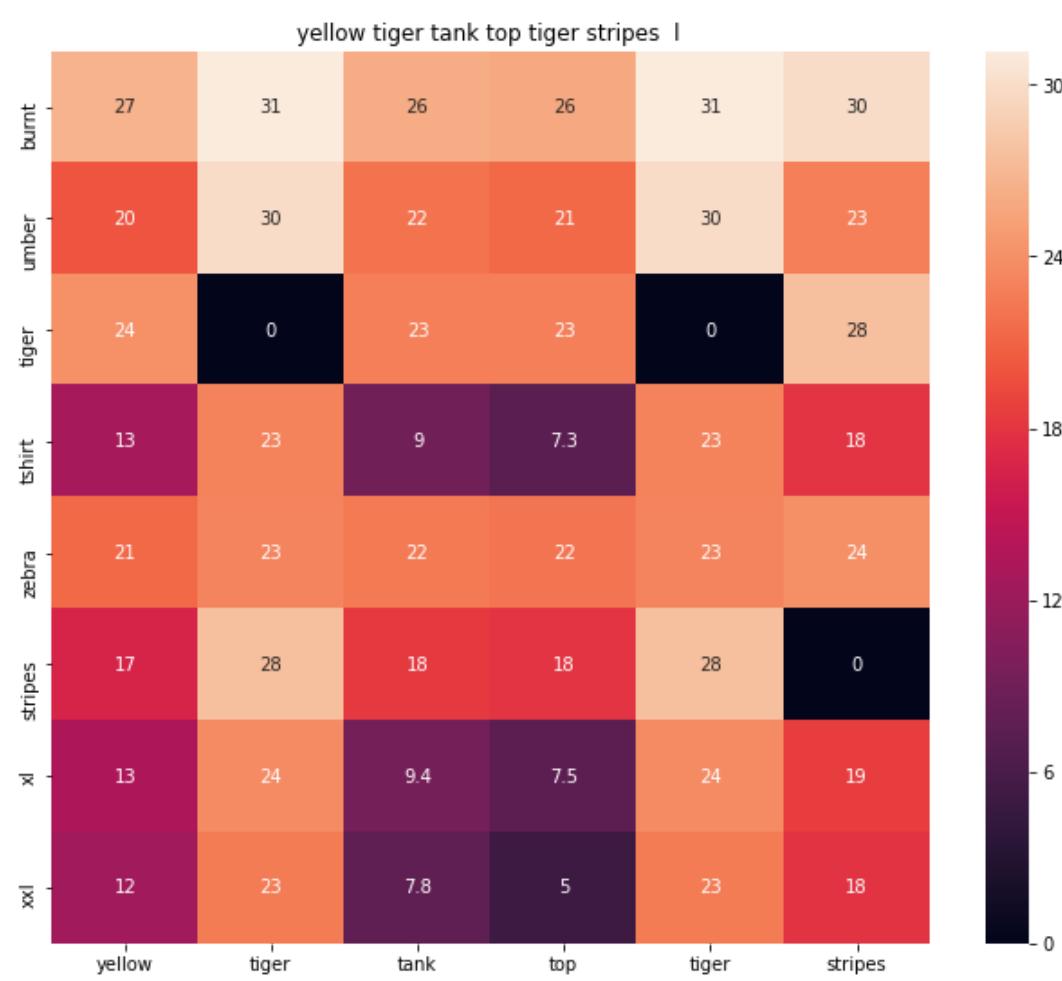
euclidean distance from input : 4.77094



ASIN : B00JXQAFZ2

Brand : Si Row

euclidean distance from input : 5.36016



ASIN : B00JXQAUWA

Brand : Si Row

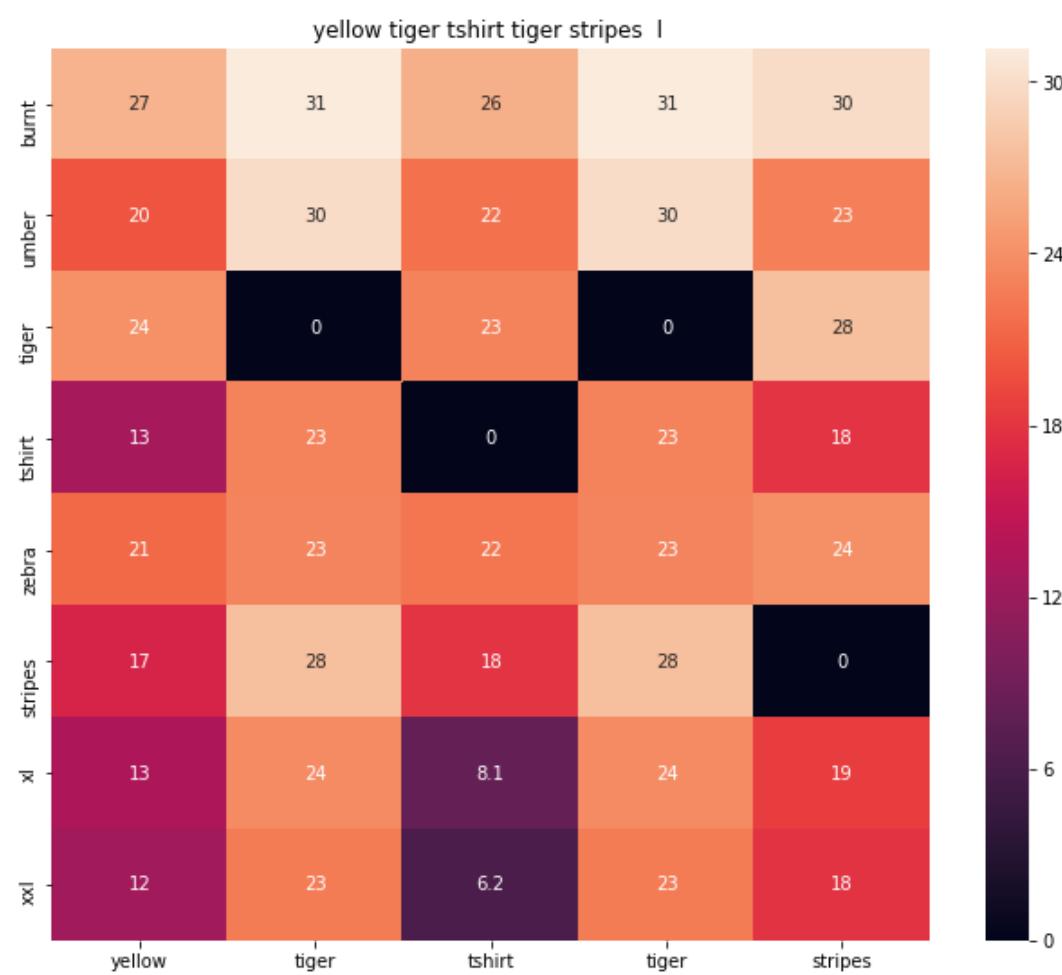
euclidean distance from input : 5.68952



ASIN : B00JXQA094

Brand : Si Row

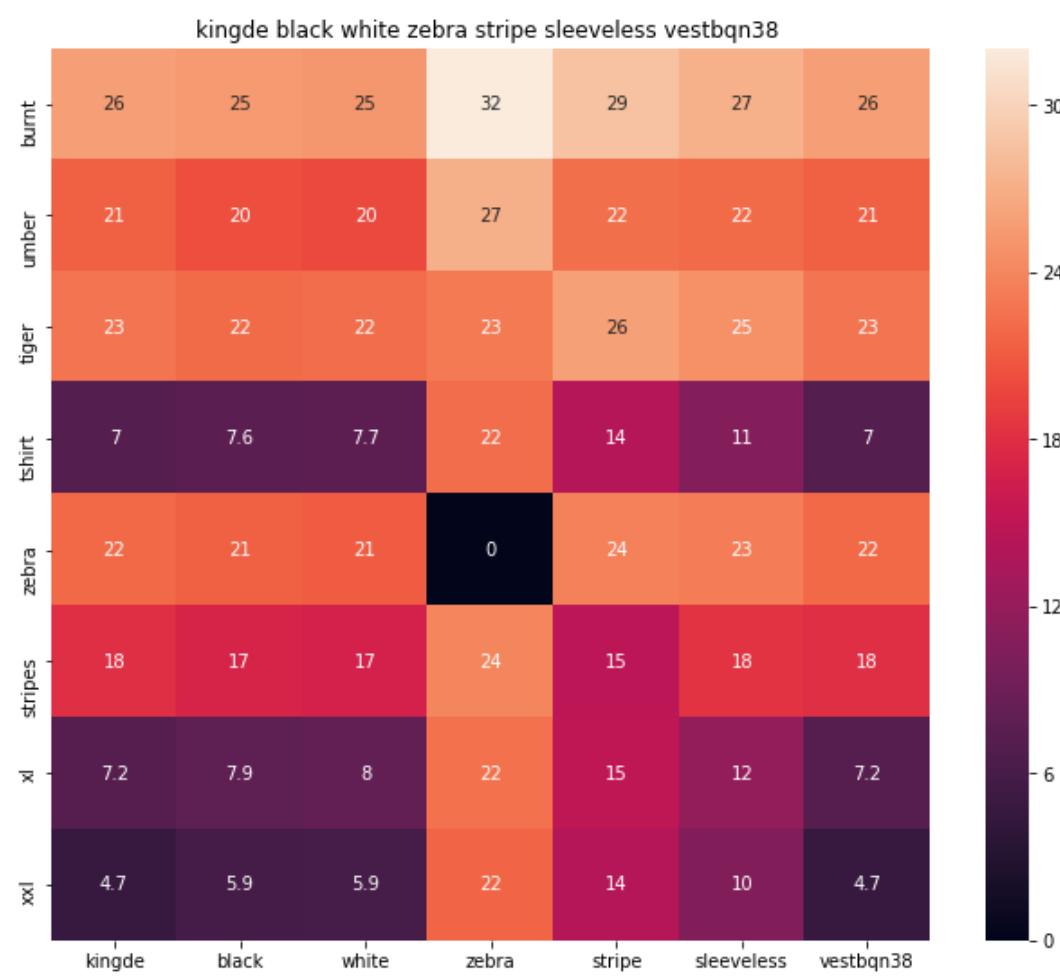
euclidean distance from input : 5.69302



ASIN : B00JXQCUIC

Brand : Si Row

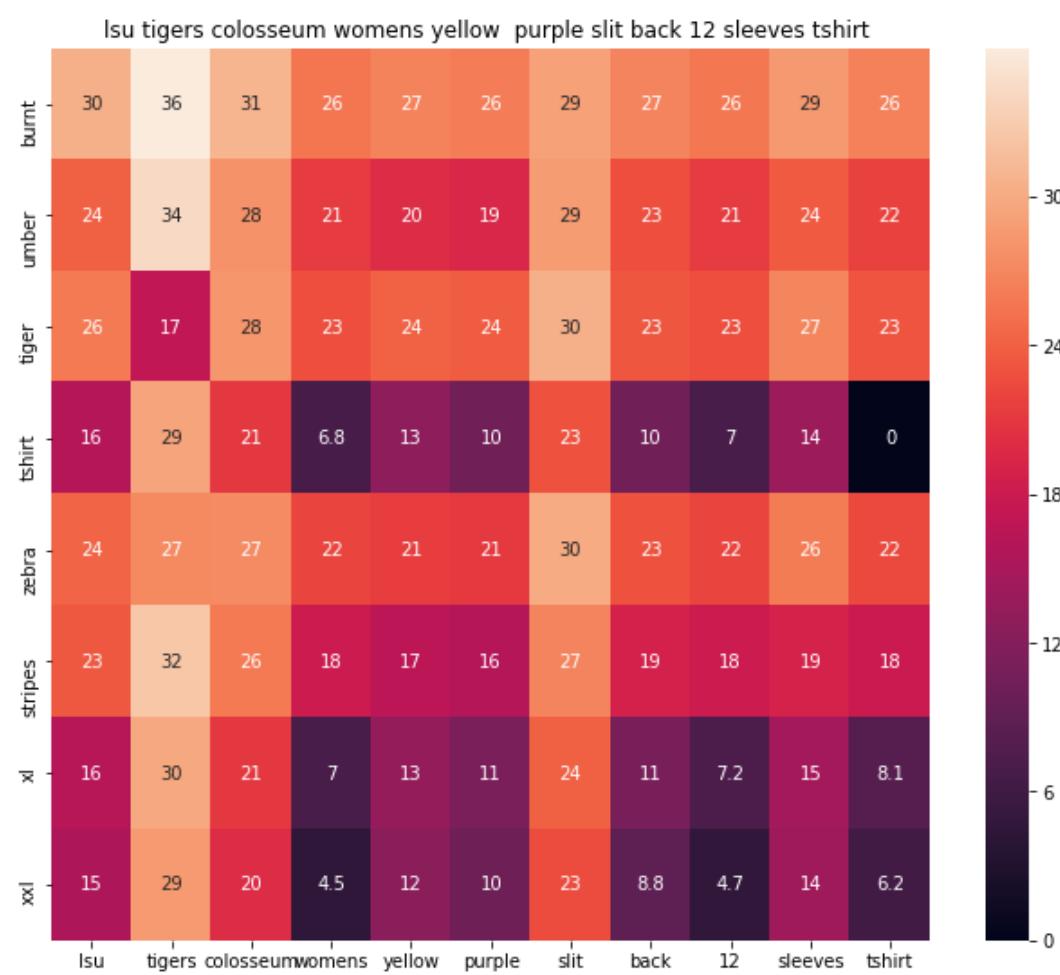
euclidean distance from input : 5.89344



ASIN : B015H41F6G

Brand : KINGDE

euclidean distance from input : 6.13299



ASIN : B073R5Q8HD

Brand : Colosseum

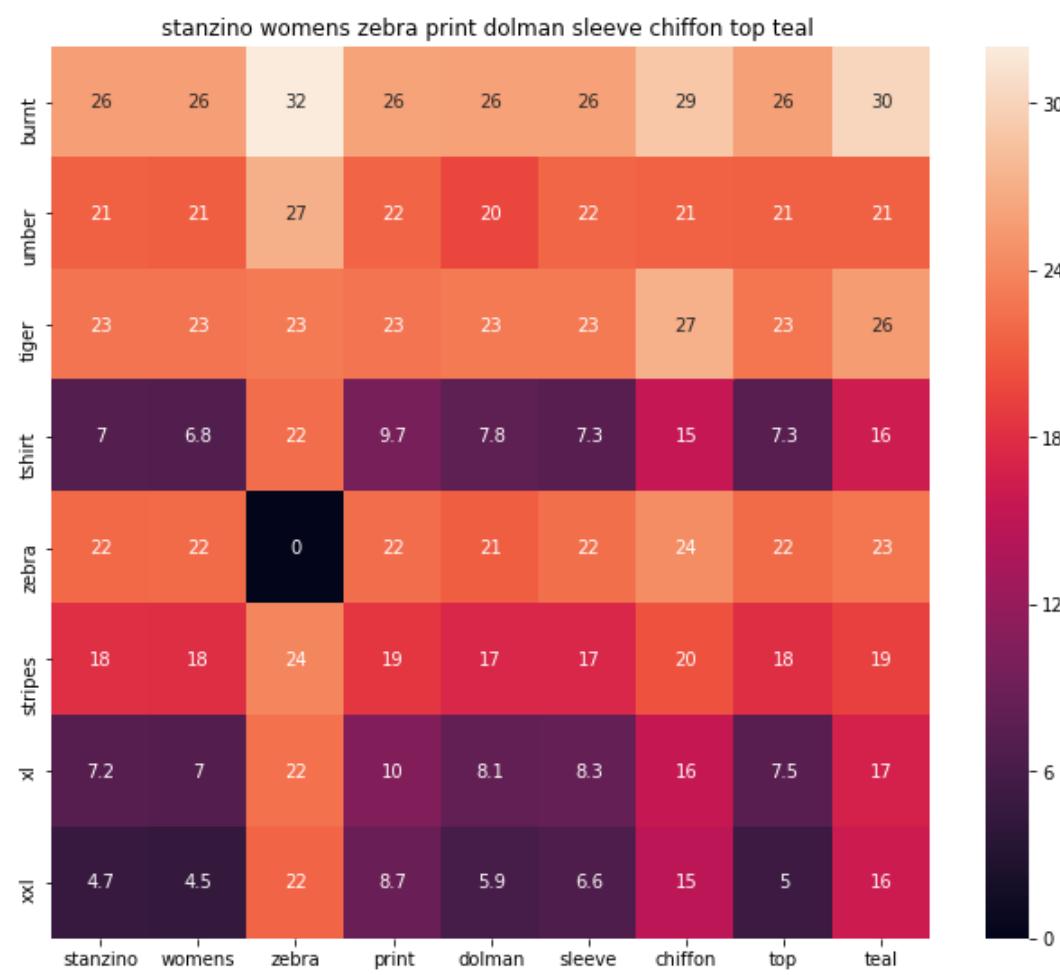
euclidean distance from input : 6.25671



ASIN : B074P8MD22

Brand : Edista

euclidean distance from input : 6.3922



ASIN : B00C0I3U3E

Brand : Stanzino

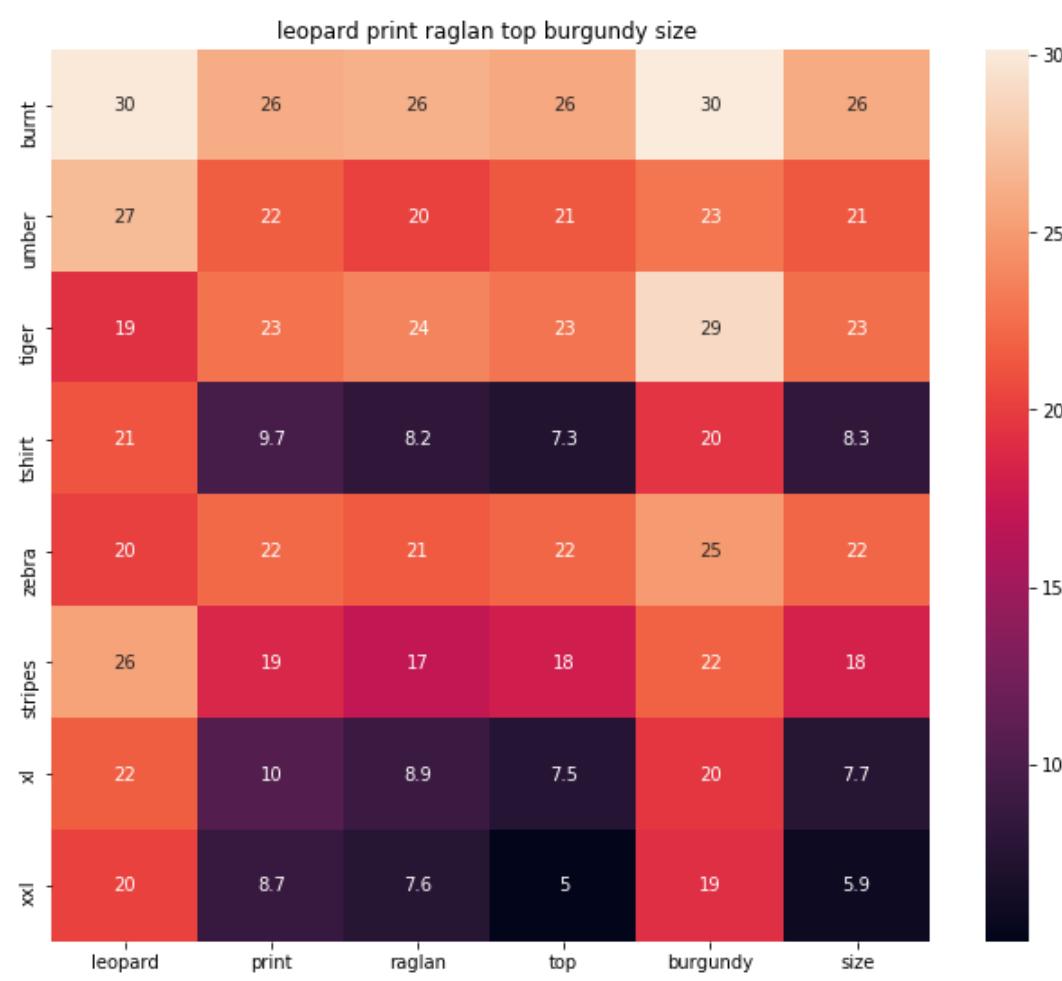
euclidean distance from input : 6.4149



ASIN : B073R4ZM7Y

Brand : Colosseum

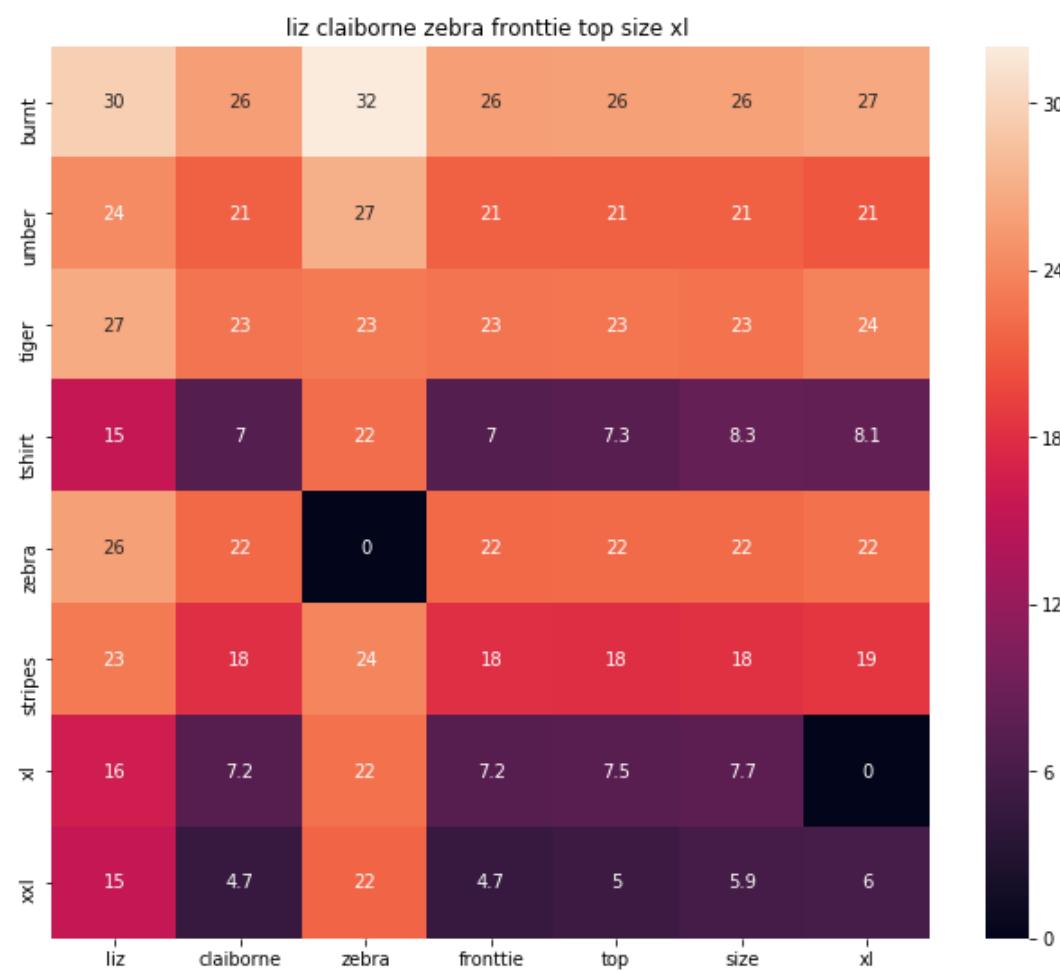
euclidean distance from input : 6.45096



ASIN : B01C60RLDQ

Brand : 1 Mad Fit

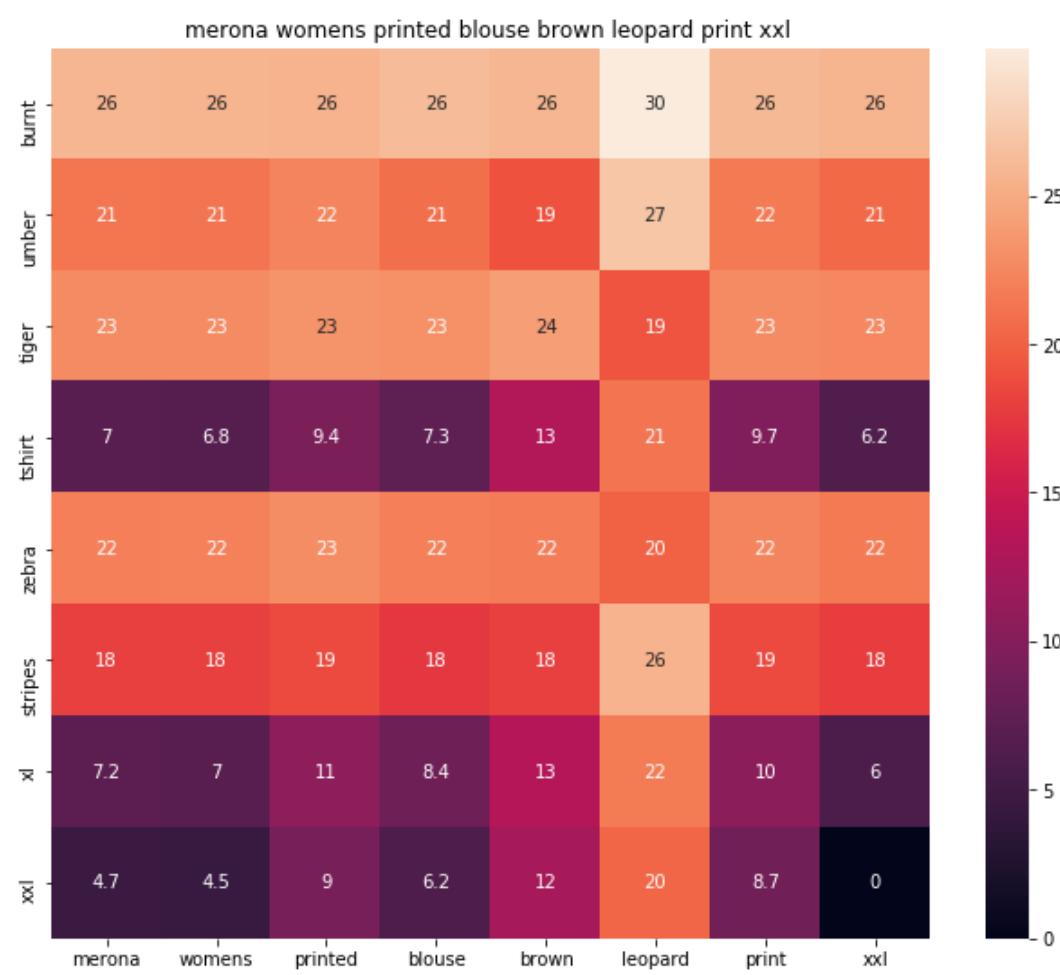
euclidean distance from input : 6.46341



ASIN : B06XBY5QXL

Brand : Liz Claiborne

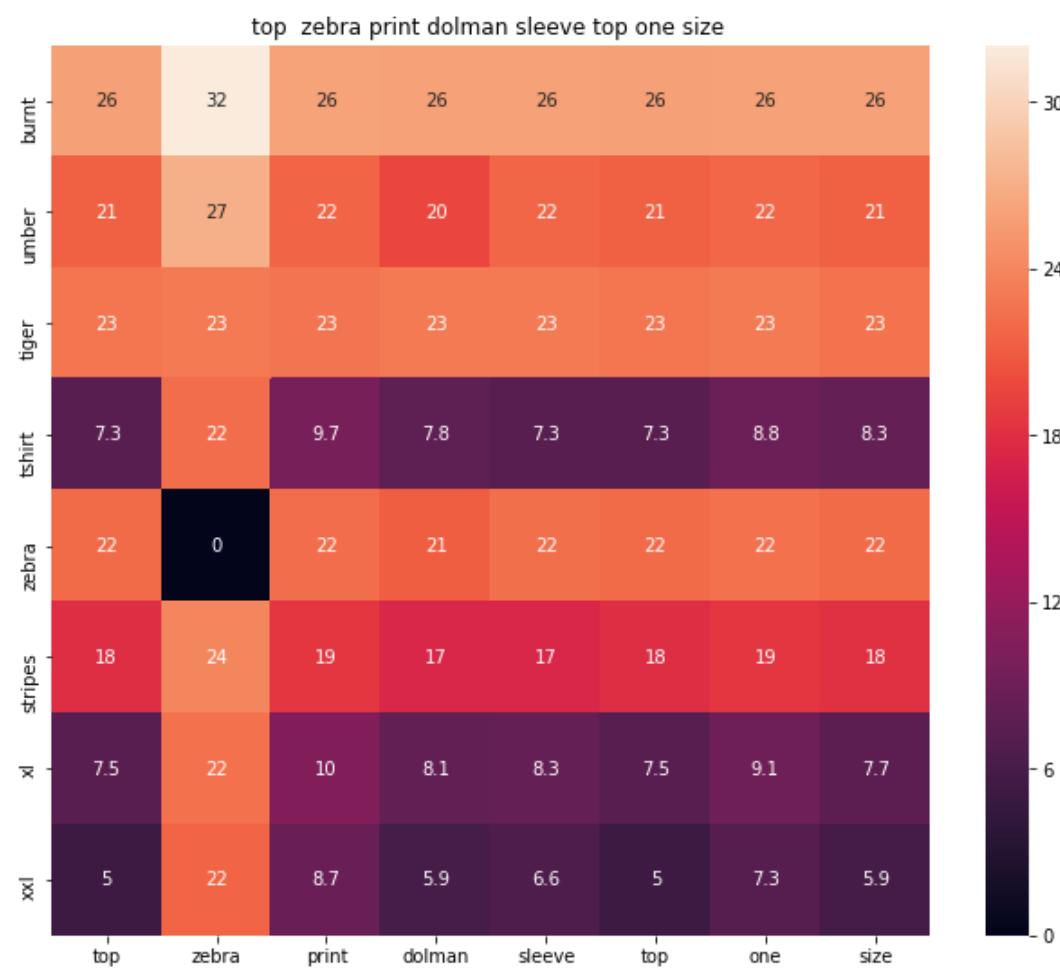
euclidean distance from input : 6.53922



ASIN : B071YF3WDD

Brand : Merona

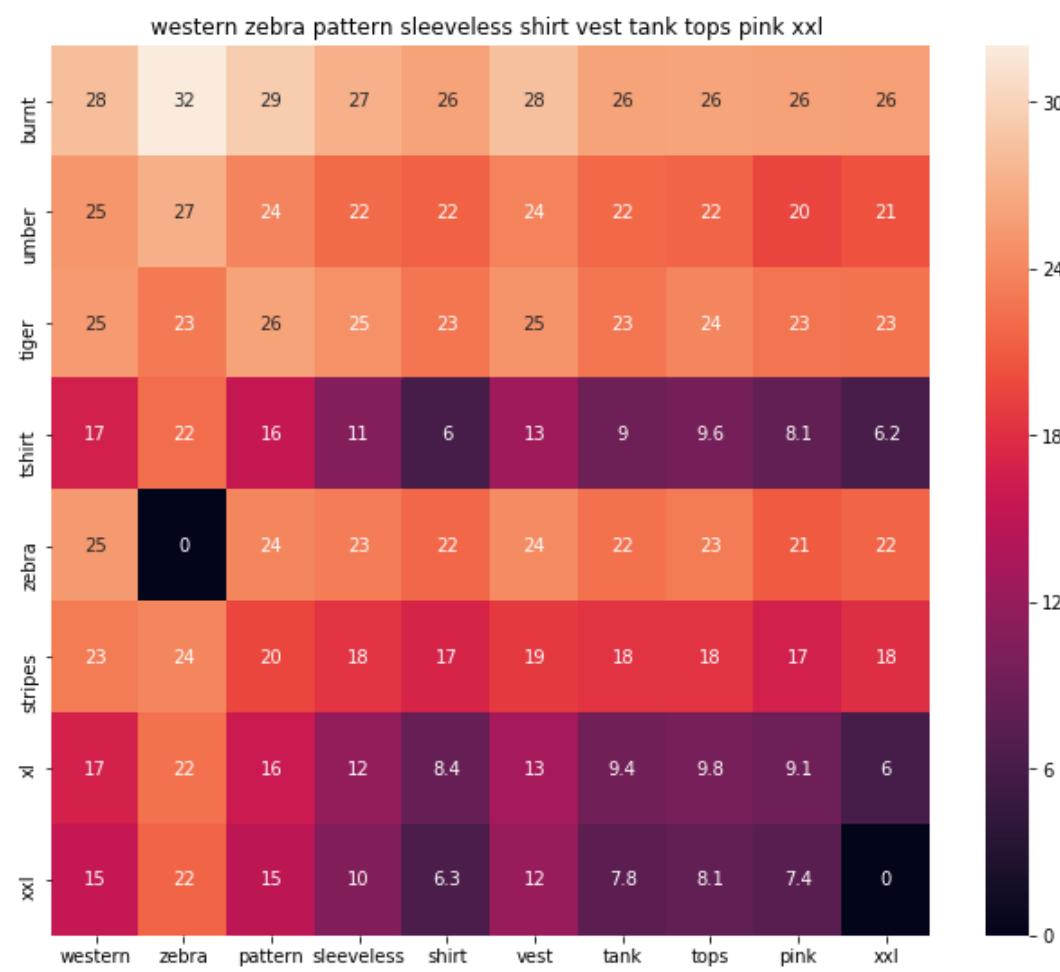
euclidean distance from input : 6.5755



ASIN : B00H8A6ZLI

Brand : Vivian's Fashions

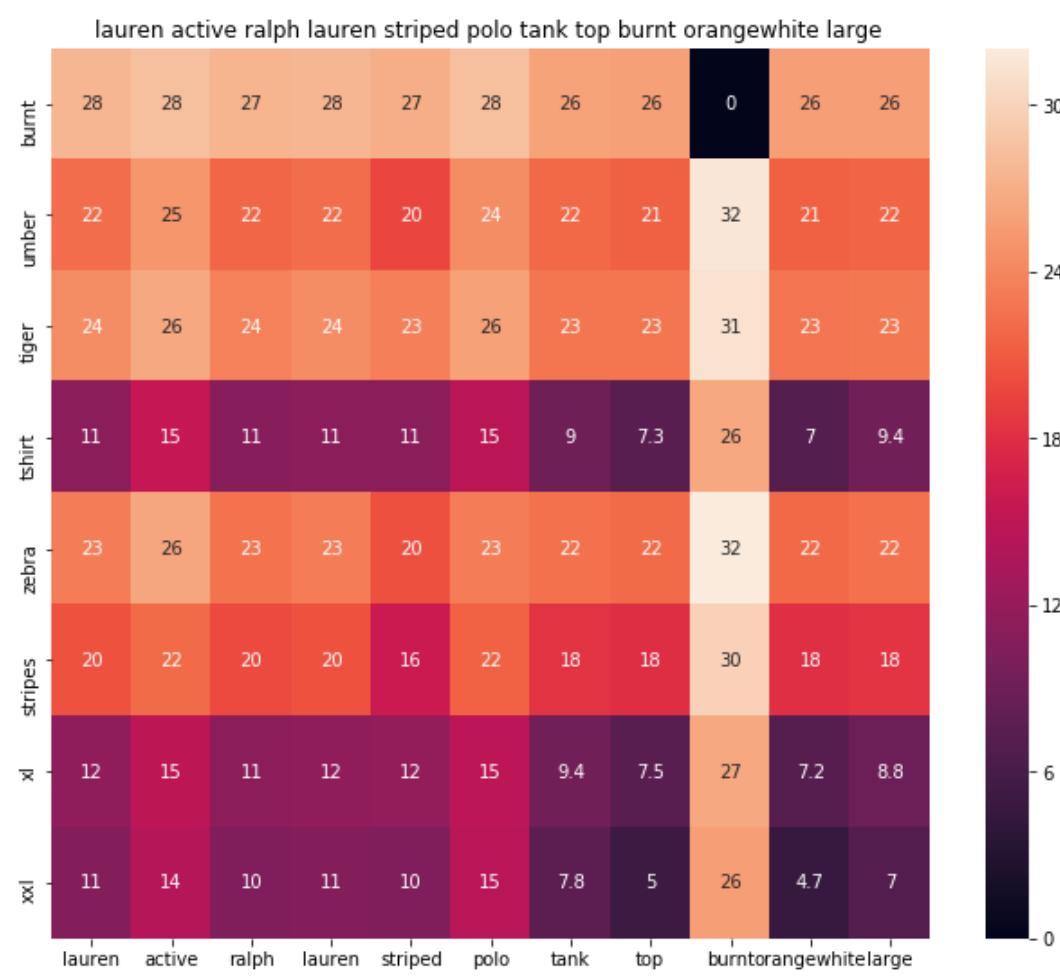
euclidean distance from input : 6.63821



ASIN : B00Z6HEXWI

Brand : Black Temptation

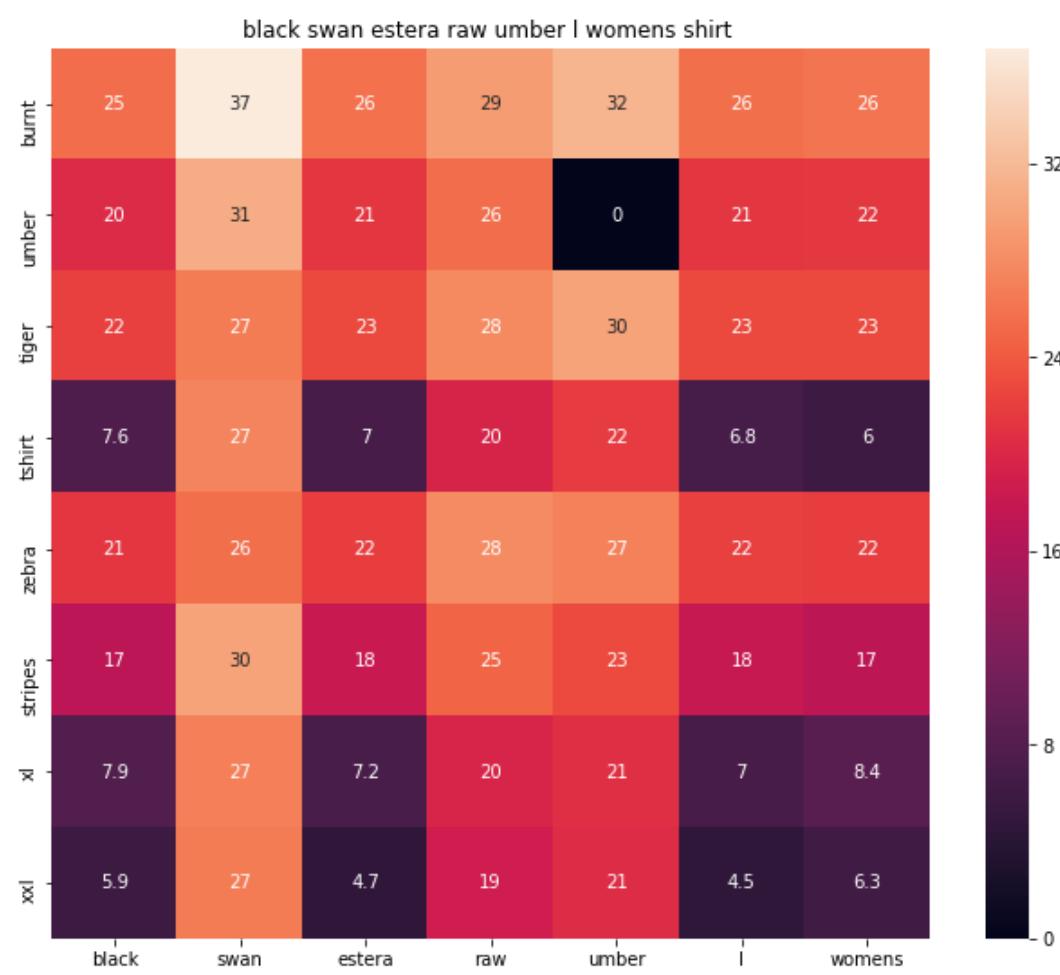
euclidean distance from input : 6.66074



ASIN : B00IILGH50Y

Brand : Ralph Lauren Active

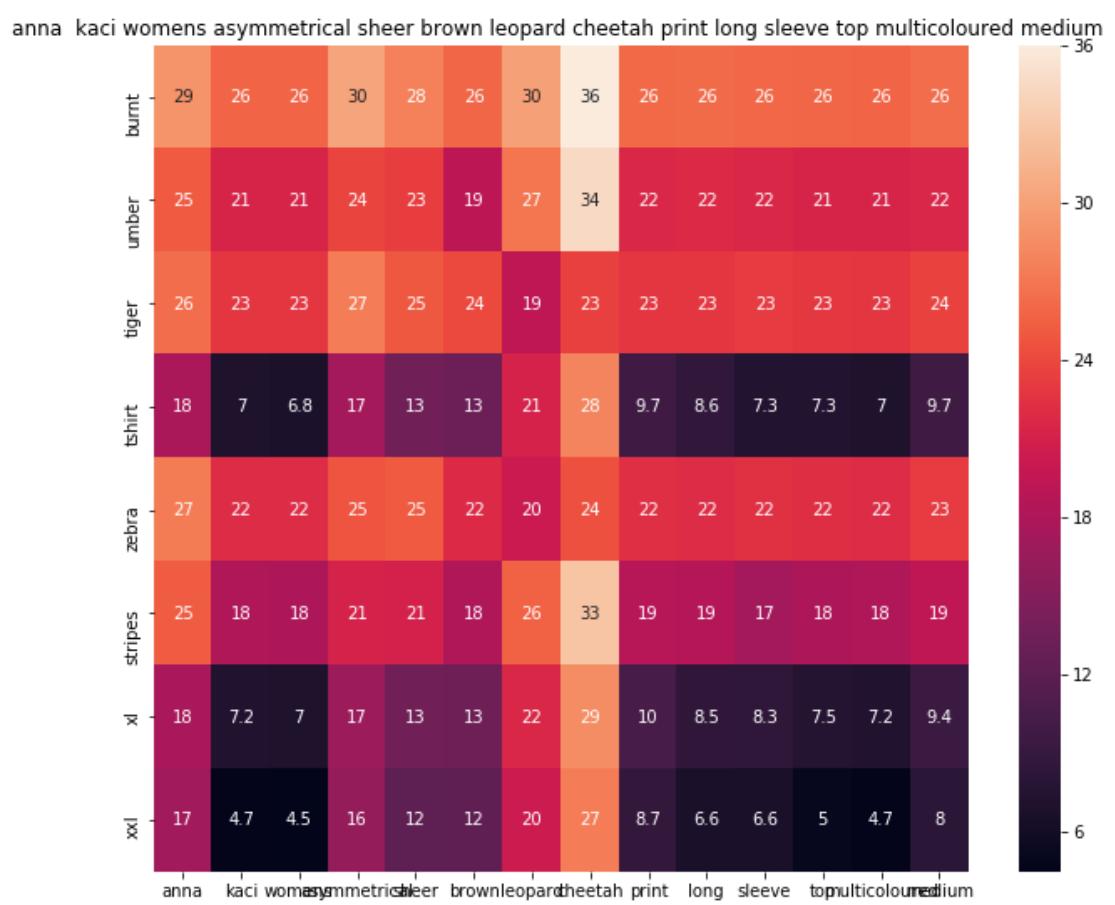
euclidean distance from input : 6.68391



ASIN : B06Y1VN8WQ

Brand : Black Swan

euclidean distance from input : 6.70576



ASIN : B00KSNTY7Y
 Brand : Anna-Kaci
 euclidean distance from input : 6.70612

=====

[9.6] Weighted similarity using brand and color.

```
In [38]: # some of the brand values are empty.
# Need to replace Null with string "NULL"
data['brand'].fillna(value="Not given", inplace=True)

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

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

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

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

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

```
In [39]: def heat_map_w2v_brand(sentance1, sentance2, url, doc_id1, doc_id2, df_id1, df_id2, model):

    # sentance1 : title1, input apparel
    # sentance2 : title2, recommended apparel
    # url: apparel image url
    # doc_id1: document id of input apparel
    # doc_id2: document id of recommended apparel
    # df_id1: index of document1 in the data frame
    # df_id2: index of document2 in the data frame
    # model: it can have two values, 1. avg 2. weighted

    #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/avg) of length 30
    # corresponds to each word in give title
    s1_vec = get_word_vec(sentance1, doc_id1, model)
    #s2_vec = np.array(#number_of_words_title2 * 300), each row is a vector(weighted/avg) of length 30
    # corresponds to each word in give title
    s2_vec = get_word_vec(sentance2, doc_id2, model)

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

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

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

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

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

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

    # in last 25 * 10:15 grids we display image
    ax2 = plt.subplot(gs[:, 10:16])
    # we dont display grid 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 [41]: def idf_w2v_brand(doc_id, w1, w2, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for w2v features
    # w2: weight for brand and color features

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

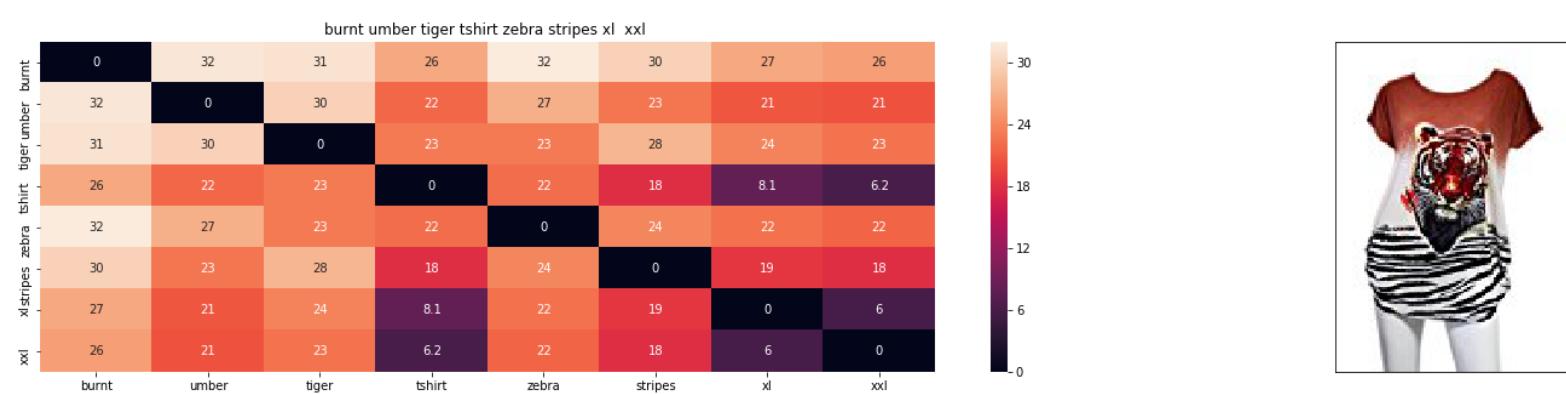
    # np.argsort will return indices of 9 smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists = np.sort(pairwise_dist.flatten())[0:num_results]

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

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

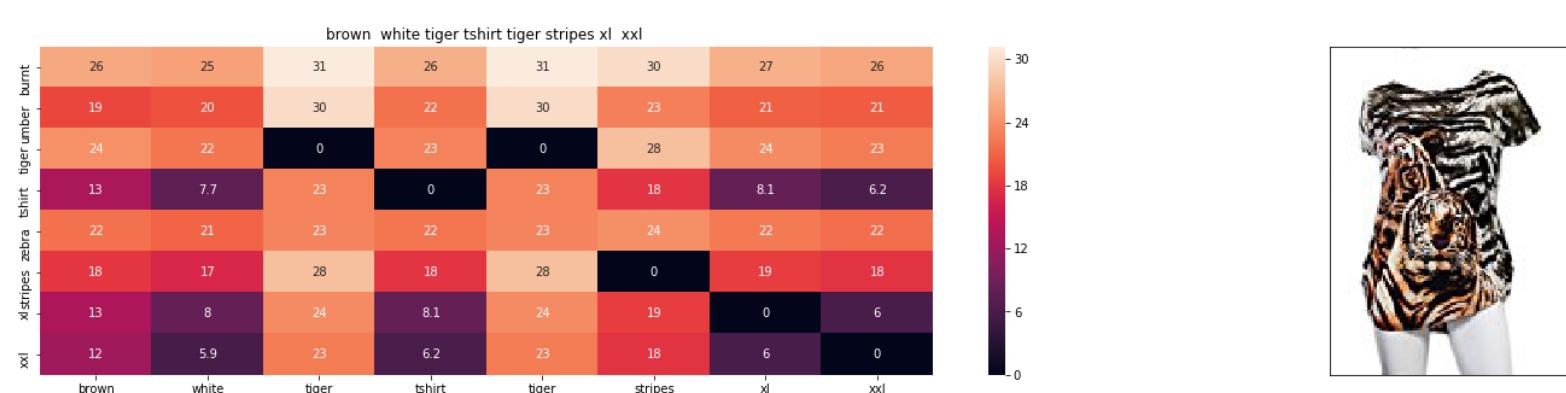
idf_w2v_brand(12566, 5, 5, 20)
# in the give heat map, each cell contains the euclidean distance between words i, j
```

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES Export to plot.ly »



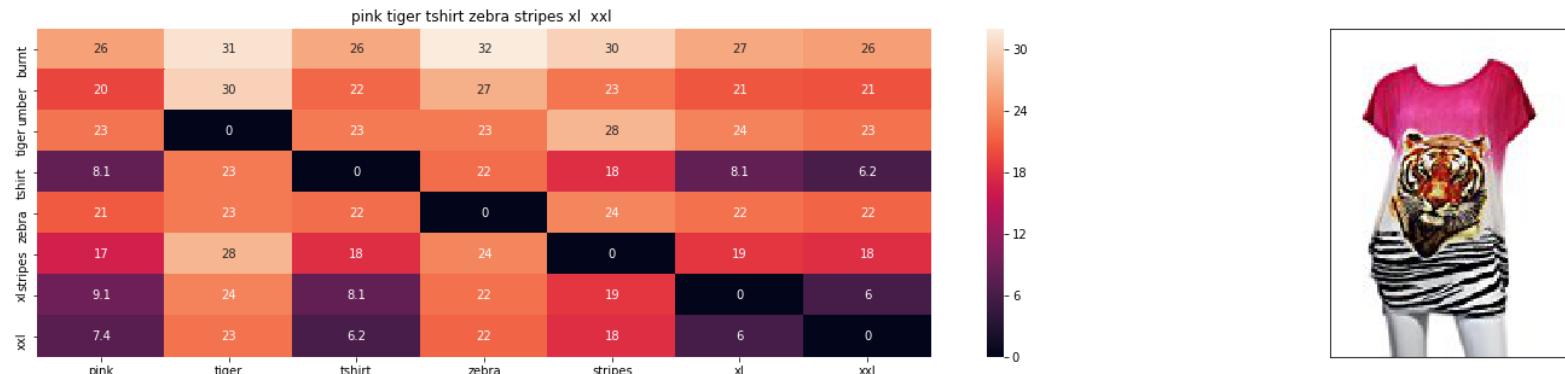
ASIN : B00JXQB5FQ
 Brand : Si Row
 euclidean distance from input : 0.001953125

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQCWTO	Si-Row	Brown	TOYS_AND_GAMES Export to plot.ly »



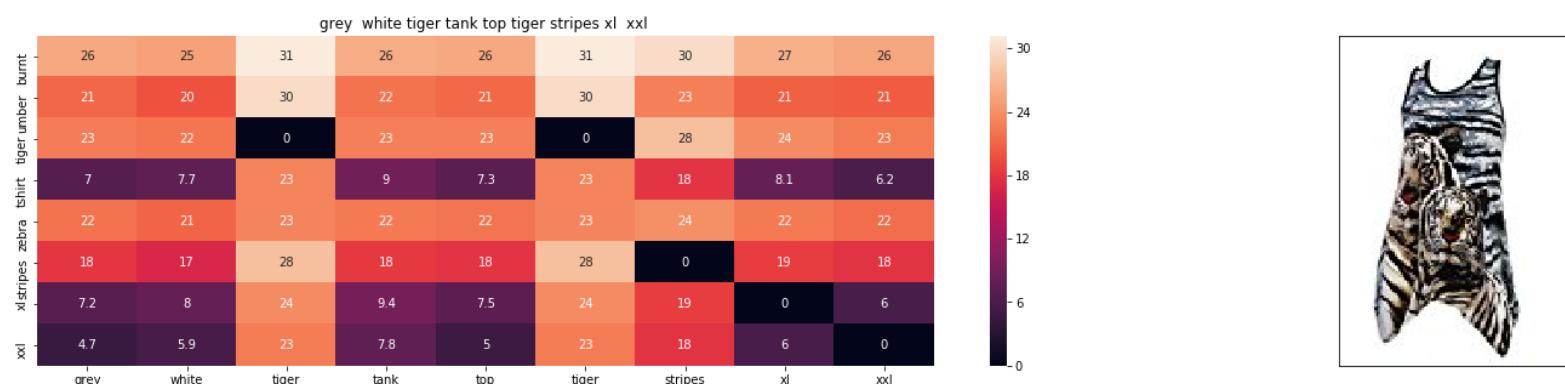
ASIN : B00JXQCWTO
 Brand : Si Row
 euclidean distance from input : 2.38547115326

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQASS6	Si-Row	Pink	TOYS_AND_GAMES Export to plot.ly »



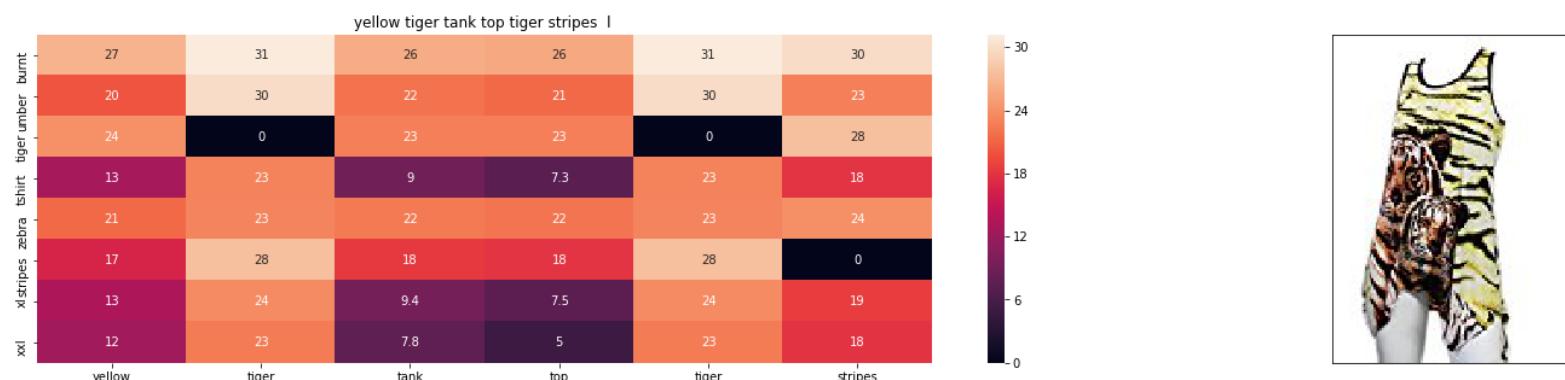
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 2.73905105609
=====

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAFZ2	Si-Row	Grey	TOYS_AND_GAMES Export to plot.ly »



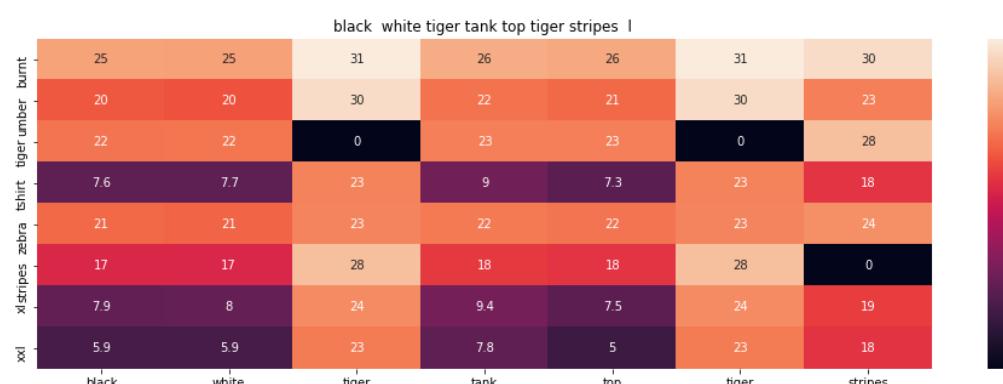
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 3.387187195
=====

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAUWA	Si-Row	Yellow	TOYS_AND_GAMES Export to plot.ly »



ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 3.5518684389
=====

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAO94	Si-Row	White	TOYS_AND_GAMES Export to plot.ly »

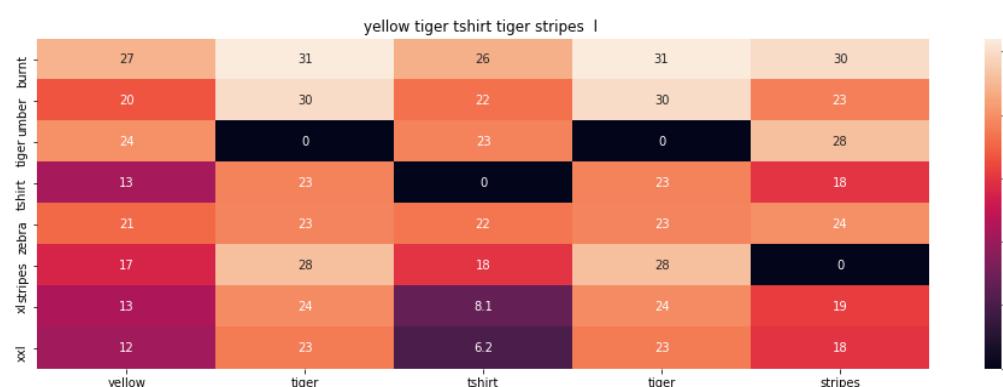


ASIN : B00JXQAO94

Brand : Si Row

euclidean distance from input : 3.5536174776

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQCUIC	Si-Row	Yellow	TOYS_AND_GAMES Export to plot.ly »

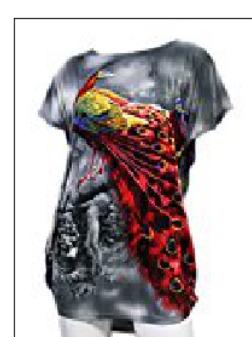
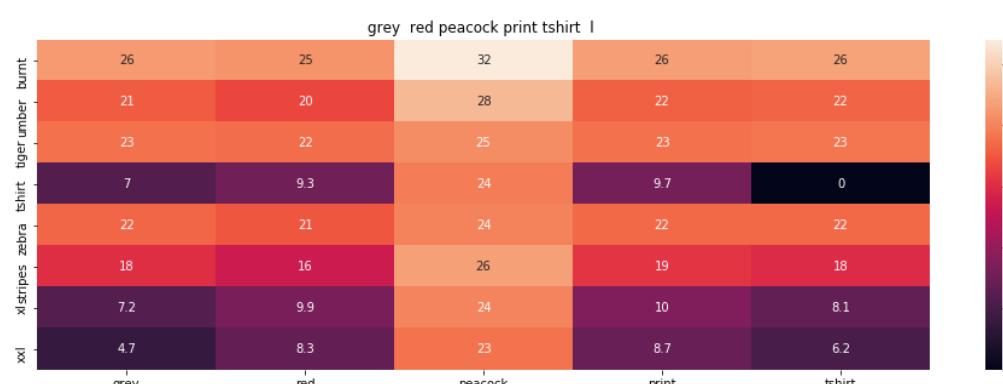


ASIN : B00JXQCUIC

Brand : Si Row

euclidean distance from input : 3.65382804889

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQCFRS	Si-Row	Grey	TOYS_AND_GAMES Export to plot.ly »

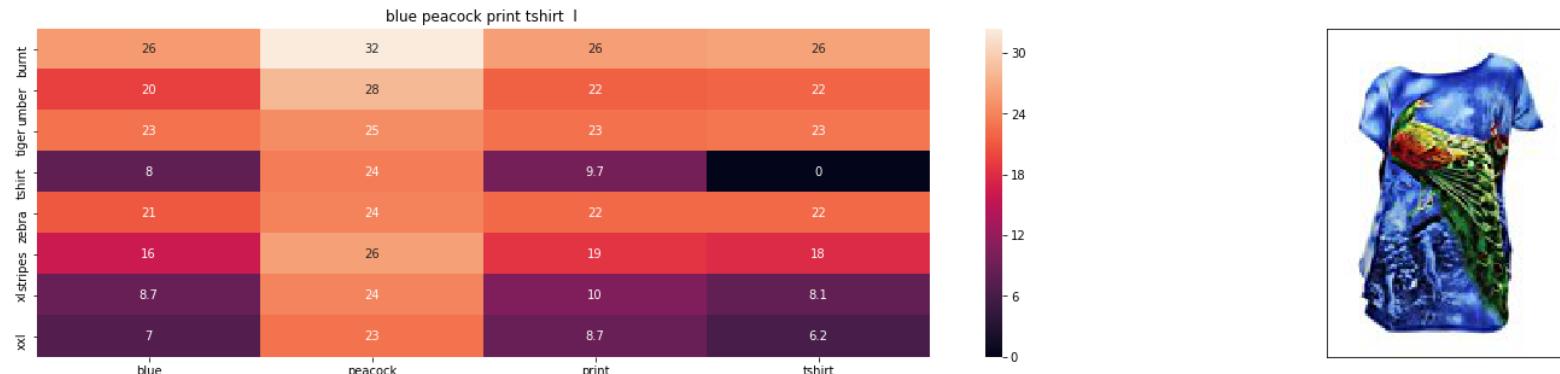


ASIN : B00JXQCFRS

Brand : Si Row

euclidean distance from input : 4.12881164569

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQC8L6	Si-Row	Blue	TOYS_AND_GAMES Export to plot.ly »

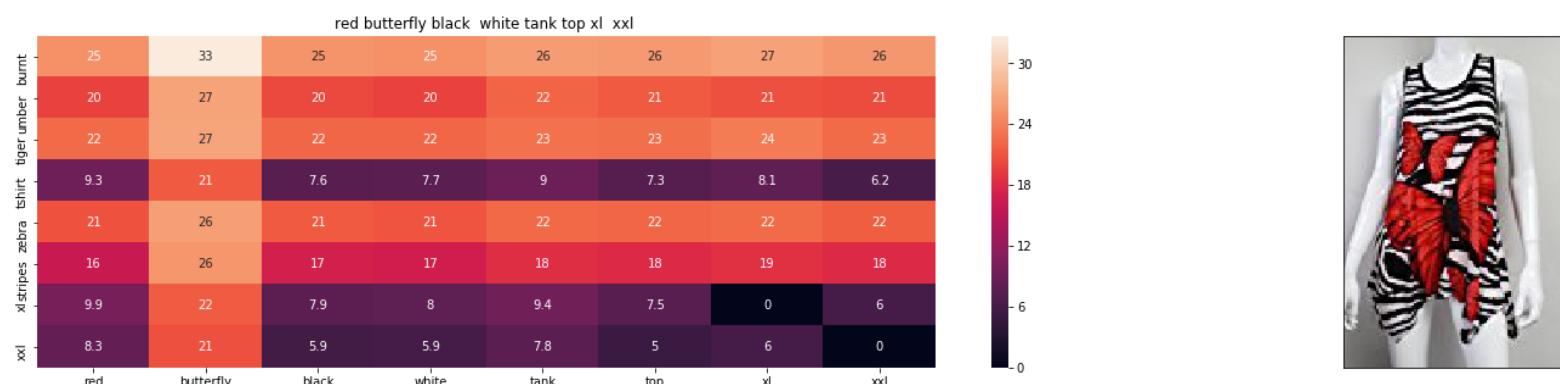


ASIN : B00JXQC8L6

Brand : Si Row

euclidean distance from input : 4.20390052813

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JV63CW2	Si-Row	Red	TOYS_AND_GAMES Export to plot.ly »

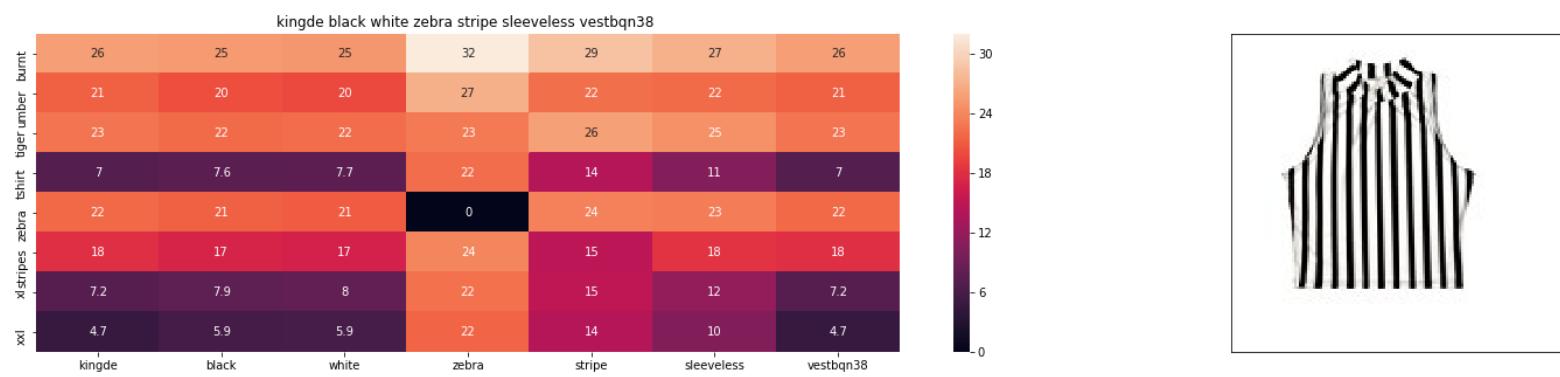


ASIN : B00JV63CW2

Brand : Si Row

euclidean distance from input : 4.28658676166

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B015H41F6G	KINGDE	White	SHIRT Export to plot.ly »

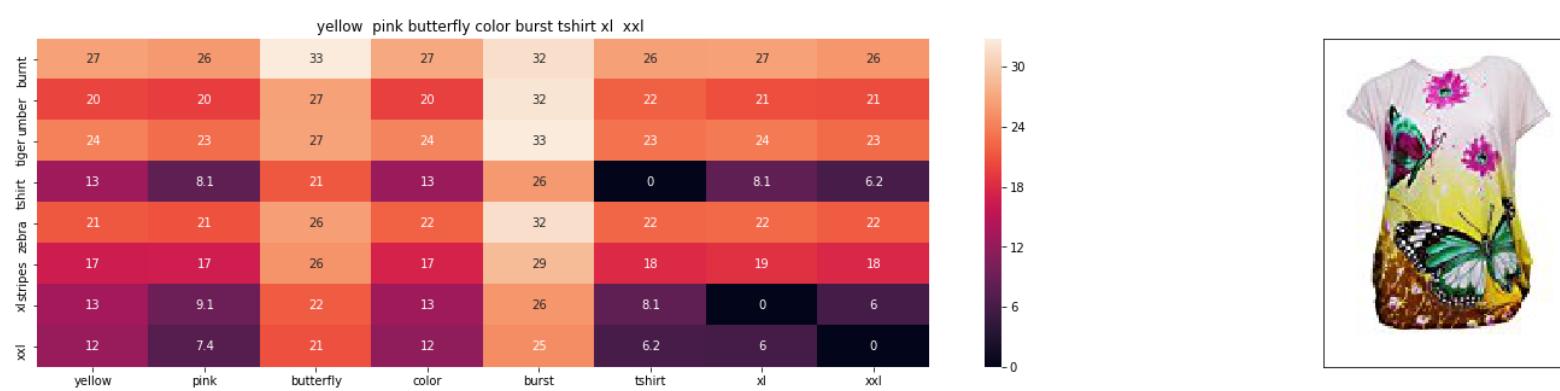


ASIN : B015H41F6G

Brand : KINGDE

euclidean distance from input : 4.38937078798

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQBBMI	Si-Row	Yellow	TOYS_AND_GAMES Export to plot.ly »

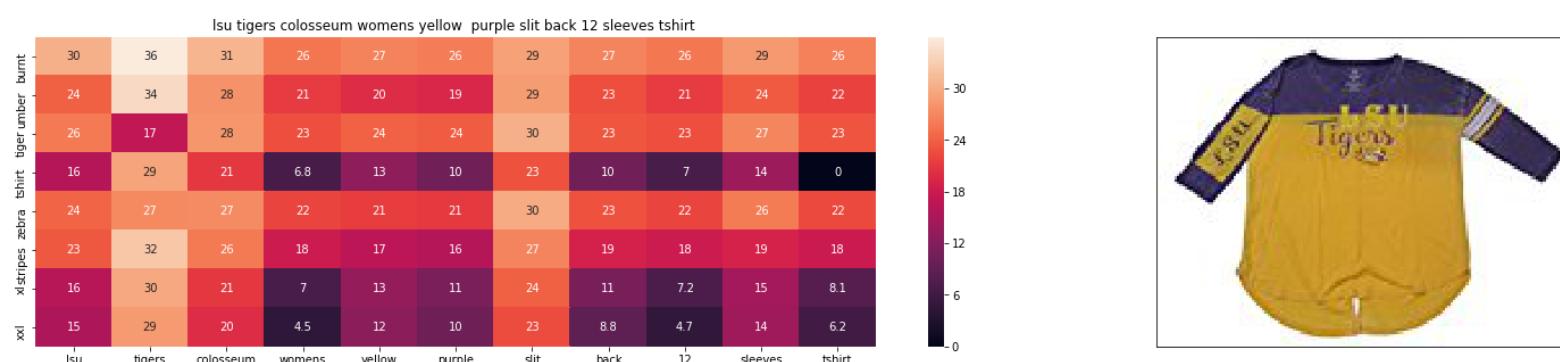


ASIN : B00JXQBBMI

Brand : Si Row

euclidean distance from input : 4.39790992755

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B073R5Q8HD	Colosseum	Yellow	SPORTING_GOODS Export to plot.ly »

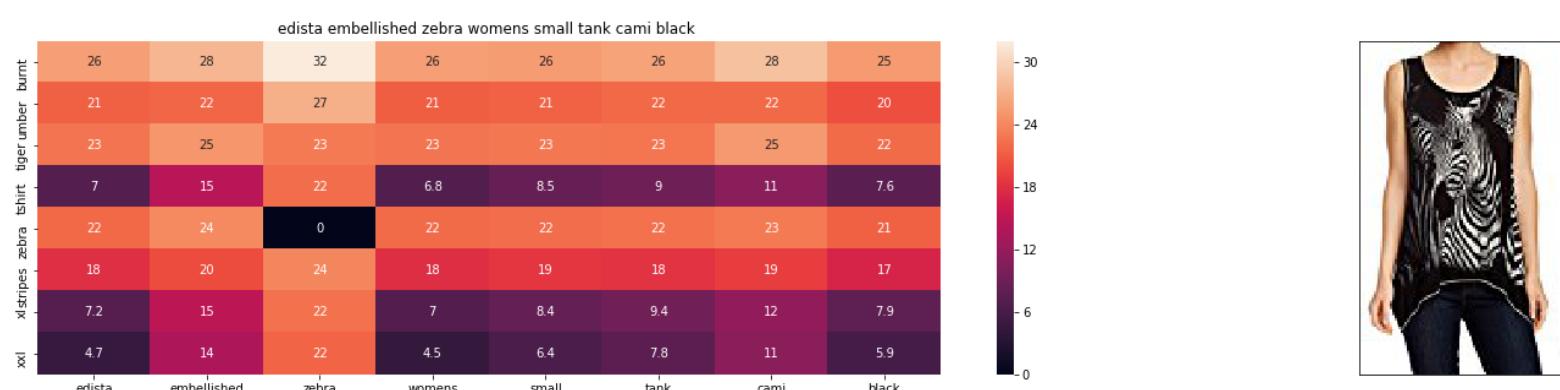


ASIN : B073R5Q8HD

Brand : Colosseum

euclidean distance from input : 4.45122858369

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B074P8MD22	Edista	Black	SHIRT Export to plot.ly »

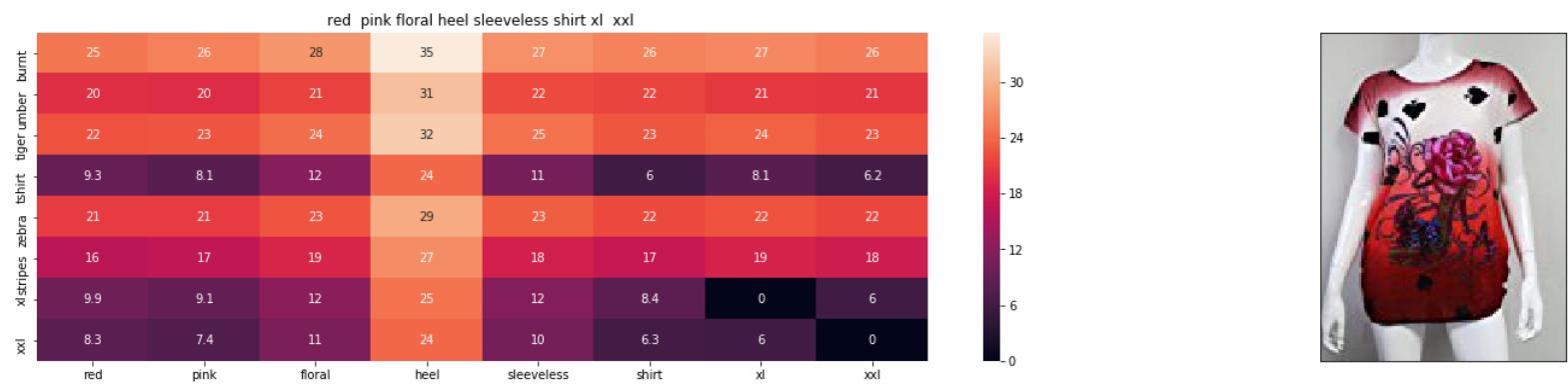


ASIN : B074P8MD22

Brand : Edista

euclidean distance from input : 4.51897779787

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JV63QQE	Si-Row	Red	TOYS_AND_GAMES Export to plot.ly »



ASIN : B00JV63QQE

Brand : Si Row

euclidean distance from input : 4.52937545794

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00C0I3U3E	Stanzino	Teal	SHIRT Export to plot.ly »

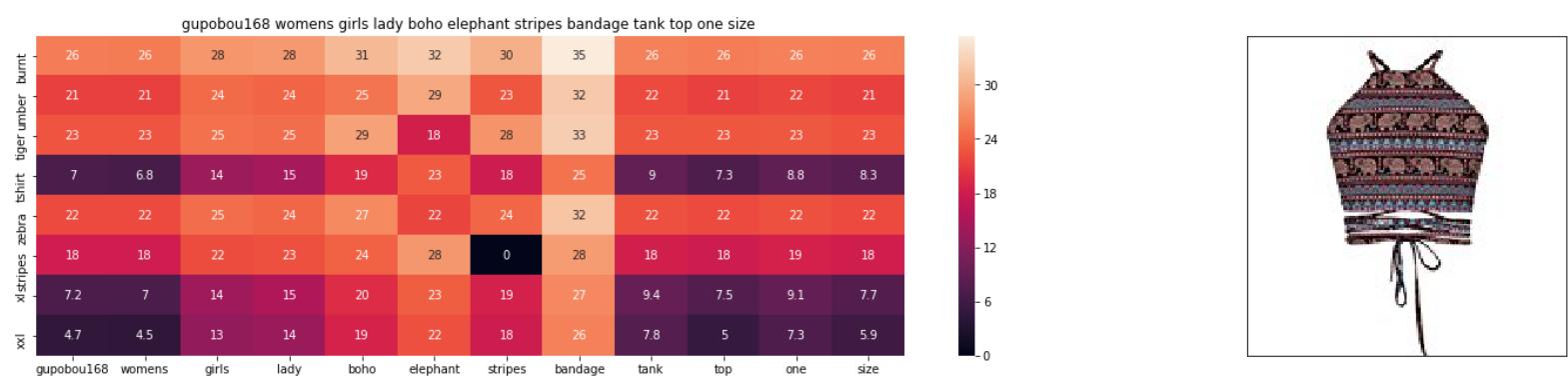


ASIN : B00C0I3U3E

Brand : Stanzino

euclidean distance from input : 4.53032614076

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B01ER18406	GuPoBoU168	Brown	SKIRT Export to plot.ly »

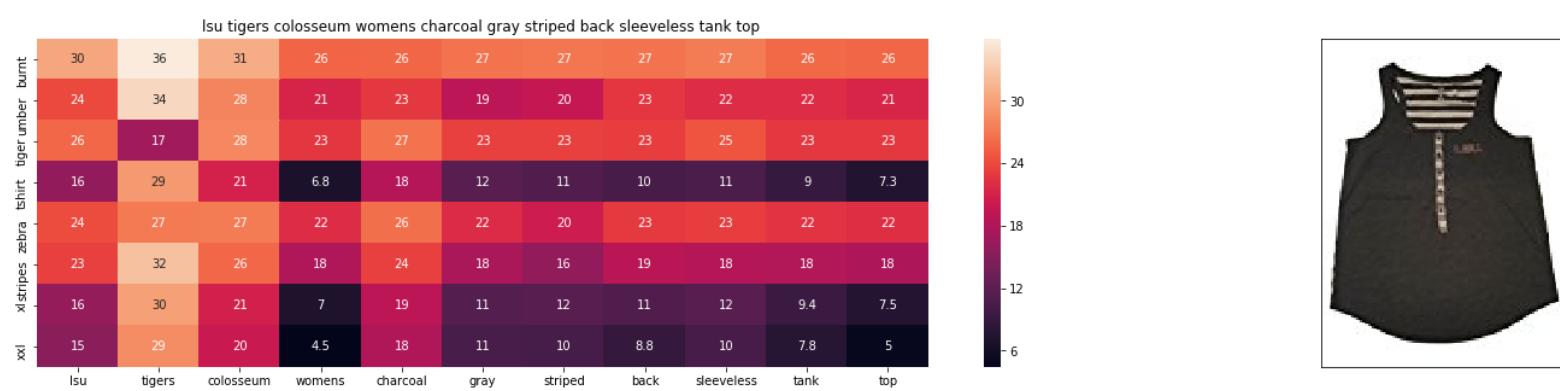


ASIN : B01ER18406

Brand : GuPoBoU168

euclidean distance from input : 4.54681702403

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B073R4ZM7Y	Colosseum	Gray	SPORTING_GOODS Export to plot.ly »

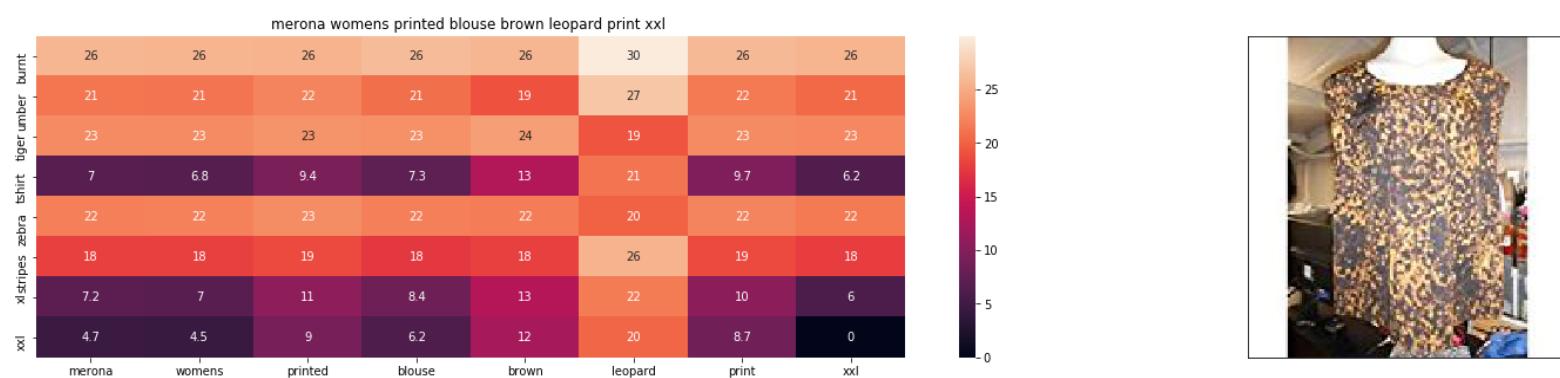


ASIN : B073R4ZM7Y

Brand : Colosseum

euclidean distance from input : 4.54835554445

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B071YF3WDD	Merona	-Brown-Leopard-Print	SHIRT Export to plot.ly »



ASIN : B071YF3WDD

Brand : Merona

euclidean distance from input : 4.61062742555

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B01C60RLDQ	1-Mad-Fit	Burgundy	SHIRT Export to plot.ly »



ASIN : B01C60RLDQ

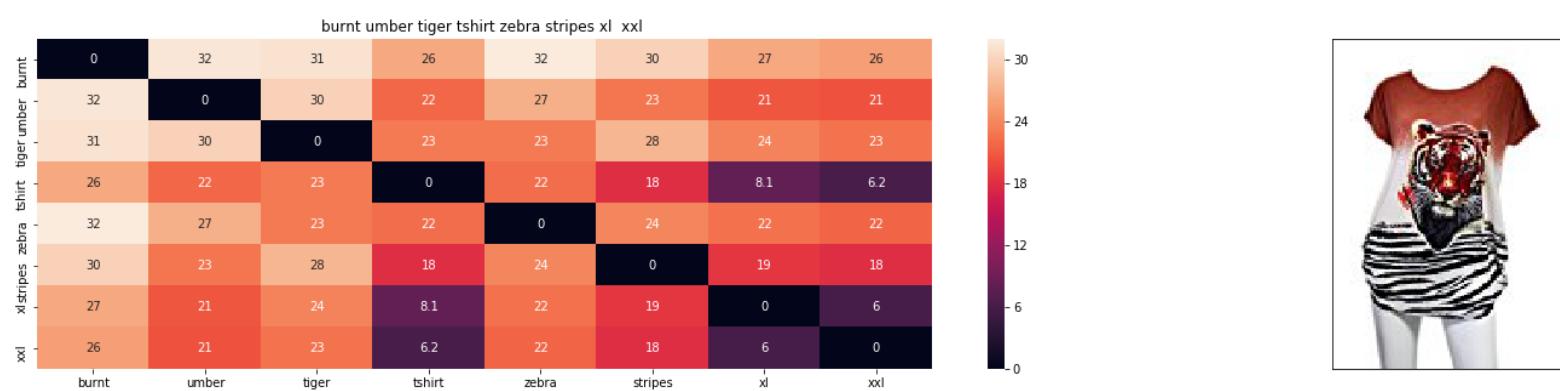
Brand : 1 Mad Fit

euclidean distance from input : 4.64591789282

In [60]: # brand and color weight =50
title vector weight = 5

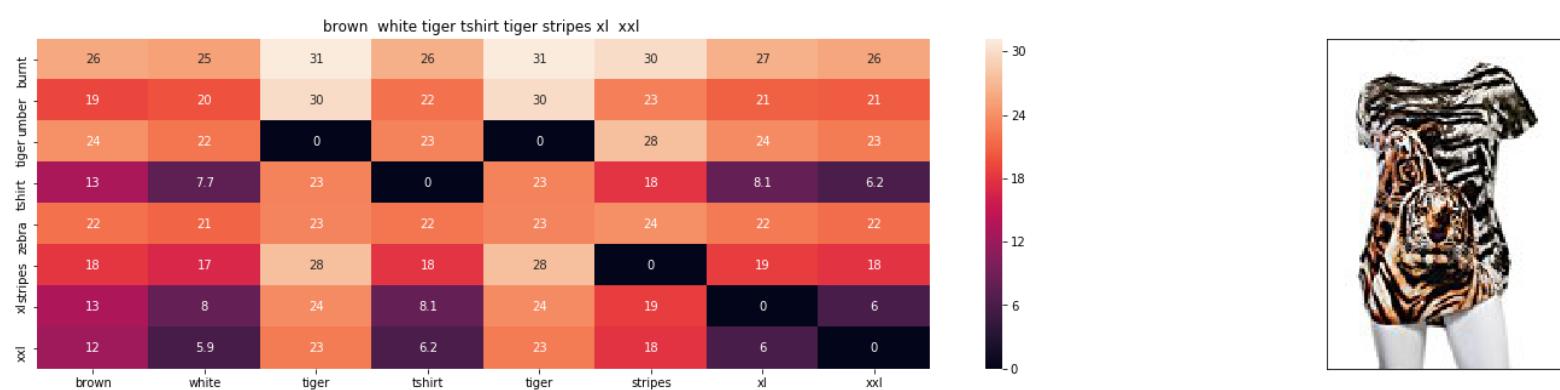
```
idf_w2v_brand(12566, 5, 50, 20)
```

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES Export to plot.ly »



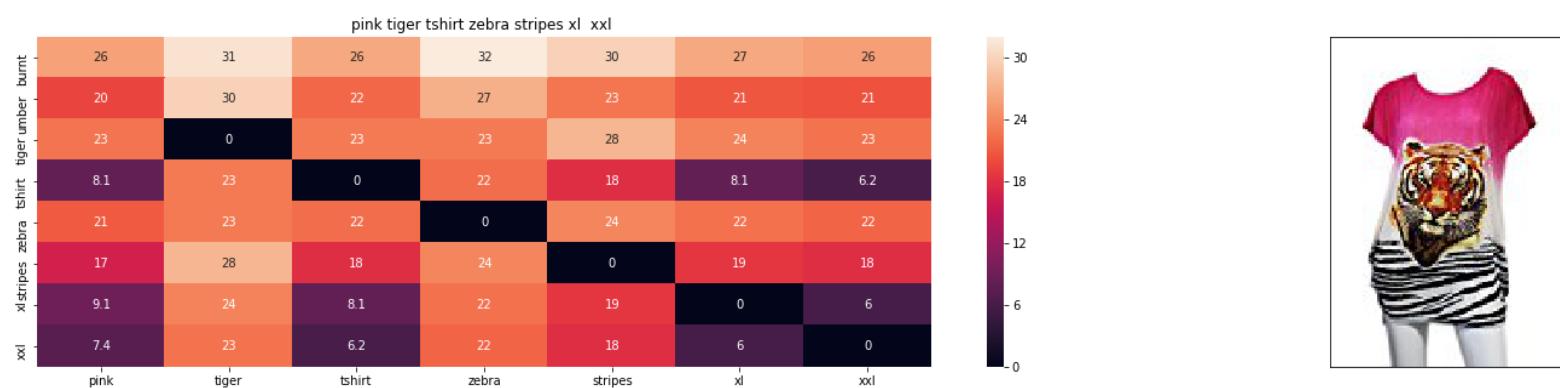
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.000355113636364

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQCWTO	Si-Row	Brown	TOYS_AND_GAMES Export to plot.ly »



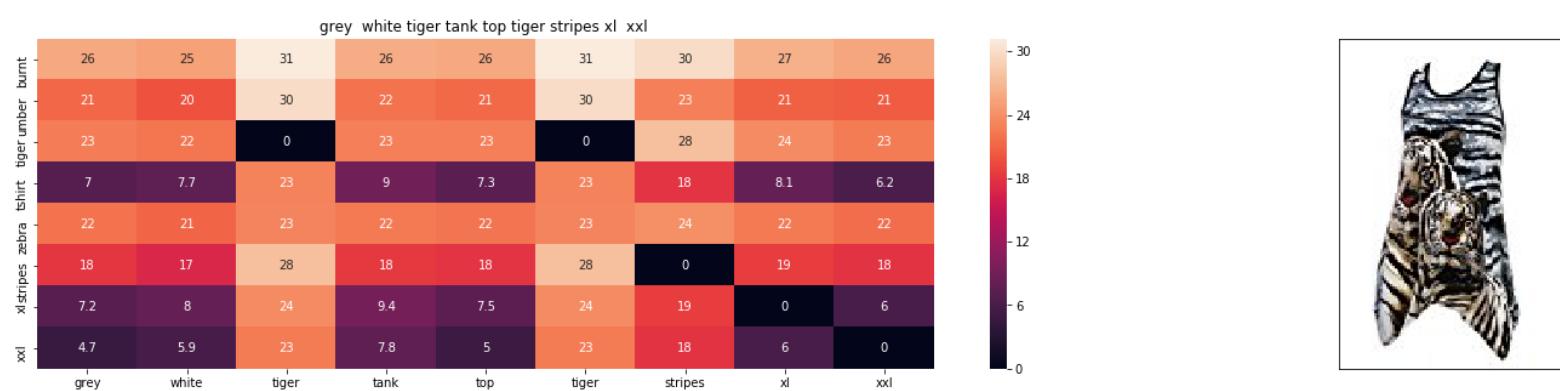
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 0.433722027865

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQASS6	Si-Row	Pink	TOYS_AND_GAMES Export to plot.ly »



ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 1.65509310669

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAFZ2	Si-Row	Grey	TOYS_AND_GAMES Export to plot.ly »

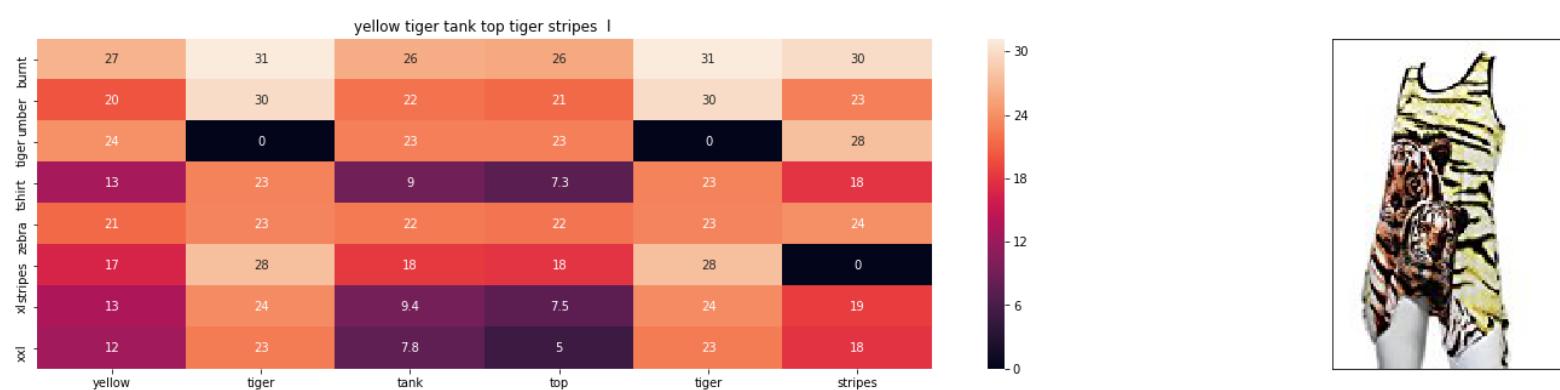


ASIN : B00JXQAFZ2

Brand : Si Row

euclidean distance from input : 1.77293604103

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAUWA	Si-Row	Yellow	TOYS_AND_GAMES Export to plot.ly »

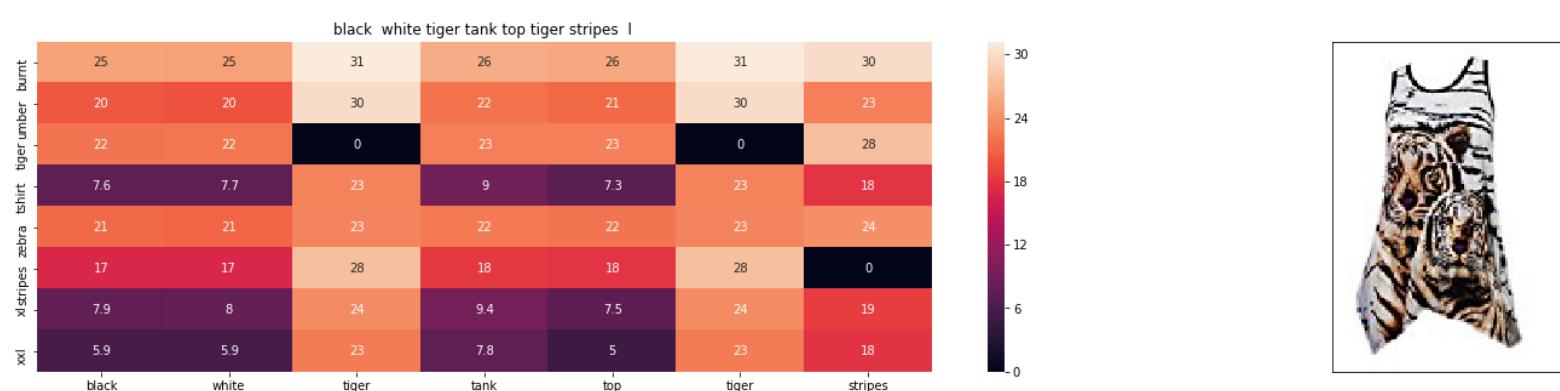


ASIN : B00JXQAUWA

Brand : Si Row

euclidean distance from input : 1.80287808538

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAO94	Si-Row	White	TOYS_AND_GAMES Export to plot.ly »

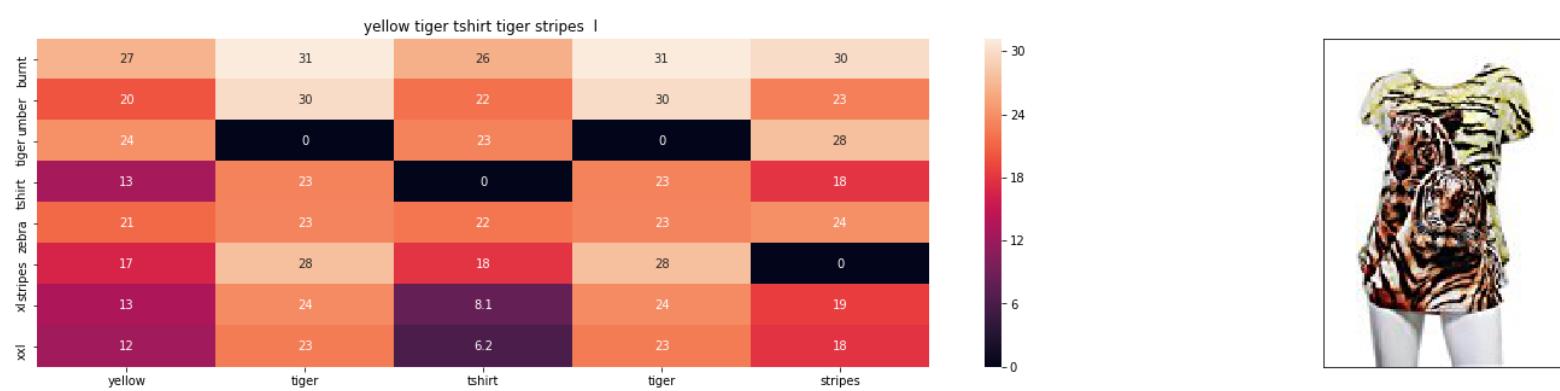


ASIN : B00JXQAO94

Brand : Si Row

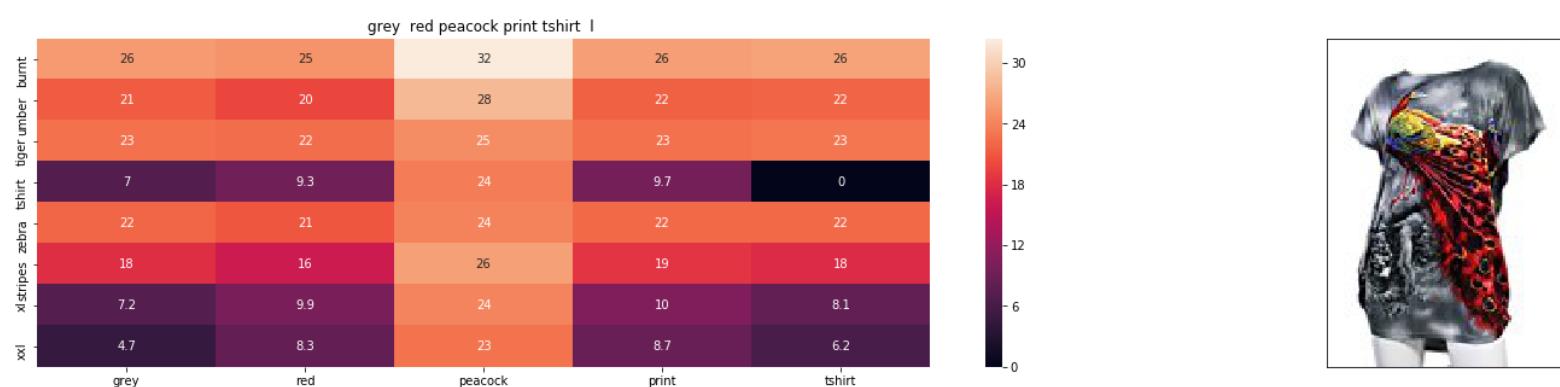
euclidean distance from input : 1.80319609241

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQCUIC	Si-Row	Yellow	TOYS_AND_GAMES Export to plot.ly »



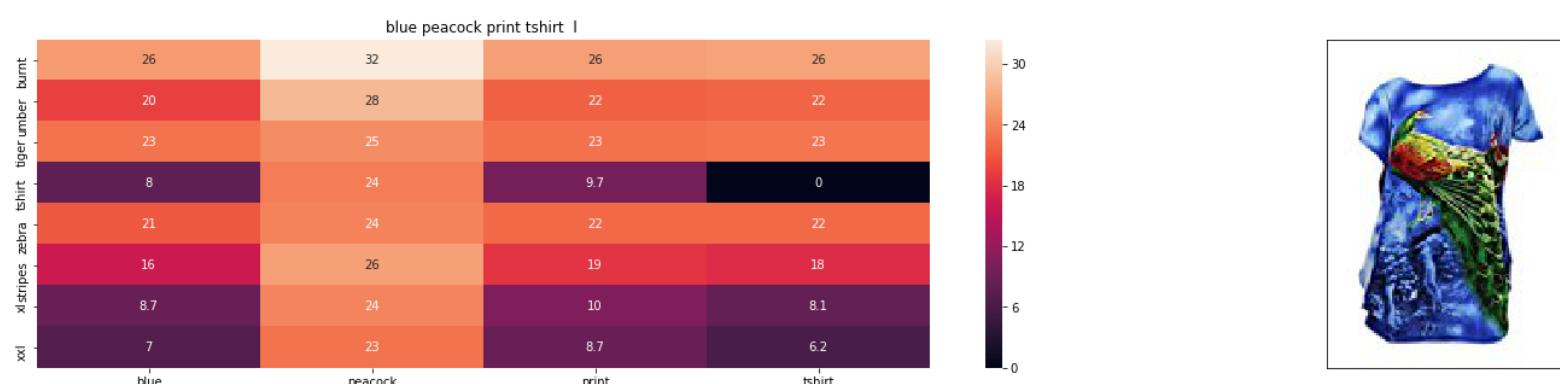
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 1.82141619628
=====

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQCFRS	Si-Row	Grey	TOYS_AND_GAMES Export to plot.ly »



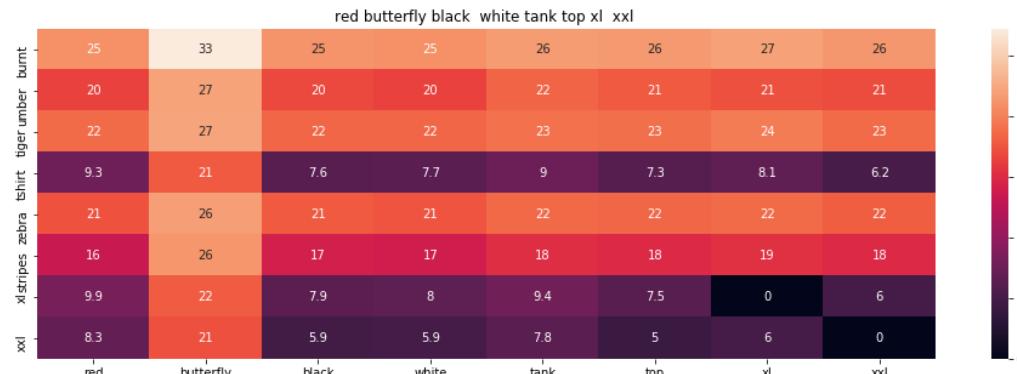
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 1.90777685025
=====

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQC8L6	Si-Row	Blue	TOYS_AND_GAMES Export to plot.ly »



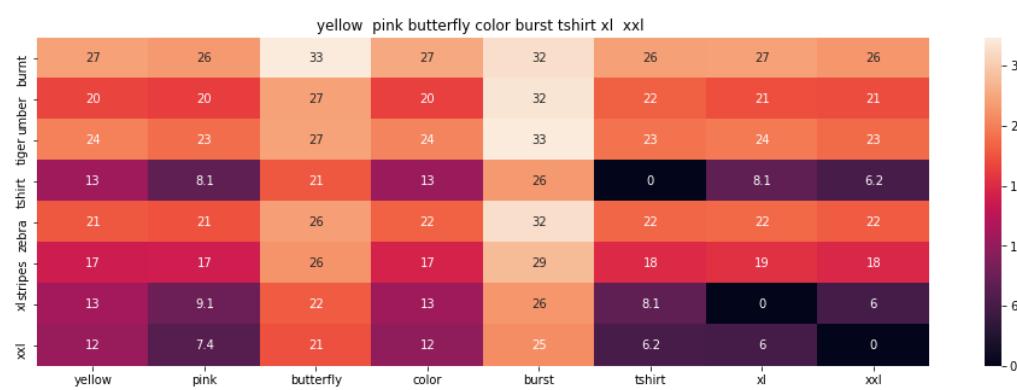
ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 1.92142937433
=====

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JV63CW2	Si-Row	Red	TOYS_AND_GAMES Export to plot.ly »



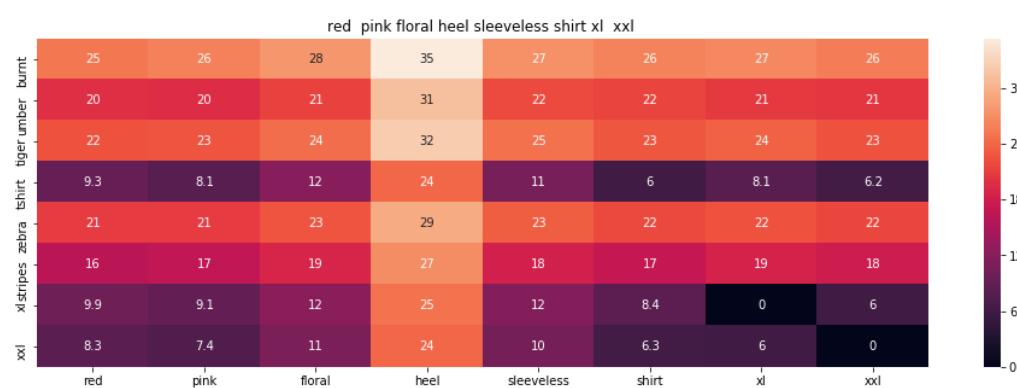
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 1.93646323497

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQBBMI	Si-Row	Yellow	TOYS_AND_GAMES Export to plot.ly »



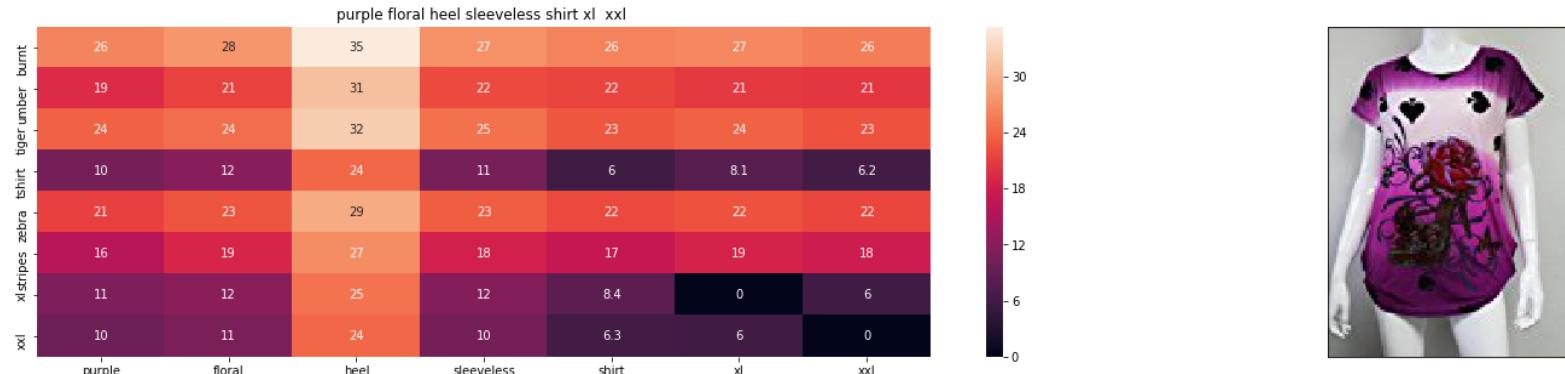
ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 1.95670381059

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JV63QQE	Si-Row	Red	TOYS_AND_GAMES Export to plot.ly »



ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 1.9806066343

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JV63VC8	Si-Row	Purple	TOYS_AND_GAMES Export to plot.ly »

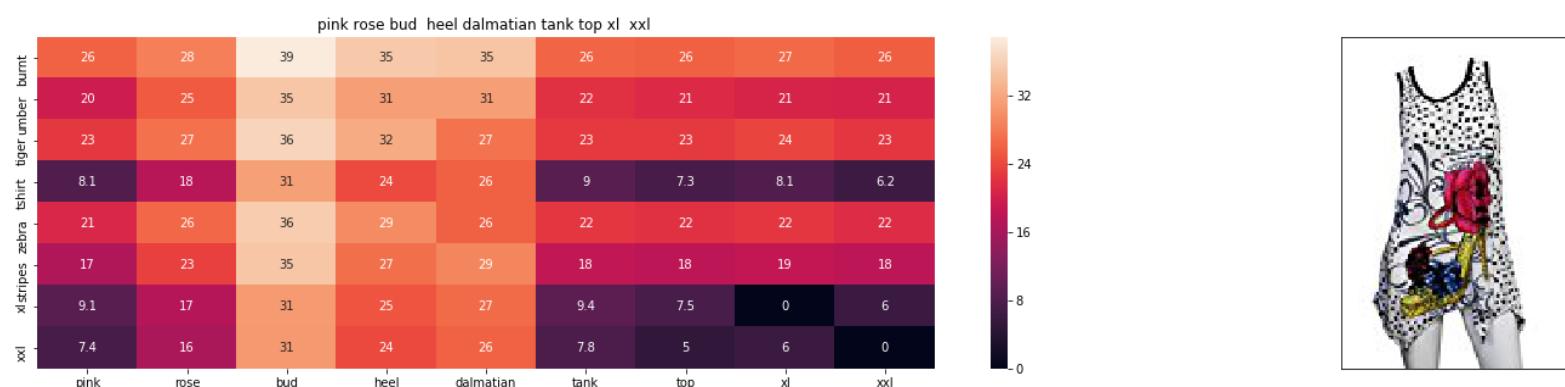


ASIN : B00JV63VC8

Brand : Si Row

euclidean distance from input : 2.01218559992

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAX2C	Si-Row	Pink	TOYS_AND_GAMES Export to plot.ly »



ASIN : B00JXQAX2C

Brand : Si Row

euclidean distance from input : 2.01335178755

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQC0C8	Si-Row	Blue	TOYS_AND_GAMES Export to plot.ly »

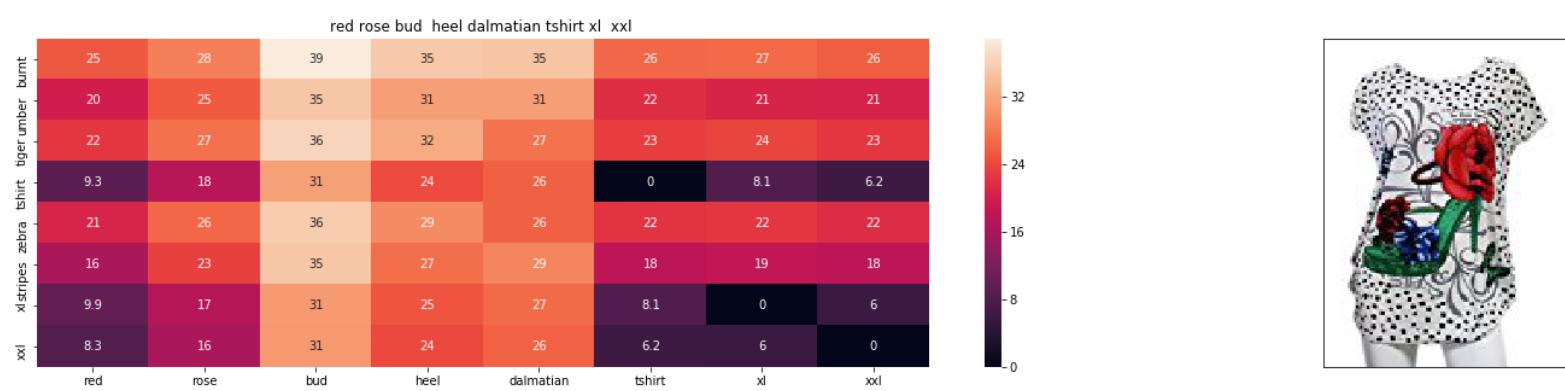


ASIN : B00JXQC0C8

Brand : Si Row

euclidean distance from input : 2.01388334827

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQABB0	Si-Row	Red	TOYS_AND_GAMES Export to plot.ly »

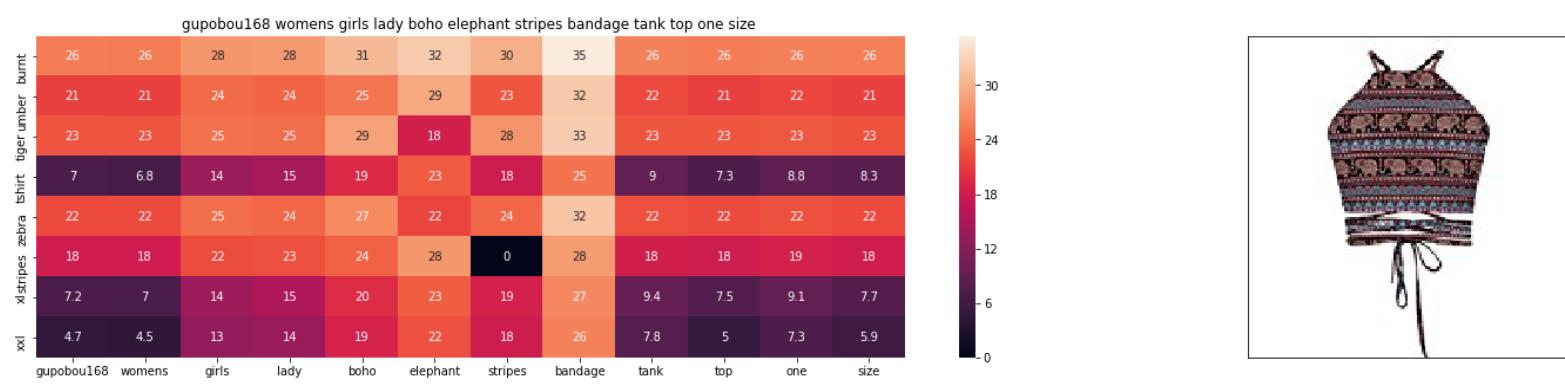


ASIN : B00JXQABB0

Brand : Si Row

euclidean distance from input : 2.0367257555

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B01ER18406	GuPoBoU168	Brown	SKIRT Export to plot.ly »

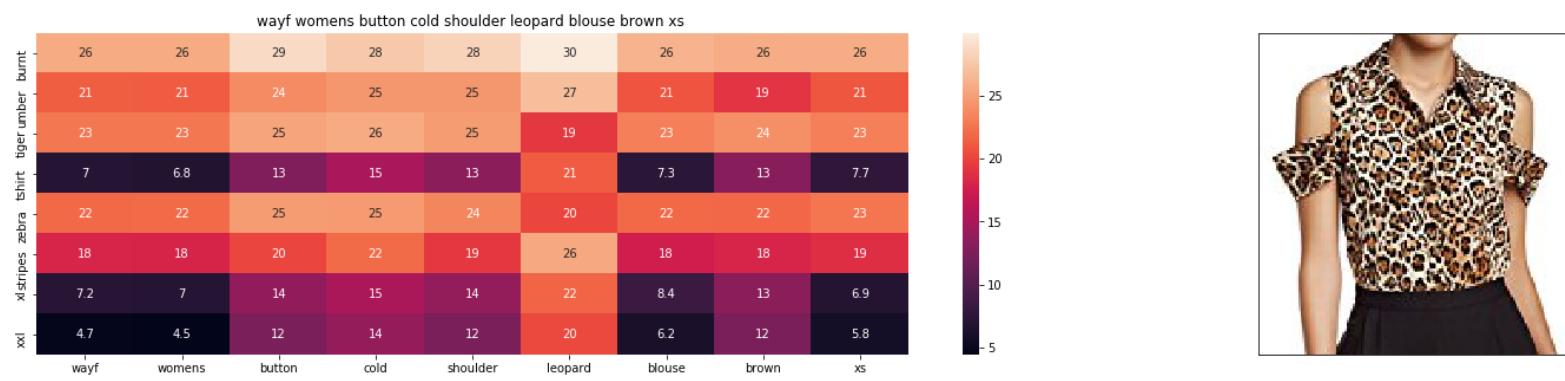


ASIN : B01ER18406

Brand : GuPoBoU168

euclidean distance from input : 2.65620416778

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B01LZ7BQ4H	WAYF	Brown	SHIRT Export to plot.ly »

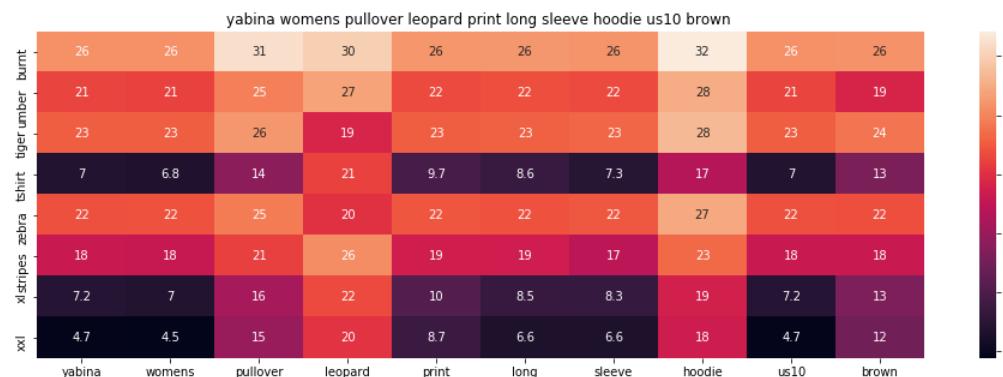


ASIN : B01LZ7BQ4H

Brand : WAYF

euclidean distance from input : 2.6849067823

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B01KJUM6JI	YABINA	Brown	BOOKS_1973_AND_LATER Export to plot.ly »

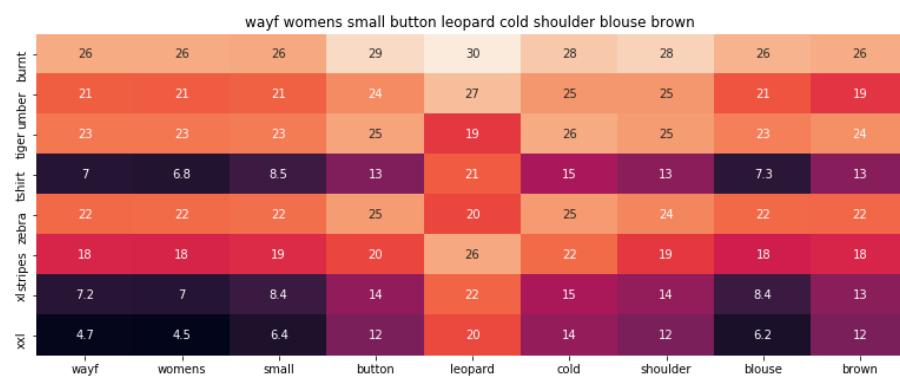


ASIN : B01KJUM6JI

Brand : YABINA

euclidean distance from input : 2.68583819266

Asin	Brand	Color	Product type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B01M06V4X1	WAYF	Brown	SHIRT Export to plot.ly »



ASIN : B01M06V4X1

Brand : WAYF

euclidean distance from input : 2.69476194865

[10.2] Keras and Tensorflow to extract features

```
In [42]: 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 [ ]: # https://gist.github.com/fchollet/f35fbc80e066a49d65f1688a7e99f069
# Code reference: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

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

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

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

# each image is converted into 25088 length dense-vector

'''

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

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

def save_bottlebeck_features():

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

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

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

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

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

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

    save_bottlebeck_features()

'''
```

[10.3] Visual features based product similarity.

```
In [44]: #load the features and corresponding ASINS info.
bottleneck_features_train = np.load('16k_data_cnn_features.npy')
asins = np.load('16k_data_cnn_feature_asins.npy')
asins = list(asins)

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

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

#get similar products using CNN features (VGG-16)
def get_similar_products_cnn(doc_id, num_results):
    doc_id = asins.index(df_asins[doc_id])
    pairwise_dist = pairwise_distances(bottleneck_features_train, bottleneck_features_train[doc_id].reshape(1,-1))

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

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

get_similar_products_cnn(12566, 20)
```



Product Title: burnt umber tiger tshirt zebra stripes xl xxl
 Euclidean Distance from input image: 0.0
 Amazon Url: www.amazon.com/dp/B00JXQB5FQ



Product Title: pink tiger tshirt zebra stripes xl xxl
 Euclidean Distance from input image: 30.0501
 Amazon Url: www.amazon.com/dp/B00JXQASS6



Product Title: yellow tiger tshirt tiger stripes l
 Euclidean Distance from input image: 41.2611
 Amazon Url: www.amazon.com/dp/B00JXQCUIC



Product Title: brown white tiger tshirt tiger stripes xl xxl
 Euclidean Distance from input image: 44.0002
 Amazon Url: www.amazon.com/dp/B00JXQCWT0



Product Title: kawaii pastel tops tees pink flower design
Euclidean Distance from input image: 47.3825
Amazon Url: www.amazon.com/dp/B071FCWD97



Product Title: womens thin style tops tees pastel watermelon print
Euclidean Distance from input image: 47.7184
Amazon Url: www.amazon.com/dp/B01JUNHBRM



Product Title: kawaii pastel tops tees baby blue flower design
Euclidean Distance from input image: 47.9021
Amazon Url: www.amazon.com/dp/B071SBCY9W



Product Title: edv cheetah run purple multi xl
Euclidean Distance from input image: 48.0465
Amazon Url: www.amazon.com/dp/B01CUPYBM0



Product Title: danskin womens vneck loose performance tee xsmall pink ombre
Euclidean Distance from input image: 48.1019
Amazon Url: www.amazon.com/dp/B01F7PHXY8



Product Title: summer alpaca 3d pastel casual loose tops tee design
Euclidean Distance from input image: 48.1189
Amazon Url: www.amazon.com/dp/B01I80A93G



Product Title: miss chievous juniors striped peplum tank top medium shadowpeach
Euclidean Distance from input image: 48.1313
Amazon Url: www.amazon.com/dp/B0177DM70S



Product Title: red pink floral heel sleeveless shirt xl xxl
Euclidean Distance from input image: 48.1695
Amazon Url: www.amazon.com/dp/B00JV63QQE



Product Title: moana logo adults hot v neck shirt black xxl
Euclidean Distance from input image: 48.2568
Amazon Url: www.amazon.com/dp/B01LX6H43D



Product Title: abaday multicolor cartoon cat print short sleeve longline shirt large
Euclidean Distance from input image: 48.2657
Amazon Url: www.amazon.com/dp/B01CR57YY0



Product Title: kawaii cotton pastel tops tees peach pink cactus design
Euclidean Distance from input image: 48.3626
Amazon Url: www.amazon.com/dp/B071WYLBZS



Product Title: chicago chicago 18 shirt women pink
Euclidean Distance from input image: 48.3836
Amazon Url: www.amazon.com/dp/B01GXAZTRY



Product Title: yichun womens tiger printed summer tshirts tops
Euclidean Distance from input image: 48.4493
Amazon Url: www.amazon.com/dp/B010NN9RX0



Product Title: nancy lopez whimsy short sleeve whiteblacklemon drop xs
Euclidean Distance from input image: 48.4788
Amazon Url: www.amazon.com/dp/B01MPX6IDX



Product Title: womens tops tees pastel peach ice cream cone print
Euclidean Distance from input image: 48.558
Amazon Url: www.amazon.com/dp/B0734GRKZL



Product Title: uswomens mary j blige without tshirts shirt
Euclidean Distance from input image: 48.6144
Amazon Url: www.amazon.com/dp/B01M0XXFKK

Create a weighted euclidean distance similarity model with title, brand, color and image, where each feature can be modified by applying weights.

```
In [18]: # All required imports
from datetime import datetime
import numpy as numpy
import pandas as pandas
import pickle
import math
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import pairwise_distances
from IPython.display import display, Image, SVG, Math, YouTubeVideo
```

```
In [10]: with open('word2vec_model', 'rb') as handle:
    model = pickle.load(handle)

# Load the original 16K dataset into data_16k_preprocessed
data_16k_preprocessed = pandas.read_pickle('pickels/16k_apperial_data_preprocessed')
```

```
In [11]: # Utility Methods

def nContaining(word):
    # return the number of documents which had the given word
    return sum(1 for blob in data_16k_preprocessed['title'] if word in blob.split())

def idf(word):
    #  $\text{idf} = \log(\frac{\text{number of docs}}{\text{number of docs which had the given word}})$ 
    return math.log(data_16k_preprocessed.shape[0] / (nContaining(word)))

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

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

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

```
In [12]: # Final model

def idf_w2v_one_hot_encoding_cnn_model(doc_id, result_count, wT, wB, wC, wI):
    # doc_id: apparel's id in given corpus
    # result_count: Number of results to show
    # wT: weight for title features
    # wB: weight for brand and color features
    # wC: weight for color features
    # wI: weight for image features

    # pairwise_dist will store the distance from given input apparel to all remaining apparels for title feature
    title_distance = pairwise_distances(title_features,title_features[doc_id].reshape(1,-1))

    # pairwise_dist will store the distance from given input apparel to all remaining apparels for brand feature
    brand_distance = pairwise_distances(brand_features,brand_features[doc_id])

    # pairwise_dist will store the distance from given input apparel to all remaining apparels for color feature
    color_distance = pairwise_distances(color_features,color_features[doc_id])

    # pairwise_dist will store the distance from given input apparel to all remaining apparels for image feature
    image_distance = pairwise_distances(image_features,image_features[doc_id].reshape(1,-1))

    # Weighted pairwise distance for all feature.
    weighted_pairwise_dist = (wT * title_distance + wB * brand_distance + wC * color_distance + wI * image_distance)/float(wT + wB + wC + wI)

    # Nearest indices of datapoints with respective to input doc_id
    nearest_indices = numpy.argsort(weighted_pairwise_dist.flatten())[0:result_count]

    # Nearest distance values of datapoints with respective to input doc_id
    nearest_distances = numpy.sort(weighted_pairwise_dist.flatten())[0:result_count]

    # List of nearest data points
    list_nearest_data_points = list(data_16k_preprocessed.index[nearest_indices])

    # Just to print the output
    for i in range(0,len(nearest_indices)):
        display(Image(url=data_16k_preprocessed['medium_image_url'].loc[list_nearest_data_points[i]], embed=True))
        print('ASIN : ',data_16k_preprocessed['asin'].loc[list_nearest_data_points[i]])
        print('TITLE : ',data_16k_preprocessed['title'].loc[list_nearest_data_points[i]])
        print('BRAND : ',data_16k_preprocessed['brand'].loc[list_nearest_data_points[i]])
        print('COLOR : ',data_16k_preprocessed['color'].loc[list_nearest_data_points[i]])
        print('Euclidean Distance from input is : ',nearest_distances[i])
        print('*125')
```

```
In [20]: start = datetime.now()

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

# Some of the brand values are empty.
# Need to replace Null with string "NULL"
data_16k_preprocessed[ 'brand'].fillna(value="Not given", inplace=True )

# Replace spaces with hyphen
brands = [x.replace(" ", "-") for x in data_16k_preprocessed[ 'brand'].values]
colors = [x.replace(" ", "-") for x in data_16k_preprocessed[ 'color'].values]

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


idf_title_vectorizer = CountVectorizer()
idf_title_features = idf_title_vectorizer.fit_transform(data_16k_preprocessed[ 'title'])

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

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

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


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

# for every title we build a weighted vector representation
for i in data_16k_preprocessed[ 'title']:
    title_features.append(build_avg_vec(i, 300, doc_id, 'weighted'))
    doc_id += 1

brand_vectorizer = CountVectorizer()
color_vectorizer = CountVectorizer()

# Creating Title Vector
print("Creating Title Features...")
title_features = numpy.array(title_features)

# Creating Brand Vector
print("\nCreating Brand Features...")
brand_features = brand_vectorizer.fit_transform(brands)

# Creating Color Vector
print("\nCreating Color Features...")
color_features = color_vectorizer.fit_transform(colors)

# Creating Image Vector
# Load the features using CNN features (VGG-16)
print("\nCreating Image Features...")
image_features = numpy.load('16k_data_cnn_features.npy')

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

Creating Title Features...

Creating Brand Features...

Creating Color Features...

Creating Image Features...

Time Taken: 0:02:30.059784

```
In [21]: start = datetime.now()
```

```
# Call out Model to process and print the output
# 1st argument : Valid document id from corpus
# 2nd argument : Number of results
# 3rd argument : Weight for title feature
# 4rd argument : Weight for brand feature
# 5rd argument : Weight for color feature
# 6rd argument : Weight for image feature
idf_w2v_one_hot_encoding_cnn_model(12566,20,99,1,1,1)

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



```
ASIN : B00JXQB5FQ
TITLE : burnt umber tiger tshirt zebra stripes xl xxl
BRAND : Si Row
COLOR : Brown
Euclidean Distance from input is : 0.003791360294117647
=====
=====
```



```
ASIN : B00JXQASS6
TITLE : pink tiger tshirt zebra stripes xl xxl
BRAND : Si Row
COLOR : Pink
Euclidean Distance from input is : 4.433812930540748
=====
=====
```



```
ASIN : B00JXQCWT0
TITLE : brown white tiger tshirt tiger stripes xl xxl
BRAND : Si Row
COLOR : Brown
Euclidean Distance from input is : 5.175207586849437
=====
=====
```



```
ASIN : B00JXQAFZ2
TITLE : grey white tiger tank top tiger stripes xl xxl
BRAND : Si Row
COLOR : Grey
Euclidean Distance from input is : 5.745065711529853
=====
=====
```



ASIN : B00JXQAUWA
TITLE : yellow tiger tank top tiger stripes 1
BRAND : Si Row
COLOR : Yellow
Euclidean Distance from input is : 6.060352198286421
=====



ASIN : B00JXQA094
TITLE : black white tiger tank top tiger stripes 1
BRAND : Si Row
COLOR : White
Euclidean Distance from input is : 6.0809096392498665
=====



ASIN : B00JXQCUIC
TITLE : yellow tiger tshirt tiger stripes 1
BRAND : Si Row
COLOR : Yellow
Euclidean Distance from input is : 6.245510198559284
=====



ASIN : B073R5Q8HD
TITLE : lsu tigers colosseum womens yellow purple slit back 12 sleeves tshirt
BRAND : Colosseum
COLOR : Yellow
Euclidean Distance from input is : 6.589048659642315
=====



ASIN : B015H41F6G
TITLE : kingde black white zebra stripe sleeveless vestbqn38
BRAND : KINGDE
COLOR : White
Euclidean Distance from input is : 6.600667142942618
=====



ASIN : B074P8MD22
TITLE : edista embellished zebra womens small tank cami black
BRAND : Edista
COLOR : Black
Euclidean Distance from input is : 6.800036236893675
=====



ASIN : B00C0I3U3E
TITLE : stanzino womens zebra print dolman sleeve chiffon top teal
BRAND : Stanzino
COLOR : Teal
Euclidean Distance from input is : 6.810369915195317
=====



ASIN : B01C60RLDQ
TITLE : leopard print raglan top burgundy size
BRAND : 1 Mad Fit
COLOR : Burgundy
Euclidean Distance from input is : 6.835738559801241
=====



ASIN : B073R4ZM7Y
TITLE : lsu tigers colosseum womens charcoal gray striped back sleeveless tank top
BRAND : Colosseum
COLOR : Gray
Euclidean Distance from input is : 6.866883645412747
=====



ASIN : B06XBY5QXL
TITLE : liz claiborne zebra fronttie top size xl
BRAND : Liz Claiborne
COLOR : Crema Multi/Black
Euclidean Distance from input is : 6.937220405129826
=====



ASIN : B071YF3WDD
TITLE : merona womens printed blouse brown leopard print xxl
BRAND : Merona
COLOR : , Brown Leopard Print
Euclidean Distance from input is : 6.958041895211689
=====



ASIN : B00H8A6ZLI
TITLE : top zebra print dolman sleeve top one size
BRAND : Vivian's Fashions
COLOR : Black
Euclidean Distance from input is : 7.011124128457415
=====



ASIN : B01IPV1SFQ
TITLE : daniel rainn orange pink ivory white print chiffon tank top 68 white ivory xl
BRAND : Daniel Rainn
COLOR : White Ivory
Euclidean Distance from input is : 7.05460895190143
=====



ASIN : B06Y1VN8WQ
TITLE : black swan estera raw umber 1 womens shirt
BRAND : Black Swan
COLOR : Raw Umber
Euclidean Distance from input is : 7.05989578572065
=====



ASIN : B00Z6HEXWI

TITLE : western zebra pattern sleeveless shirt vest tank tops pink xxl

BRAND : Black Temptation

COLOR : Multicolored

Euclidean Distance from input is : 7.062816754512964

=====



ASIN : B00KSNTY7Y

TITLE : anna kaci womens asymmetrical sheer brown leopard cheetah print long sleeve top multicoloured medium

BRAND : Anna-Kaci

COLOR : Multicoloured

Euclidean Distance from input is : 7.063357693544827

=====

Time Taken: 0:00:06.627154

Conclusion

1. Created a weighted euclidean distance similarity model with title, brand, color and image, where each feature can be modified by applying weights.
2. This is a first-cut model, whereas in actual production environment, group of models are used to get the best output, which must be in-compliance with business rules of the organization.