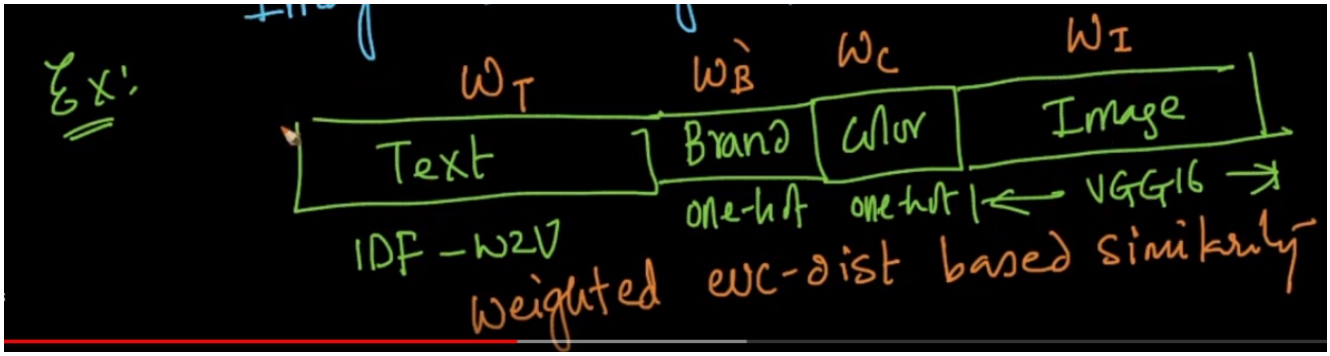
Assignment



In [1]:

```
#import all the necessary packages.
# For Image Processing
from PIL import Image
import requests
from io import BytesIO

# For Text Processing
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from collections import Counter # To count each of the list

# Plotting purpose
import plotly
import plotly.figure_factory as ff
from plotly.graph_objs import Scatter, Layout
import matplotlib.pyplot as plt # For plotting purpose
import seaborn as sns # Plot purpose
from matplotlib import gridspec # To divide the region to plot

# Metrics
from sklearn.metrics.pairwise import cosine_similarity # Cal. cosine sim.
from sklearn.metrics import pairwise_distances  # Cal. distance b/w of each pair of datapts

import numpy as np # fast computation purpose
import pandas as pd # Print the data in tabular form
import warnings
import os # environment
import math # Apply some math concept
import time # To see execution time
import re # Regular Expression
from scipy.sparse import hstack # Merge all features
import pickle # Read file from pickle
import itertools

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

In [2]:

```
# Load Asin, CNN features of images
asins = np.load('16k_data_cnn_feature_asins.npy')
cnn_feature = np.load('16k_data_cnn_features.npy')
print('Shape of asins: {}\nShape of cnn_feature: {}'.format(asins.shape,cnn_feature.shape))
```

```
Shape of asins: (16042,)
Shape of cnn_feature: (16042, 25088)
```

In [3]:

```
# Load original dataset
data = pd.read_pickle('pickels/16k_apperal_data_preprocessed')
# Store asins values from original data
d_asins = list(data['asin'])

print('Shape of data: {}'.format(data.shape))
```

```
Shape of data: (16042, 7)
```

In [4]:

```
# Display data to see everything is working fine.
data.head()
```

Out[4]:

	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

Dealing with Text

In [5]:

```
# Import w2v file
with open('word2vec_model', 'rb') as handle:
    model = pickle.load(handle)
```

In [6]:

```
# Featurize TFIDF...
# We are going to need idf values and their feature names only!!
title_tfidf = TfidfVectorizer()
title_idf_feature = title_tfidf.fit_transform(data['title'])
```

In [7]:

```
# Create dict store key as feature names and value as idf_score
title_idf = {}
for index, fname in enumerate(title_tfidf.get_feature_names()):
    title_idf[fname] = title_tfidf.idf_[index]
```

In [8]:

```
# https://stackoverflow.com/questions/29216889/slicing-a-dictionary
dict(itertools.islice(title_idf.items(), 5))
```

Out[8]:

```
{'00': 9.584415607682967,
 '000': 9.98988071579113,
 '00008066': 9.98988071579113,
 '0000844': 9.98988071579113,
 '000085200': 9.98988071579113}
```

In [9]:

```
np.array([1,2,3,4])/2
```

Out[9]:

```
array([0.5, 1. , 1.5, 2. ])
```

In [10]:

```
def idf_w2v(model, sentence):

    vec_word = np.zeros(300,)
    vec_idf = 0
    # Iterate each word
    for i in sentence.split():
        if i in model.keys() and i in title_idf.keys():
            vec_word += title_idf[i]*model[i]
            vec_idf += title_idf[i]

    return vec_word/vec_idf
```

In [11]:

```
title_w2v = np.zeros((data['title'].shape[0],300))

for i in range(data.shape[0]):
    title_w2v[i] = idf_w2v(model,data['title'].iloc[i])
```

Dealing with Brand and Color

In [12]:

```
# One hot encoding using countervizer

# Need to replace Null with string "NULL"
data['brand'].fillna(value="Not given", inplace=True )

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

# Brand
brand_vec = CountVectorizer(binary=True)
brand_onehot = brand_vec.fit_transform(brands)
brand_feat = brand_vec.get_feature_names()

# Color
color_vec = CountVectorizer(binary=True)
color_onehot = color_vec.fit_transform(colors)
color_feat = color_vec.get_feature_names()
```

Dealing with Images

In [13]:

```
# Already pretrained
cnn_feature[:5]
```

Out[13]:

Out[13]:

```
array([[0.11759627, 0.        , 0.        , ..., 0.        , 0.67754525,
        0.        ],
       [0.06240697, 0.        , 0.        , ..., 0.        , 0.6599147 ,
        0.        ],
       [0.39736205, 0.        , 0.        , ..., 0.        , 0.6154604 ,
        0.        ],
       [0.17635377, 0.        , 0.        , ..., 0.        , 0.53826314,
        0.        ],
       [0.4577136 , 0.        , 0.        , ..., 0.        , 0.5774003 ,
        0.        ]], dtype=float32)
```

Recomd

In [14]:

```
#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)
```

In [15]:

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

    final_dist = []

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

    # final_dist = np.array(#number of words in title1 * #number of words in title2)
    # final_dist[i,j] = euclidean distance between vectors i, j
    return np.array(final_dist)
```

In [16]:

```
title_idf_feature  = title_idf_feature.astype(np.float)
vocab = model.keys()

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  title_tfidf.vocabulary_:
                vec.append(title_idf_feature[doc_id, title_tfidf.vocabulary_[i]] * model[i])
            elif m_name == 'avg':
                vec.append(model[i])
        else:
            # if the word in our courpus is not there in the google word2vec corpus, we are just ignori
ng it
            vec.append(np.zeros(shape=(300,)))
    # we will return a numpy array of shape (#number of words in title * 300 ) 300 = len(w2v_model[word
])
```

```
    # each row represents the word2vec representation of each word (weighted/avg) in given sentance
    return  np.array(vec)
```

In [17]:

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

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

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

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

    data_matrix = [['Asin','Brand', 'Color', 'title'],
                [data['asin'].loc[df_id1],brands[doc_id1], colors[doc_id1], sentance1], # input apparel's features
                [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2], sentance2]] # recommonded apparel's features

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


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

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

    # in first 25*10 grids we plot heatmap
    ax1 = plt.subplot(gs[:, :-5])
    # ploting the heap map based on the pairwise distances
    ax1 = sns.heatmap(np.round(s1_s2_dist,6), annot=True)
    # set the x axis labels as recommended apparels title
    ax1.set_xticklabels(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 [18]:

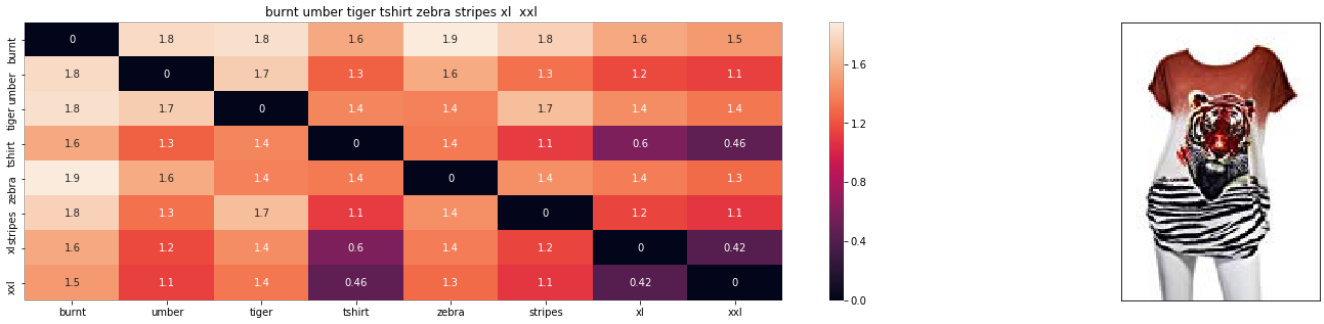
```
def idf_w2v_brand_color(index, wt,wb,wc,wi,n_feat):
    # pairwise distance whereas this X parameter act as corpus idf titles and Y parameter act as one idf title
    title_dist = pairwise_distances(title_w2v_title_w2v[index].reshape(1,-1))
```

```
    title_dist = pairwise_distances(title_w2v,title_w2v[index].reshape(1,-1))

    # pairwise distance whereas this X parameter act as corpus brand and Y parameter act as one brand
    brand_dist = pairwise_distances(brand_onehot,brand_onehot[index].reshape(1,-1))

    # pairwise distance whereas this X parameter act as corpus color and Y parameter act as one color
    color_dist = pairwise_distances(color_onehot,color_onehot[index].reshape(1,-1))

    # pairwise distance whereas this X parameter act as corpus images and Y parameter act as one image
    img_dist = pairwise_distances(cnn_feature,cnn_feature[index].reshape(1,-1))
    pw_dist = (wt * title_dist + wb * brand_dist + wc * color_dist + wi * img_dist)/float(wt+wb+wc+wi)

    # Sort in ascending because we are using euc. dist.
    ind_sort = np.argsort(pw_dist.flatten())[0:n_feat]

    # Based on index sort, get their distance score
    pw_dist = pw_dist[ind_sort].flatten()

    # Also store title with index sort
    df_indices = list(data.index[ind_sort])

    # Based on we got ind_sort value, we want to plot the recomment using heatmap
    for i in range(len(ind_sort)):

        heat_map_w2v_brand(data['title'].loc[df_indices[0]], \
                           data['title'].loc[df_indices[i]], \
                           data['medium_image_url'].loc[df_indices[i]], \
                           ind_sort[0], \
                           ind_sort[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 :', pw_dist[i])
        print('='*125)
```

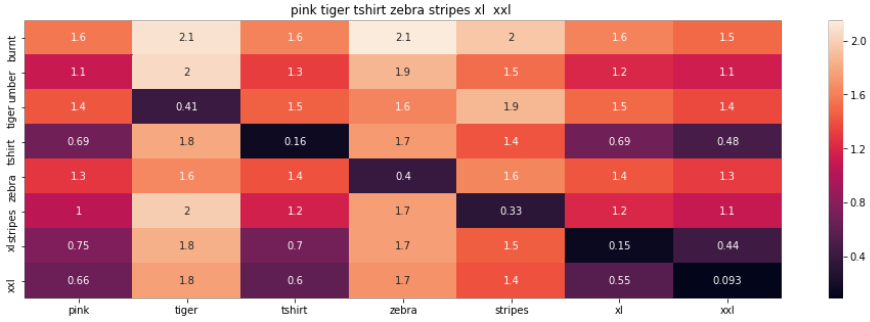
When I Put Title weight highest

In [20]:

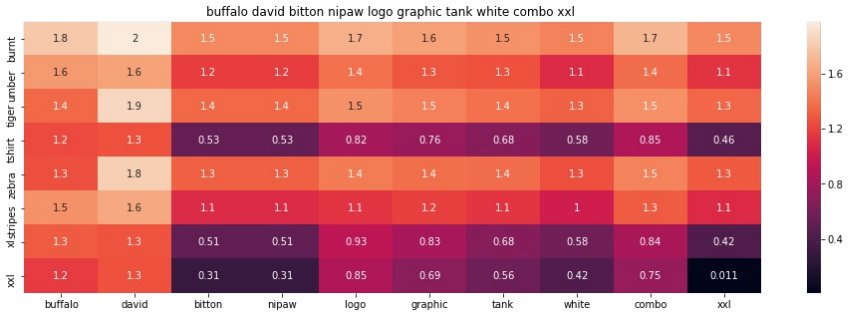
```
idf_w2v_brand_color(12566,20,1,1,1,5)
```



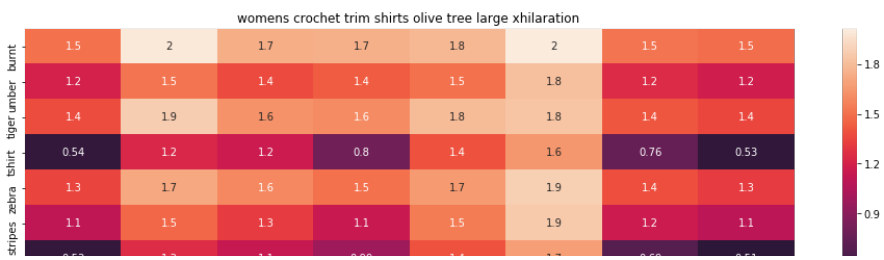

```
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 3.524037113137361e-07
=============================================================================================================
======================
```

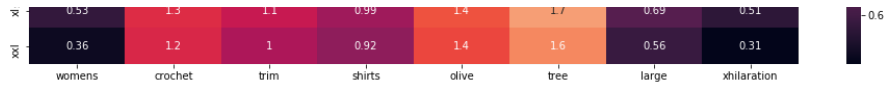


ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 2.847047198575608

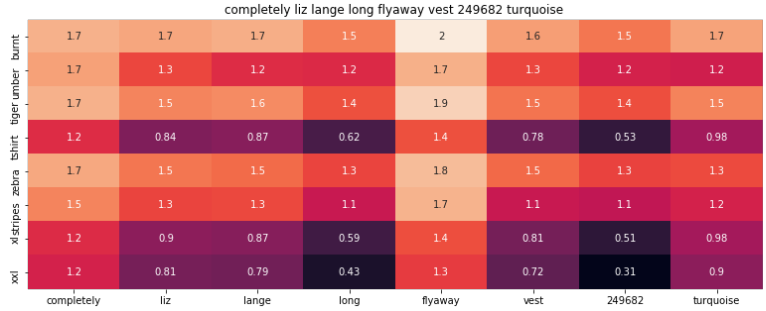


ASIN : B018H5AZXQ
Brand : Buffalo
euclidean distance from input : 3.009161385528684





ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 3.0851707128696004
==========

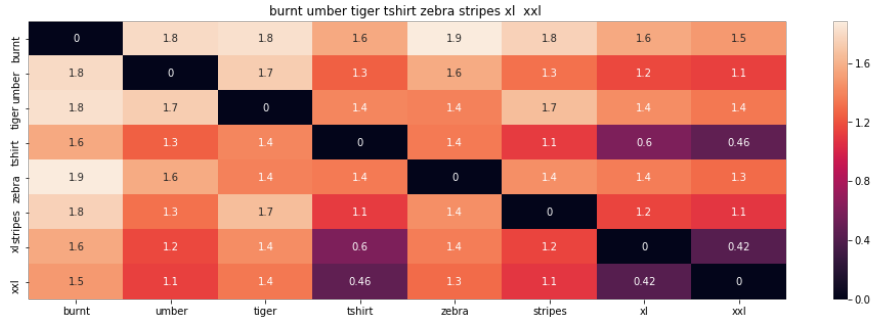


ASIN : B074LTBWSW
Brand : Liz Lange
euclidean distance from input : 3.128606119620645
==========

When I Put brand weight highest

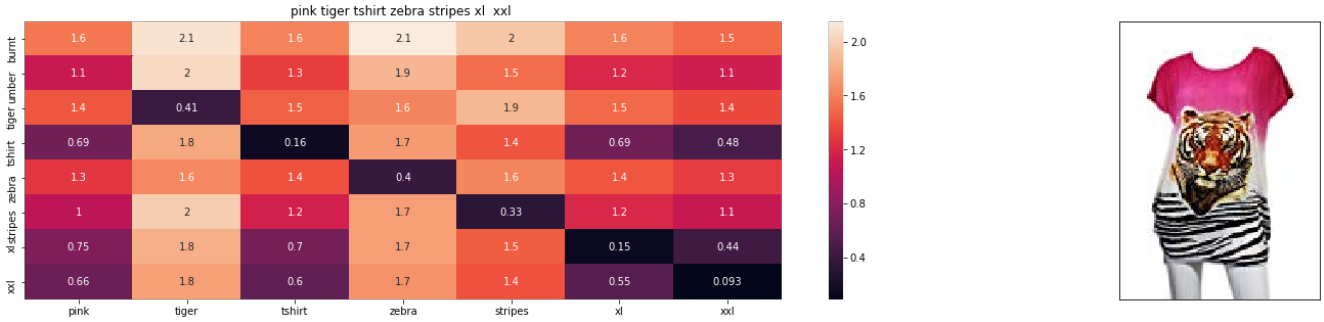
In [21]:

```
idf_w2v_brand_color(12566,1,20,1,1,5)
```



ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 3.277844015152052e-07
==========

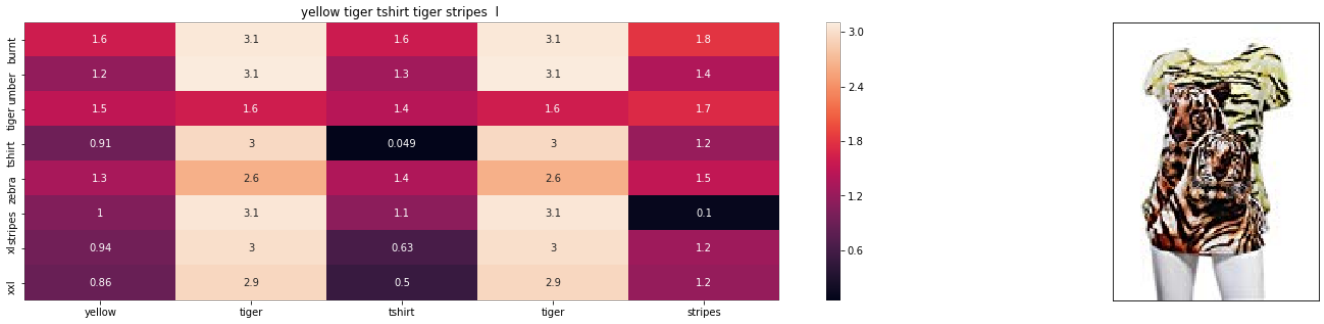
==========



ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 2.204431651909357

==========

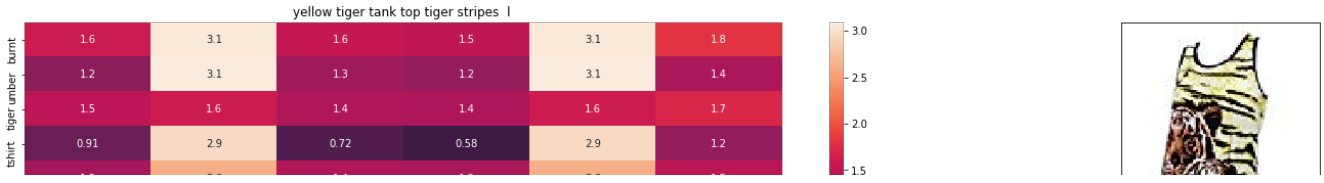
==========

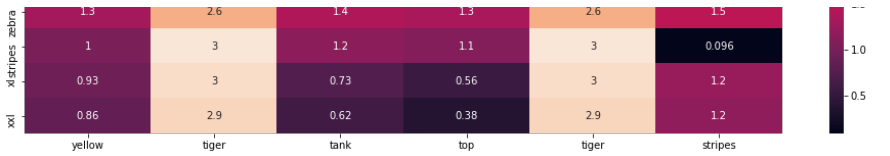


ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 2.3873902274282846

==========

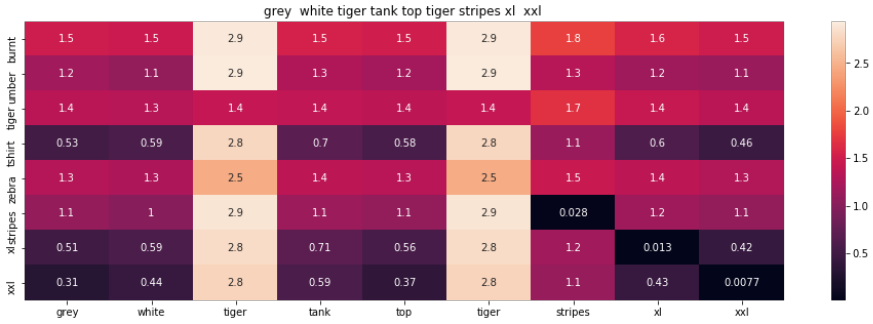
==========





ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 2.443557690368855

==========



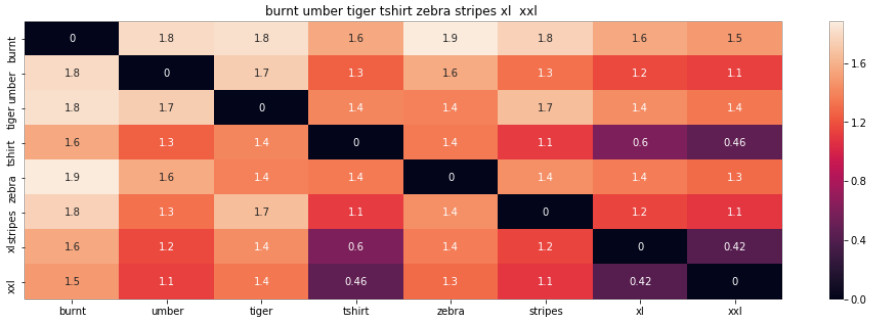
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 2.4480314034792743

==========

When I Put color weight highest

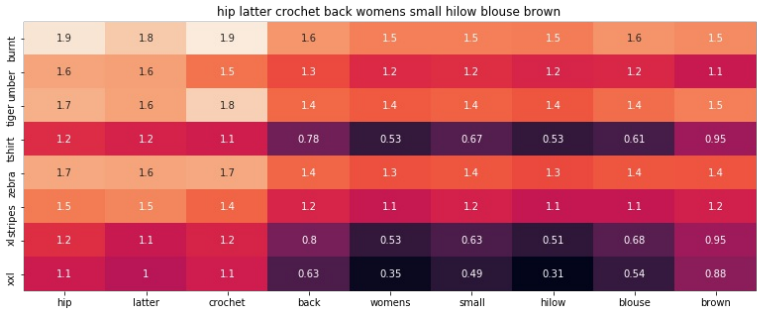
In [22]:

```
idf_w2v_brand_color(12566,1,1,20,1,5)
```

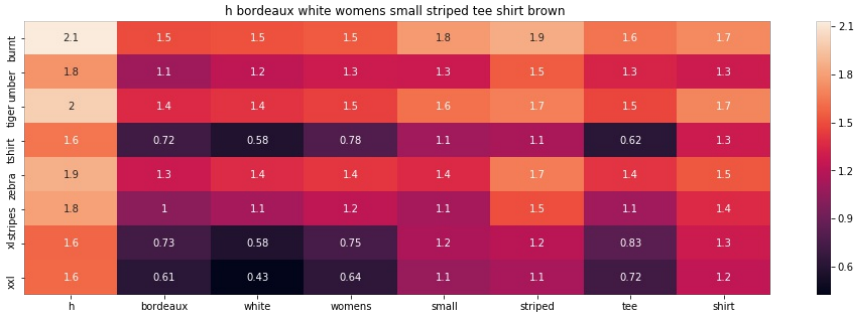


ASIN : B00JXQB5FQ

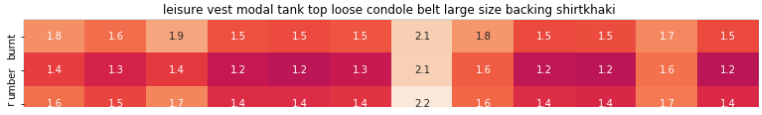
Brand : Si Row
euclidean distance from input : 3.277844015152052e-07
==========

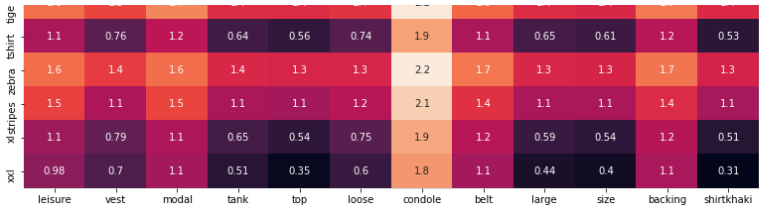


ASIN : B074MJN1K9
Brand : Hip
euclidean distance from input : 2.1189781032405866
==========

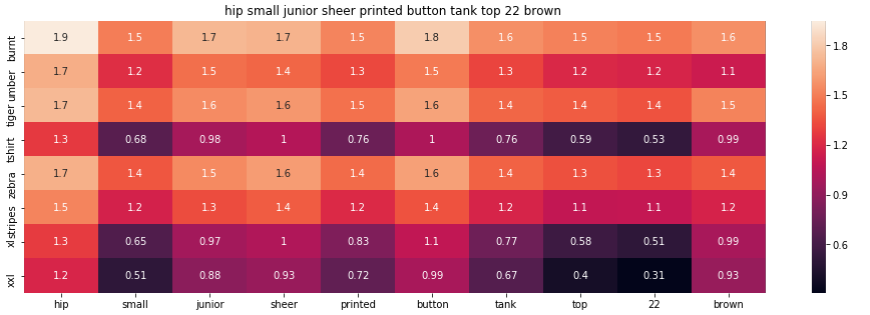


ASIN : B072BVB47Z
Brand : H By Bordeaux
euclidean distance from input : 2.1365242256396018
==========





ASIN : B014OUHUZY
Brand : Black Temptation
euclidean distance from input : 2.224791790770683

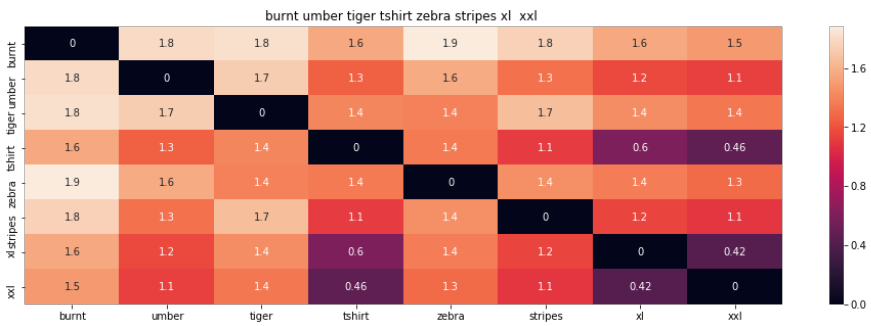


ASIN : B071LDTQ1F
Brand : Hip
euclidean distance from input : 2.3278722357995307

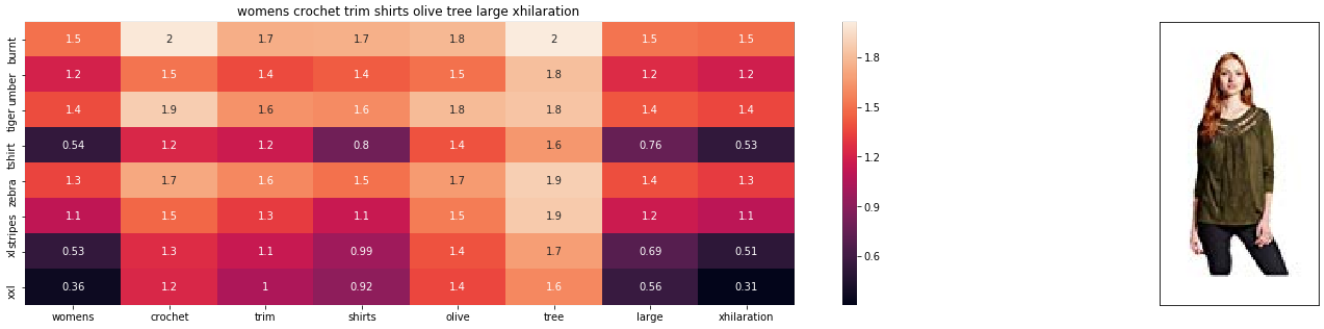
When I Put image weight highest

In [25]:

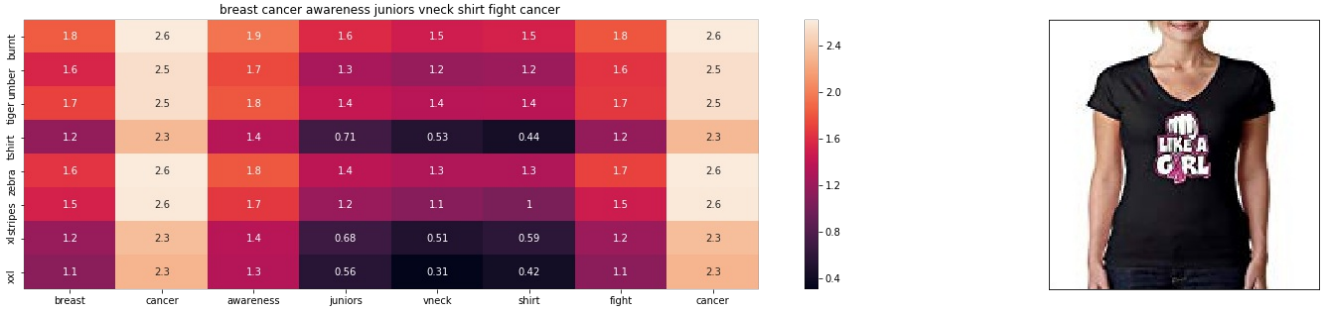
```
idf_w2v_brand_color(12566,1,1,1,20,5)
```



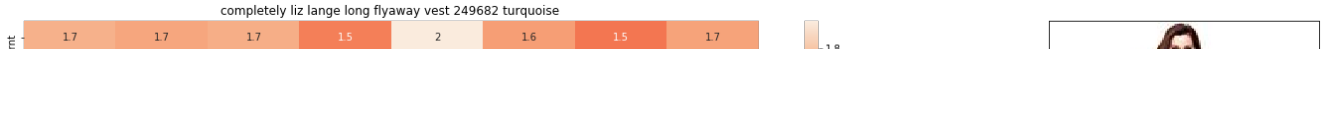
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 6.531068483246085e-06

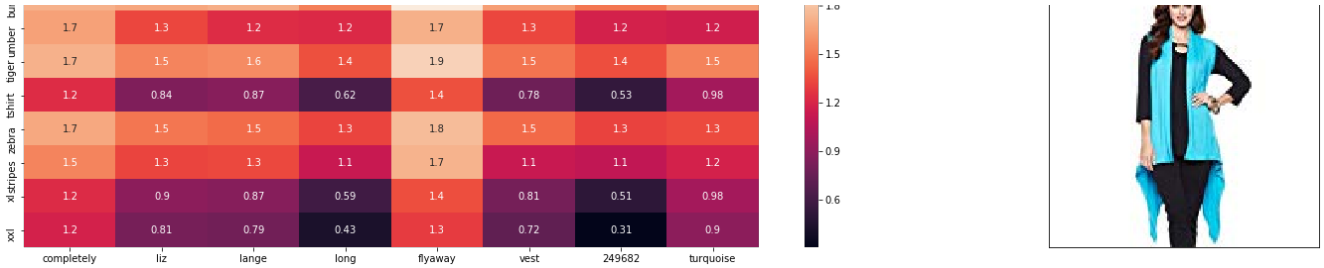


ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 32.55369071482225

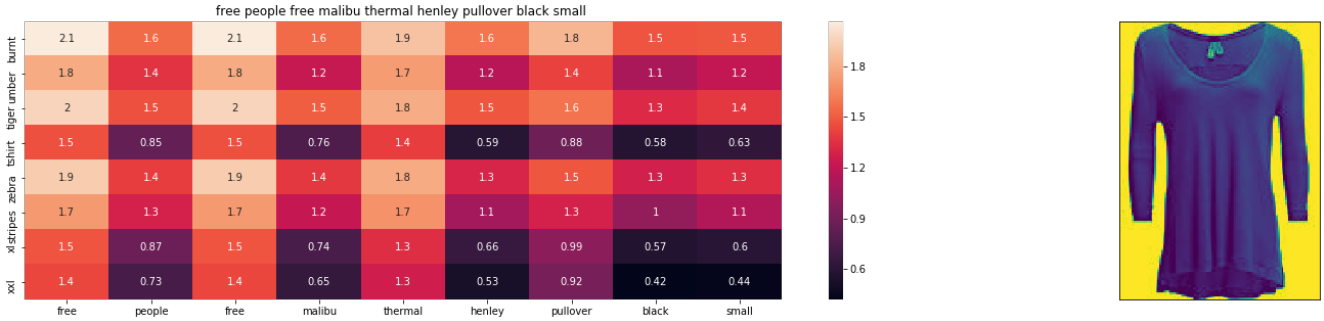


ASIN : B016CU40IY
Brand : Juiceclouds
euclidean distance from input : 33.54351664926501





ASIN : B074LTBWSW
Brand : Liz Lange
euclidean distance from input : 34.44161415781852
==========



ASIN : B074MXY984
Brand : We The Free
euclidean distance from input : 34.4553628863689
==========

Observation:

1. Giving the more weightage in color or title didn't given the good recommended
2. Apart from other weightage (color or brand), it gives better resultant output

Conclusion

1. Load the preprocessed data and cnn images feature data
2. Perform IDF W2V on title, OneHotEncoding on brand and color and already pretrained images features using cnn
3. We define a **display_img** function to show the image
4. We define a **get_distance** function to get the distance between two vectors to inorder to display output
5. We define a **get_word_vec** function to get the evaluation of idf w2v inorder to display score in output
6. We define a **heat_map_w2v_brand** function to get overall figure configuration and distance calculated between 2 vectors for every index sort values
7. we define **idf_w2v_brand_color** function to to evaluate weighted euc. dist similarity of 4 features
8. Got the display output

In [ ]: