Apparel Recommendations using recommended weighted

In [14]:
```
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")
```

load data

In [2]:
```
img_data = np.load('16k_data_cnn_features.npy')
asins = np.load('16k_data_cnn_feature_asins.npy')
data = pd.read_pickle('pickels/16k_apperal_data_preprocessed')
df_asins = list(data['asin'])
asins = list(asins)
```

In [3]:
```
import pickle
with open('word2vec_model', 'rb') as handle:
    model = pickle.load(handle)
```

In [4]:
```
data['brand'].fillna(value="Not given", inplace=True )

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

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

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

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

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

In [5]:
```
idf_title_vectorizer = CountVectorizer()
idf_title_features = idf_title_vectorizer.fit_transform(data['title'])
```

In [8]:
```
vocab = model.keys()
# this function will add the vectors of each word and returns the avg vector of giv
en sentance
def build_avg_vec(sentence, num_features, doc_id, m_name):
    # sentace: its title of the apparel
    # num_features: the lenght of word2vec vector, its values = 300
    # m_name: model information it will take two values
        # if  m_name == 'avg', we will append the model[i], w2v representation of w
ord i
        # if m_name == 'weighted', we will multiply each w2v[word] with the idf(wor
d)

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

    for word in sentence.split():
        nwords += 1
        if word in vocab:
            if m_name == 'weighted' and word in  idf_title_vectorizer.vocabulary_:
                featureVec = np.add(featureVec, idf_title_features[doc_id, idf_titl
e_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 sentance, its of shape (1, 300)
    return featureVec
```

In [10]:
```
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)
#print ("hi")
```

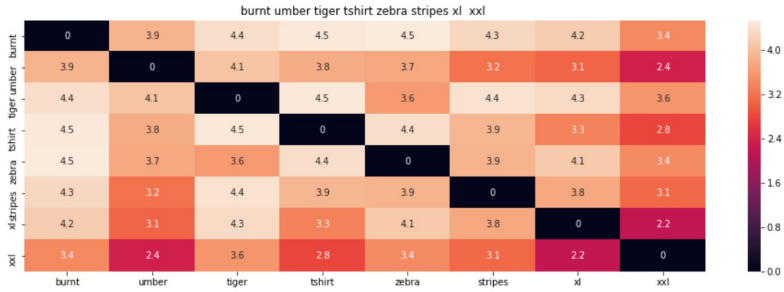
In [9]:
```
def display_img(url,ax,fig):
    # we get the url of the apparel and download it
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    # we will display it in notebook
    plt.imshow(img)
```

In [11]:
```
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 w
ord i
        # if m_name == 'weighted', we will multiply each w2v[word] with the idf(wor
d)
    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.vocabula
ry_[i]] * model[i])
            elif m_name == 'avg':
                vec.append(model[i])
        else:
            # if the word in our courpus is not there in the google word2vec corpus
, we are just ignoring it
            vec.append(np.zeros(shape=(300,)))
    # we will return a numpy array of shape (#number of words in title * 300 ) 300
= len(w2v_model[word])
    # each row represents the word2vec representation of each word (weighted/avg) i
n given sentance
    return  np.array(vec)

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

    final_dist = []
    # for each vector in vec1 we caluclate the distance(euclidean) to all vectors i
n vec2
    for i in vec1:
        dist = []
        for j in vec2:
            # np.linalg.norm(i-j) will result the euclidean distance between vector
s 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_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(weighte
d/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(weighte
d/avg) of length 300 corresponds to each word in give title
    s2_vec = get_word_vec(sentance2, doc_id2, model)
```

Text, brand and color features to recommend similar products

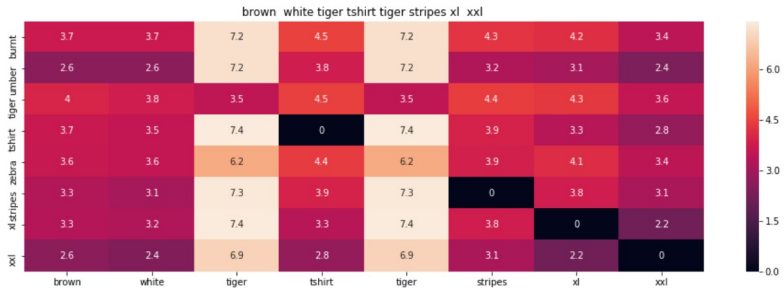
In [12]:
```
def idf_w2v_brand(doc_id, w1, w2, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand and color features

    # pairwise_dist will store the distance from given input apparel to all remaini
ng 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].r
eshape(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_in
dices[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i],df_i
ndices[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
B00JXQB5FQ	Si-Row	Brown
B00JXQB5FQ	Si-Row	Brown



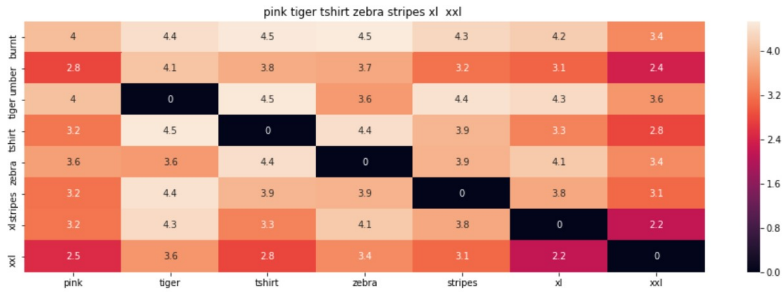
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.00034526698291301725
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JXQCWTO	Si-Row	Brown



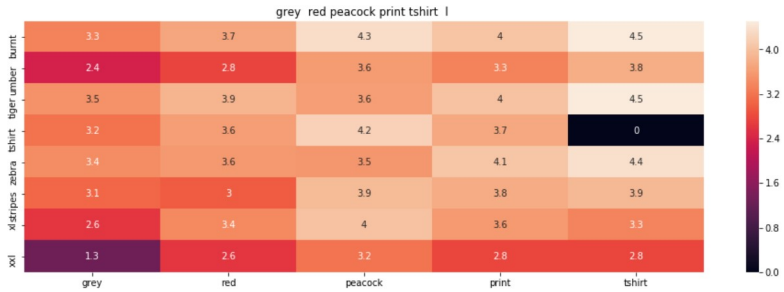
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 0.6528800964355469
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JXQASS6	Si-Row	Pink



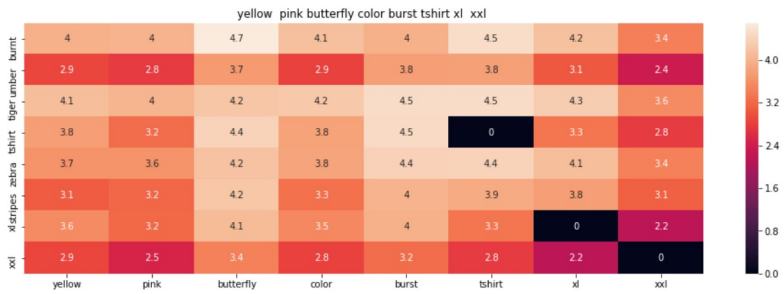
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 1.0017034055609373
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JXQCFRS	Si-Row	Grey



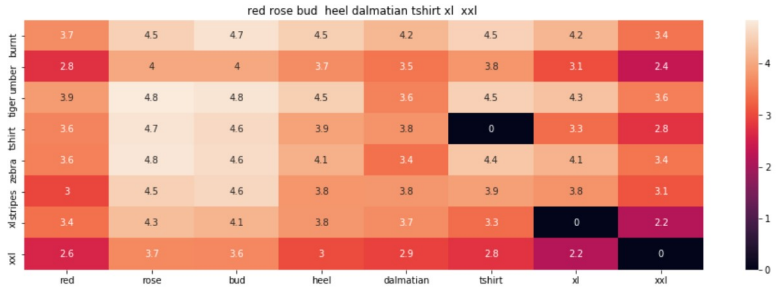
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 1.2372833730597166
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JXQBBMI	Si-Row	Yellow



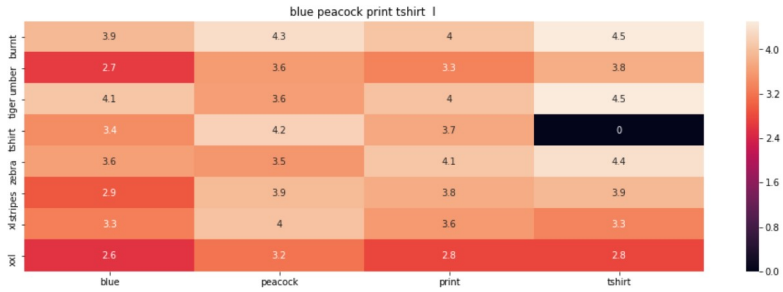
```
ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 1.268622684659448
================================================================================
==========================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JXQABB0	Si-Row	Red



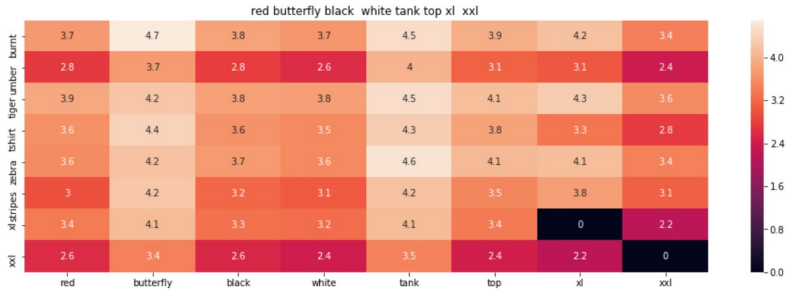
```
ASIN : B00JXQABB0
Brand : Si Row
euclidean distance from input : 1.2733485223669674
================================================================================
==========================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JXQC8L6	Si-Row	Blue



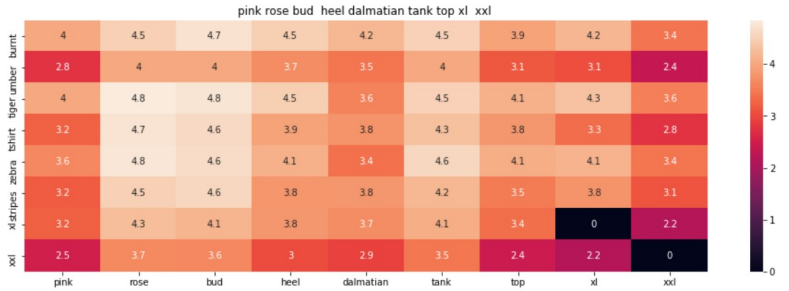
```
ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 1.2818686963934567
================================================================================
==========================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JV63CW2	Si-Row	Red



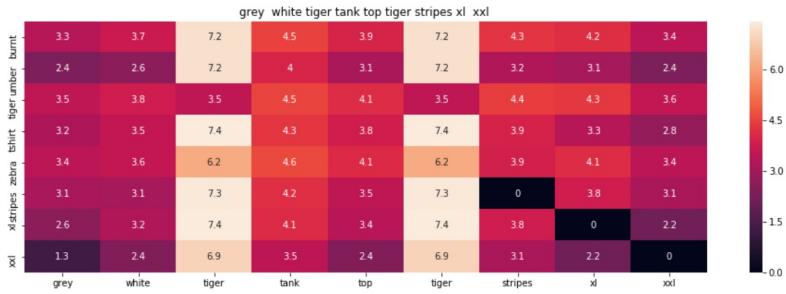
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 1.294748926343408
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JXQAX2C	Si-Row	Pink



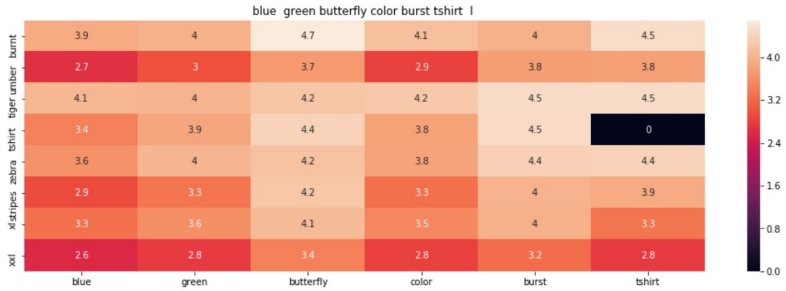
ASIN : B00JXQAX2C
Brand : Si Row
euclidean distance from input : 1.3236359598059324
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JXQAFZ2	Si-Row	Grey



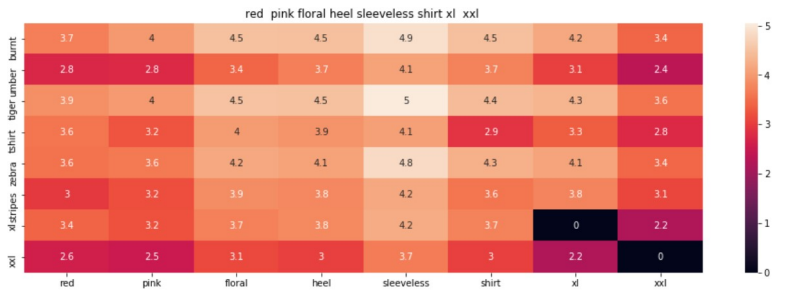
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 1.3293567659277585
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JXQC0C8	Si-Row	Blue



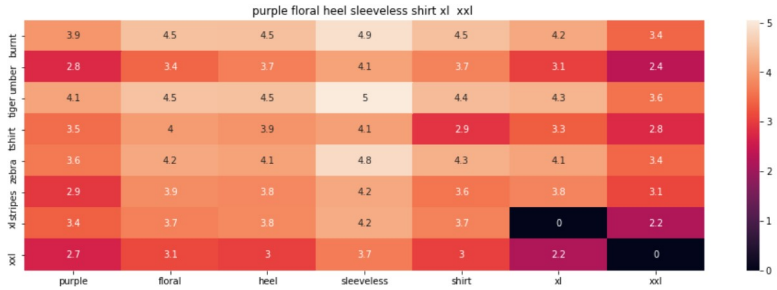
ASIN : B00JXQC0C8
Brand : Si Row
euclidean distance from input : 1.3499385358709959
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JV63QQE	Si-Row	Red



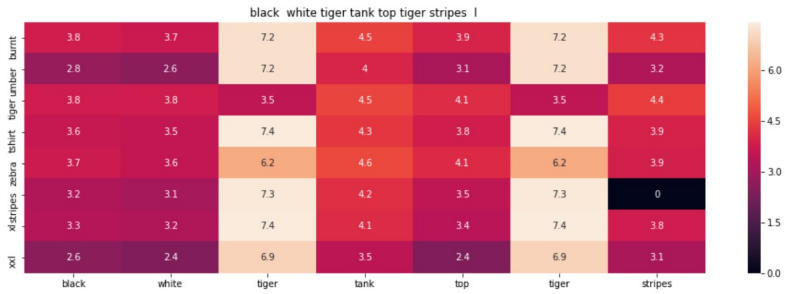
```
ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 1.364989996137109
================================================================================
==============================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JV63VC8	Si-Row	Purple



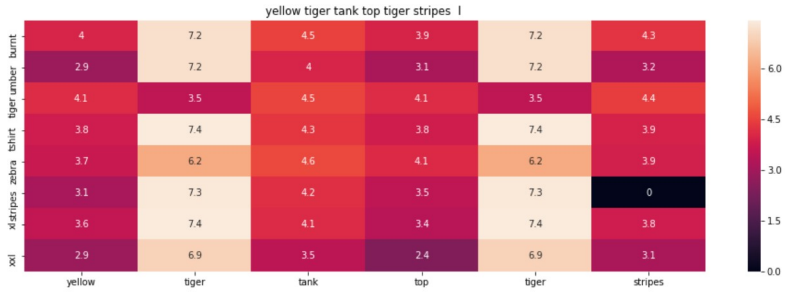
```
ASIN : B00JV63VC8
Brand : Si Row
euclidean distance from input : 1.408390236081567
================================================================================
==============================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JXQAO94	Si-Row	White



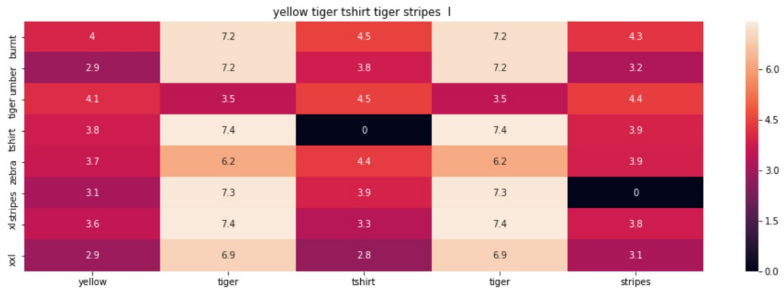
```
ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 1.5019503595251706
================================================================================
==============================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JXQAUWA	Si-Row	Yellow



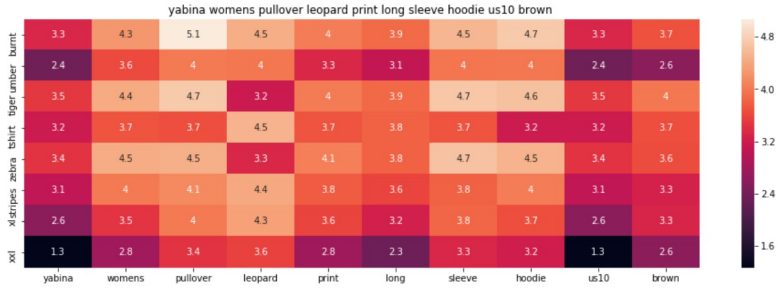
```
ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 1.5693790437597896
================================================================================
==========================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B00JXQCUIC	Si-Row	Yellow



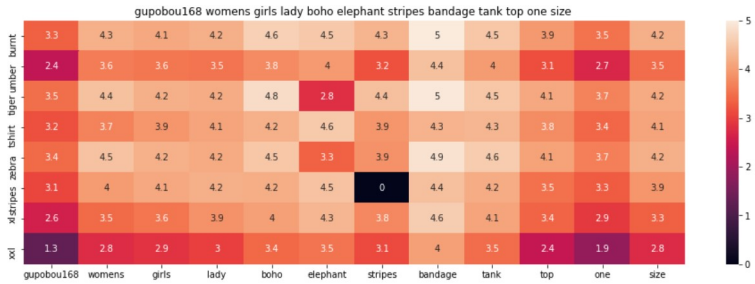
```
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 1.6333652498144773
================================================================================
==========================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B01KJUM6JI	YABINA	Brown



ASIN : B01KJUM6JI
Brand : YABINA
euclidean distance from input : 1.7323768689989183
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B01ER184O6	GuPoBoU168	Brown

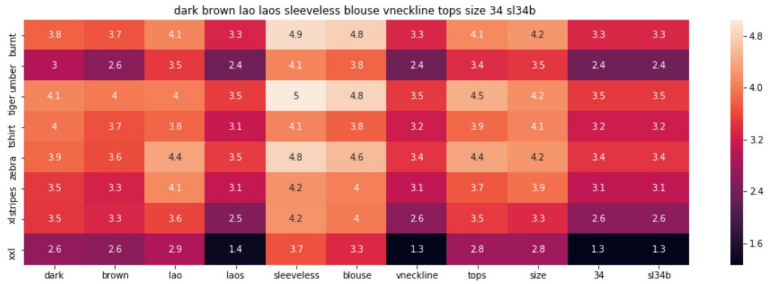


ASIN : B01ER184O6
Brand : GuPoBoU168
euclidean distance from input : 1.7352905347704026
==========
==========

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B074MH886R	Bobeau	Brown

```
ASIN : B074MH886R
Brand : Bobeau
euclidean distance from input : 1.7428852155565355
================================================================================
============================================
```

Asin	Brand	Color
B00JXQB5FQ	Si-Row	Brown
B074J48RGW	Nanon	Brown



```
ASIN : B074J48RGW
Brand : Nanon
euclidean distance from input : 1.7484709337114428
================================================================================
============================================
```

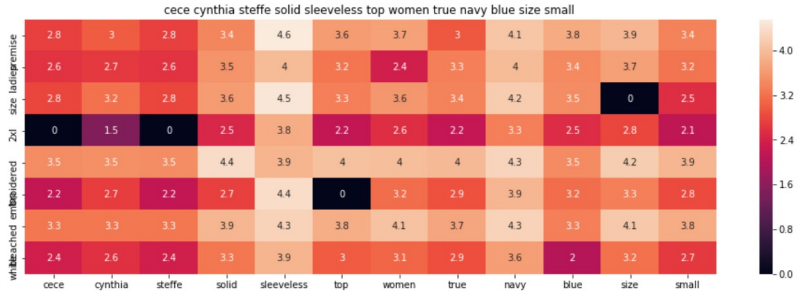
Text, brand, color and image features to recommend similar products

In [13]:
```
def idf_w2v_brand(doc_id, w1, w2, w3, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand and color features
    # w3: weight for image

    # pairwise_dist will store the distance from given input apparel to all remaini
ng 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].r
eshape(1,-1))
    ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id])
    doc_id = asins.index(df_asins[doc_id])
    img_dist = pairwise_distances(img_data, img_data[doc_id].reshape(1,-1))
    pairwise_dist   = (w1 * idf_w2v_dist +  w2 * ex_feat_dist + w3 * img_dist)/floa
t(w1 + w2 + w3)

    # 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_in
dices[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i],df_i
ndices[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, 7, 5, 10, 10)
# in the give heat map, each cell contains the euclidean distance between words i,
j
```

Asin	Brand	Color
B01M0IDUCV	Premise	Bleached-White
B01M0IDUCV	Premise	Bleached-White

ASIN : B01M0IDUCV
Brand : Premise
euclidean distance from input : 1.0875823716158999
==========
==========

Asin	Brand	Color
B01M0IDUCV	Premise	Bleached-White
B01N4NQ7LX	CeCe-by-Cynthia-Steffe	True-Navy

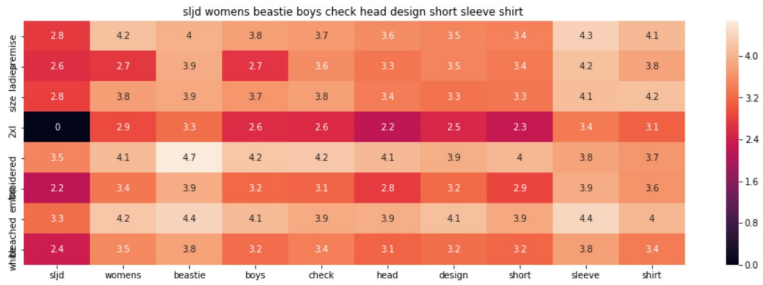


ASIN : B01N4NQ7LX
Brand : CeCe by Cynthia Steffe
euclidean distance from input : 14.846799927662005
==========
==========

Asin	Brand	Color
B01M0IDUCV	Premise	Bleached-White
B01IU645VU	Outback-Red	Brown-Stripe

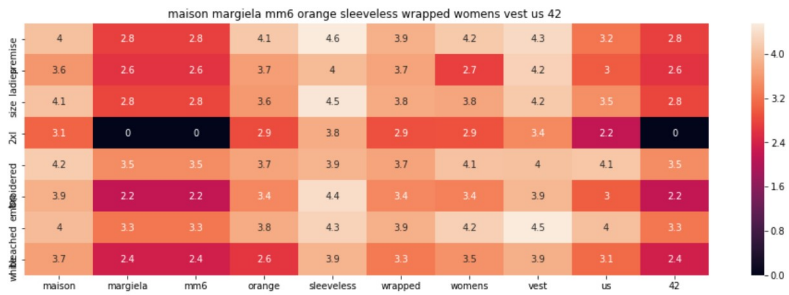
ASIN : B01IU645VU
Brand : Outback Red
euclidean distance from input : 19.76997139816953
==========
==========

Asin	Brand	Color
B01M0IDUCV	Premise	Bleached-White
B01FQLKKMK	SLJD	Grey



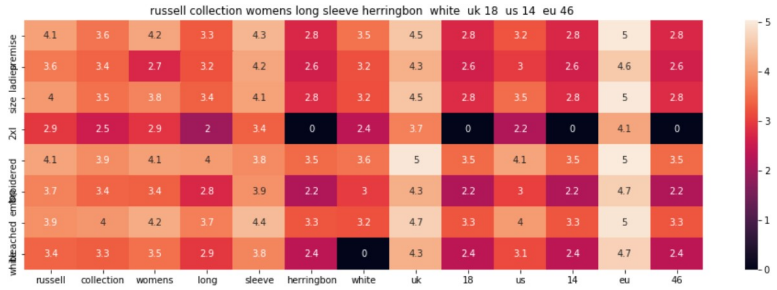
ASIN : B01FQLKKMK
Brand : SLJD
euclidean distance from input : 21.031672418497013
==========
==========

Asin	Brand	Color
B01M0IDUCV	Premise	Bleached-White
B01MXI5L4G	Maison-Margiela-MM6	Orange



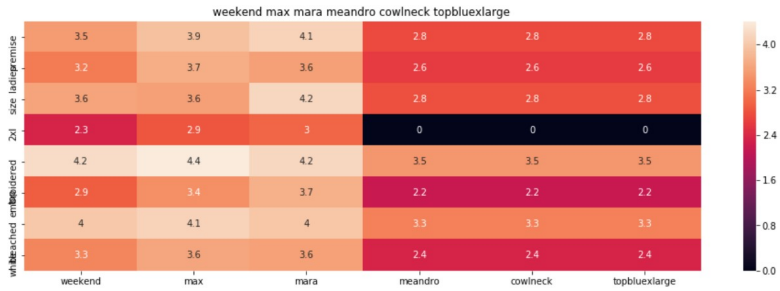
ASIN : B01MXI5L4G
Brand : Maison Margiela MM6
euclidean distance from input : 22.643057736483488
==========
==========

Asin	Brand	Color
B01M0IDUCV	Premise	Bleached-White
B00K77AN5S	Russell-Collection	White



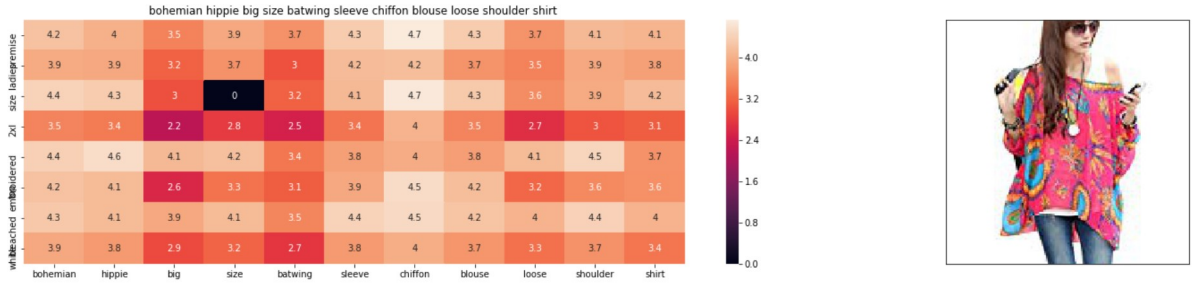
ASIN : B00K77AN5S
Brand : Russell Collection
euclidean distance from input : 22.73682546632027
==========
==========

Asin	Brand	Color
B01M0IDUCV	Premise	Bleached-White
B01MG83UB4	MaxMara	Blue



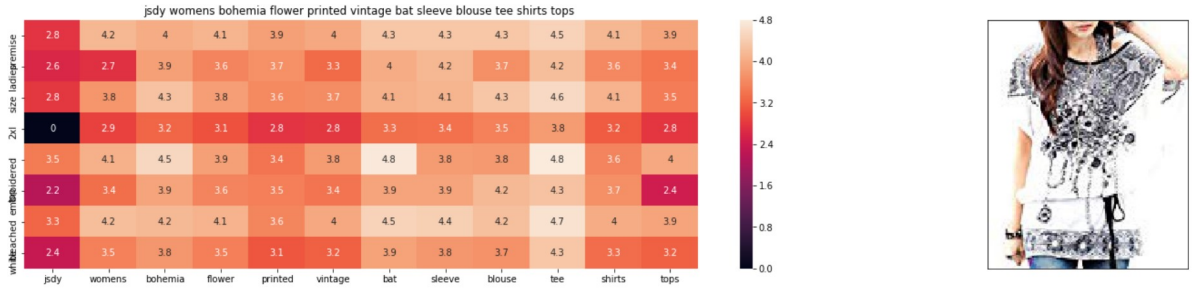
ASIN : B01MG83UB4
Brand : MaxMara
euclidean distance from input : 22.840919998938574
==========
==========

Asin	Brand	Color
B01M0IDUCV	Premise	Bleached-White
B00YC92VRU	Display-Promotion	R



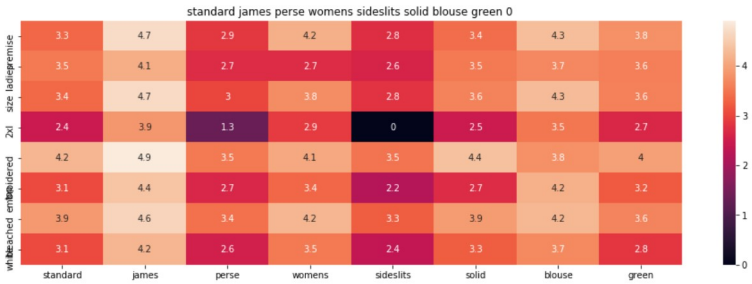
ASIN : B00YC92VRU
Brand : Display Promotion
euclidean distance from input : 22.927506734403885
==========
==========

Asin	Brand	Color
B01M0IDUCV	Premise	Bleached-White
B00L8RE3PC	JSDY-Cloth	White



ASIN : B00L8RE3PC
Brand : JSDY-Cloth
euclidean distance from input : 22.979918956920855
==========
==========

Asin	Brand	Color
B01M0IDUCV	Premise	Bleached-White
B071LMW4YG	Standard-James-Perse	Green



```
ASIN : B071LMW4YG
Brand : Standard James Perse
euclidean distance from input : 22.981746500188653
================================================================================
==========================================
```

Summary and Observation

1.Used Text, brand, color and image features to recommend similar products.
2.Given weights feature for all 4 features and played around with it, to see how image recommendations vary based on features.

In [ ]: