RecommendationSystem

July 9, 2018

1 Now Takling About project

In this project we are Taking the amazon data for recommendation.

The model we have used here is Simple Weighted Euclidean distances

Now coming to the time consuming code-

1.) Brute-Force algorithm- 1-2:30 Hrs.

2.) Image-to dense vector Using Keras and Tensorflow as backend - 12+ hrs for an i5 core processor.

3.) and other code to execeute time arround 1 hr.

The Overall project is Divided into 5 Phases.

1.)Importing libraries and prepariing dataset.

2.)Missing data values for various features.

3.)Missing data values for various features.

4.)Data Pre processing using Stop word removal.

5.)Creating the IDF weighted Word2Vec Title ,Brand and Color using One Hot Encoder and Visual by VGG16 CovNet.

Install all nessary packages using pip in your windows using command prompt.

2 [1]Importing Libraries and Preparing Dataset

```
In [21]: from PIL import Image as PilImage
         import itertools
         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
from sklearn.feature_extraction.text import CountVectorizer
import pickle
#Deep-Learning Library
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
from keras import applications
import plotly
import plotly.figure_factory as ff
from plotly.graph_objs import Scatter, Layout

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

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

```
In [2]: # we have give a json file which consists of all information about
        # the products
        # loading the data using pandas' read_json file.
        data = pd.read_json('tops_fashion.json')
```

```
In [3]: print ("Number of Data Points :", data.shape[0])
        print ("Number of Features /Variables:" ,data.shape[1])
```

```
Number of Data Points : 183138
Number of Features /Variables: 19
```

```
In [4]: data.columns
```

```
Out[4]: 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 project. 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 apperal, ex: SHIRT/TSHIRT) 5. medium_image_url (url of the image) 6. title (title of the product.) 7. formatted_price (price of the product)

```
In [5]: data= data[['asin', 'brand', 'color', 'medium_image_url', 'product_type_name', 'title'
        print ("Number of Data Points :", data.shape[0])
        print ("Number of Features /Variables:" ,data.shape[1])
        data.head()

Number of Data Points : 183138
Number of Features /Variables: 7
```

Out[5]:

	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

3 [2]Missing data values for various features

Now we will find the basic stats for each various feature for Data cleaning and storing it

[2.1]Basic stats for the feature: product_type_name

```
In [6]: print(data['product_type_name'].describe())

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

```
In [7]: # names of different product types
        print(data['product_type_name'].unique())
        # find the 10 most frequent product_type_names.
```

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

```
['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']
```

```
Out[7]: [('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)]
```

[2.2]Basic stats for the feature: brand

```
In [8]: print(data['brand'].describe())
```

```
count     182987
unique     10577
top         Zago
freq          223
Name: brand, dtype: object
```

As we can see that there are only 10577 unique brands and total missing brands are 183138-182987=151

```
In [9]: brand_count = Counter(list(data['brand']))
        brand_count.most_common(10)
```

```
Out[9]: [('Zago', 223),
         ('XQS', 222),
         ('Yayun', 215),
         ('YUNY', 198),
         ('XiaoTianXin-women clothes', 193),
         ('Generic', 192),
         ('Boohoo', 190),
         ('Alion', 188),
         ('Abetteric', 187),
         ('TheMogan', 187)]
```

Thge Most common Brand is Zago

[2.3]Basic stats for the feature: color

```
In [10]: print(data['color'].describe())
         color_count = Counter(list(data['color']))
         color_count.most_common(10)

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


Out[10]: [(None, 118182),
          ('Black', 13207),
          ('White', 8616),
          ('Blue', 3570),
          ('Red', 2289),
          ('Pink', 1842),
          ('Grey', 1499),
          ('*', 1388),
          ('Green', 1258),
          ('Multi', 1203)]
```

As we can see 7380 are unique product.
64956 of 183138 products have color information. That's approx 35.4%.
7.2% of products are black in color

[2.4] Basic stats for the feature: formatted_price

```
In [11]: print(data['formatted_price'].describe())
         price_count = Counter(list(data['formatted_price']))
         price_count.most_common(10)

count      28395
unique      3135
```

```
top       $19.99
freq         945
Name: formatted_price, dtype: object


Out[11]: [(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)]
```

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

[2.5]Basic stats for the feature : title

```
In [12]: print(data['title'].describe())

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


In [13]: data.to_pickle('pickels/180k_apparel_data')
```

To reduce this large data we will take formatted price such that our null entries are removed for better data frame

```
In [14]: 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
```

Now taking color in it for removing null in the color section

```
In [15]: data = data.loc[~data['color'].isnull()]
         print('Number of data points After eliminating price=NULL :', data.shape[0])

Number of data points After eliminating price=NULL : 28385


In [16]: data.to_pickle('pickels/28k_apparel_data')
```

I have reduced the to 28K because i want to understand some basics operation on small data first so 28k is enough for understanding it .For those who want to work on the 180K data can work on it.I have commented out the code for image 28K data so those who are usiung this can download it from here.

In [17]:
```
'''
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[17]:
```
"\nfrom PIL import Image\nimport requests\nfrom io import BytesIO\n\nfor index, row in
```

4 [3] Removing the duplicated Items by Title name

[3.1] Understanding the Duplicates

In [18]:
```
data = pd.read_pickle('pickels/28k_apparel_data')
```

In [19]:
```
print(sum(data.duplicated('title')))
```

```
2325
```

we have 2325 products which have same title but different color

[3.2] Remove Duplicates : Part 1

In [20]:
```
data = pd.read_pickle('pickels/28k_apparel_data')
data = data[['asin', 'brand', 'color', 'medium_image_url', 'product_type_name', 'title
data.head()
```

Out[20]:

	asin	brand	color	\
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	
11	B001LOUGE4	Fitness Etc.	Black	
15	B003BSRPB0	FeatherLite	White	
21	B014ICEDNA	FNC7C	Purple	

	medium_image_url	product_type_name	\
4	https://images-na.ssl-images-amazon.com/images...	SHIRT	

```
6   https://images-na.ssl-images-amazon.com/images...          SHIRT
11  https://images-na.ssl-images-amazon.com/images...          SHIRT
15  https://images-na.ssl-images-amazon.com/images...          SHIRT
21  https://images-na.ssl-images-amazon.com/images...          SHIRT

                                                  title formatted_price
4   Featherlite Ladies' Long Sleeve Stain Resistan...          $26.26
6   Women's Unique 100% Cotton T - Special Olympic...           $9.99
11            Ladies Cotton Tank 2x1 Ribbed Tank Top          $11.99
15  FeatherLite Ladies' Moisture Free Mesh Sport S...          $20.54
21  Supernatural Chibis Sam Dean And Castiel Short...           $7.50
```

```
In [21]: # 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 [22]: data_sorted.sort_values('title',inplace=True, ascending=False)
         data_sorted.head()
```

Out[22]:

```
              asin     brand        color  \
61973   B06Y1KZ2WB    Éclair   Black/Pink
133820  B010RV33VE  xiaoming         Pink
81461   B01DDSDLNS  xiaoming        White
75995   B00X5LYO9Y  xiaoming  Red Anchors
151570  B00WPJG35K  xiaoming        White

                                        medium_image_url product_type_name  \
61973   https://images-na.ssl-images-amazon.com/images...            SHIRT
133820  https://images-na.ssl-images-amazon.com/images...            SHIRT
81461   https://images-na.ssl-images-amazon.com/images...            SHIRT
75995   https://images-na.ssl-images-amazon.com/images...            SHIRT
151570  https://images-na.ssl-images-amazon.com/images...            SHIRT

                                                  title formatted_price
61973   Éclair Women's Printed Thin Strap Blouse Black...          $24.99
133820  xiaoming Womens Sleeveless Loose Long T-shirts...          $18.19
81461   xiaoming Women's White Long Sleeve Single Brea...          $21.58
75995   xiaoming Stripes Tank Patch/Bear Sleeve Anchor...          $15.91
151570  xiaoming Sleeve Sheer Loose Tassel Kimono Woma...          $14.32
```

Some examples of dupliacte titles that differ only in the last few words. Titles 1: 16. woman's place is in the house and the senate shirts for Womens XXL White 17. woman's place is in the house and the senate shirts for Womens M Grey

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

```
In [23]: indices = []
         for i,row in data_sorted.iterrows():
             indices.append(i)
         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'
             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', 'Qu
                 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 string
                 # example: a =['a', 'b', 'c', 'd']
                 # b = ['a', 'b', 'd']
                 # itertools.zip_longest(a,b): will give [('a','a'), ('b','b'), ('c','d'), ('d
                 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 consid
                 # if the number of words in which both strings differ are < 2 , we are consid
                 if (length - count) > 2: # number of words in which both sensences differ
                     # if both strings are differ by more than 2 words we include the 1st stri
                     stage1_dedupe_asins.append(data_sorted['asin'].loc[indices[i]])

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

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

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

```
In [24]: data = data.loc[data['asin'].isin(stage1_dedupe_asins)]
         print('Number of data points : ', data.shape[0])
```

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

```
In [25]: data.to_pickle('pickels/17k_apperal_data')
```

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

The Method below mentioned is brute-force attack method

```
In [4]: data=pd.read_pickle('pickels/17k_apperal_data')
        # O(n^2) time.
        # Takes about an hour to run on a decent computer.

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

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

                b = data['title'].loc[j].split()
                # store the list of words of jth string in b, ex: b = ['tokidoki', 'The', 'Que

                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
                # example: a =['a', 'b', 'c', 'd']
                # b = ['a', 'b', 'd']
                # itertools.zip_longest(a,b): will give [('a','a'), ('b','b'), ('c','d'), ('d'
                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 conside
                if (length - count) < 3:
                    indices.remove(j)
        data = data.loc[data['asin'].isin(stage2_dedupe_asins)]
        print('Number of data points after stage two of dedupe: ',data.shape[0])
        data.to_pickle('pickels/16k_apperal_data')
```

```
Number of data points after stage two of dedupe:  16435
```

```
In [5]: data.head()
```

Out[5]:

	asin	brand	color	\
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	
15	B003BSRPB0	FeatherLite	White	
27	B014ICEJ1Q	FNC7C	Purple	
46	B01NACPBG2	Fifth Degree	Black	

	medium_image_url	product_type_name	\
4	https://images-na.ssl-images-amazon.com/images...	SHIRT	
6	https://images-na.ssl-images-amazon.com/images...	SHIRT	
15	https://images-na.ssl-images-amazon.com/images...	SHIRT	
27	https://images-na.ssl-images-amazon.com/images...	SHIRT	
46	https://images-na.ssl-images-amazon.com/images...	SHIRT	

	title	formatted_price
4	Featherlite Ladies' Long Sleeve Stain Resistan...	$26.26
6	Women's Unique 100% Cotton T - Special Olympic...	$9.99
15	FeatherLite Ladies' Moisture Free Mesh Sport S...	$20.54
27	Supernatural Chibis Sam Dean And Castiel O Nec...	$7.39
46	Fifth Degree Womens Gold Foil Graphic Tees Jun...	$6.95

5 [4] Data Pre processing using Stop word removal

```
In [6]: data = pd.read_pickle('pickels/16k_apperal_data')
        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: {"aren't", 'having', 'd', "you've", "you'll", 'it', 'but', 'than', "shouldn
```

```
In [7]: 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")
```

```
6.43211533216225 seconds
```

```
In [8]: data.head()
```

Out[8]:

	asin	brand	color	\
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	
15	B003BSRPB0	FeatherLite	White	
27	B014ICEJ1Q	FNC7C	Purple	
46	B01NACPBG2	Fifth Degree	Black	

	medium_image_url	product_type_name	\
4	https://images-na.ssl-images-amazon.com/images...	SHIRT	
6	https://images-na.ssl-images-amazon.com/images...	SHIRT	
15	https://images-na.ssl-images-amazon.com/images...	SHIRT	
27	https://images-na.ssl-images-amazon.com/images...	SHIRT	
46	https://images-na.ssl-images-amazon.com/images...	SHIRT	

	title	formatted_price
4	featherlite ladies long sleeve stain resistant...	$26.26
6	womens unique 100 cotton special olympics wor...	$9.99
15	featherlite ladies moisture free mesh sport sh...	$20.54
27	supernatural chibis sam dean castiel neck tshi...	$7.39
46	fifth degree womens gold foil graphic tees jun...	$6.95

```
In [9]: data.to_pickle('pickels/16k_apperal_data_preprocessed')
```

```
In [10]: data=pd.read_pickle('pickels/16k_apperal_data_preprocessed')
         print(data.shape[0])
```

```
16435
```

```
In [11]: title_vectorizer = CountVectorizer()
         title_features   = title_vectorizer.fit_transform(data['title'])
         title_features.get_shape()
```

```
Out[11]: (16435, 12684)
```

As in the above we can see that 16435 are titles and 12684 are the word in document corpus

6 [5] Creating the IDF weighted Word2Vec Title ,Brand and Color using One Hot Encoder and Visual by VGG16 CovNet.

6.1 [5.1] Defining The Features

6.1.1 [5.1.1] Idf Title Features and W2V Title Features

```
In [3]: data=pd.read_pickle('pickels/16k_apperal_data_preprocessed')
        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 #
        # idf_title_features[doc_id, index_of_word_in_corpus] = number of times the word occur
```

6.1.2 [5.1.2] Defining the Features for Brands and Color

```
In [4]: # some of the brand values are empty.
        # Need to replace Null with string "NULL"
        data['brand'].fillna(value="Not given", inplace=True )

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

        colors = [x.replace(" ", "-") for x in data['color'].values]

        #One-Hot Encoding using Count vectorizer
        brand_vectorizer = CountVectorizer()
        brand_features = brand_vectorizer.fit_transform(brands)

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

6.1.3 [5.1.3] Defining the Features for Visuals

```
In [5]: #load the features and corresponding ASINS info.
        bottleneck_features_train = np.load('16k_data_cnn_features.npy')
        asins = np.load('16k_data_cnn_feature_asins.npy')
        asins = list(asins)

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

6.2 [5.2] Utility Functions for the IDF Word2Vec Weighted Title

```
In [12]: with open('word2vec_model', 'rb') as handle:
             model = pickle.load(handle)
         # vocab = stores all the words that are there in google w2v model
         vocab = model.keys()
         doc_id = 0
         w2v_title_weight = []
         # for every title 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 d
         w2v_title_weight = np.array(w2v_title_weight)
```

```
In [6]: def n_containing(word):
            # return the number of documents which had the given word
            return sum(1 for blob in data['title'] if word in blob.split())

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

```
In [7]: # 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 will find its idf value
            idf_val = idf(i)

            # to calculate idf_title_features need to replace the count values with the idf va
            # idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0] will ret
            for j in idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0]:

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

```
In [8]: 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
                # 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_
                    elif m_name == 'avg':
                        vec.append(model[i])
                else:
                    # if the word in our courpus is not there in the google word2vec corpus,ju
                    vec.append(np.zeros(shape=(300,)))
            # return a numpy array of shape (#number of words in title * 300 ) 300 = len(w2v_m
            # each row represents the word2vec representation of each word (weighted/avg) in g
            return  np.array(vec)

In [10]: def get_distance(vec1, vec2):
             # vec1 = np.array(#number_of_words_title1 * 300), each row is a vector of length
             # vec2 = np.array(#number_of_words_title2 * 300), each row is a vector of length

             final_dist = []
             # for each vector in vec1  caluclate the distance(euclidean) to all vectors in ve
             for i in vec1:
                 dist = []
                 for j in vec2:
                     # np.linalg.norm(i-j) will result the euclidean distance between vectors
                     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 [9]: # this function will add the vectors of each word and returns the avg vector of given
        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',append the model[i], w2v representation of word i
                # if m_name == 'weighted',multiply each w2v[word] with the idf(word)

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

```
            for word in sentence.split():
                nwords += 1
                if word in vocab:
                    if m_name == 'weighted' and word in  idf_title_vectorizer.vocabulary_:
                        featureVec = np.add(featureVec, idf_title_features[doc_id, idf_title_ve
                    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
```

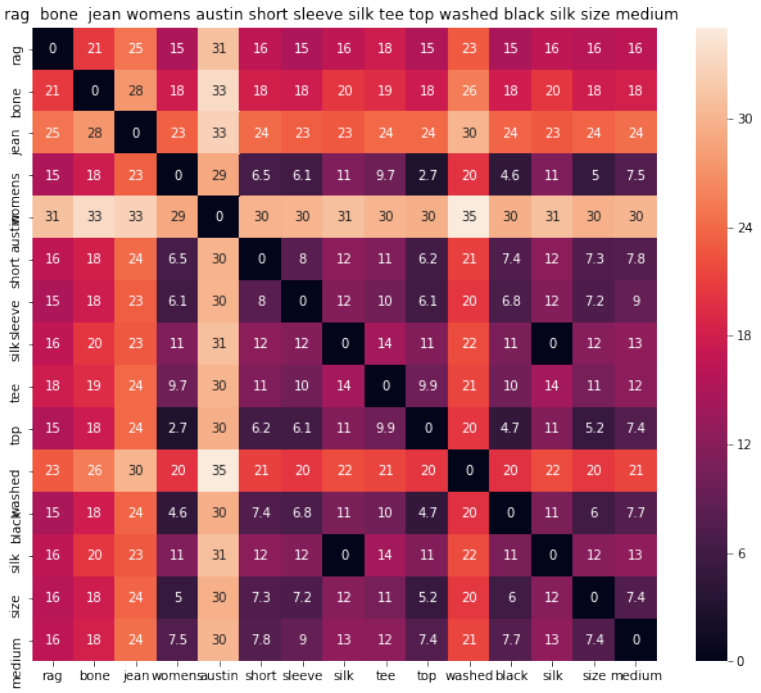
```
In [22]: def display_img(url,ax,fig):
             #get the url of the apparel and download it
             response = requests.get(url)
             img = PilImage.open(BytesIO(response.content))
             #display it in notebook
             plt.imshow(img)
```

```
In [13]: #The Below code is is for the weighted IDF Word2Vec
         def heat_map_w2v(sentence1, sentence2, url, doc_id1, doc_id2, model):
             # sentance1 : title1, input apparel
             # sentance2 : title2, recommended apparel
             # url: apparel image url
             # doc_id1: document id of input apparel
             # doc_id2: document id of recommended apparel
             # model: it can have two values, 1. avg 2. weighted

             #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/
             s1_vec = get_word_vec(sentence1, doc_id1, model)
             #s2_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/
             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 im
             gs = gridspec.GridSpec(2, 2, width_ratios=[4,1],height_ratios=[2,1])
             fig = plt.figure(figsize=(15,15))

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

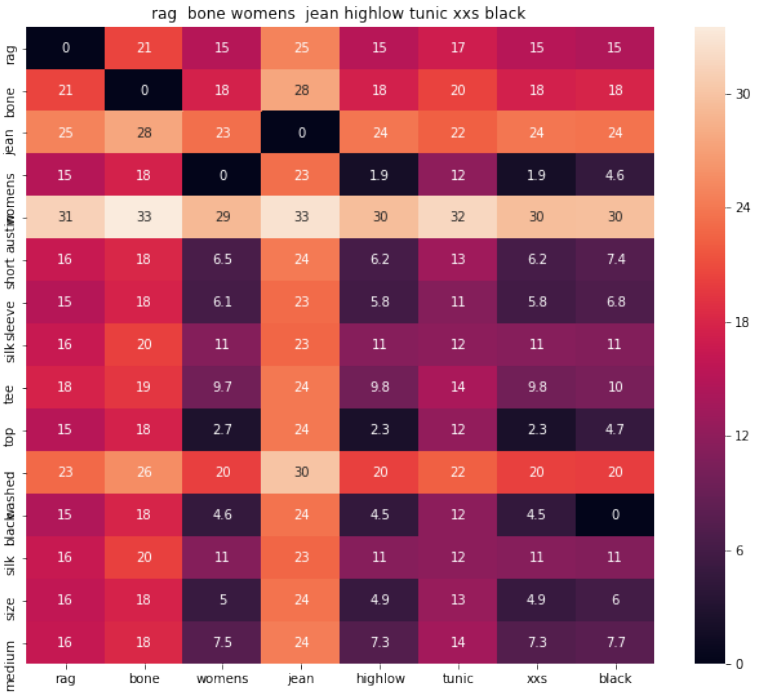
```
            ax.set_xticklabels(sentence2.split())
            # set the y axis labels as input apparels title
            ax.set_yticklabels(sentence1.split())
            # set title as recommended apparels title
            ax.set_title(sentence2)

            ax = plt.subplot(gs[1])
            # 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 [22]:
```
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
    # the metric used here is cosine, the coside distance is mesured as K(X, Y) = <X,
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].resh

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

In [23]:
```
weighted_w2v_model(12577,20)
```

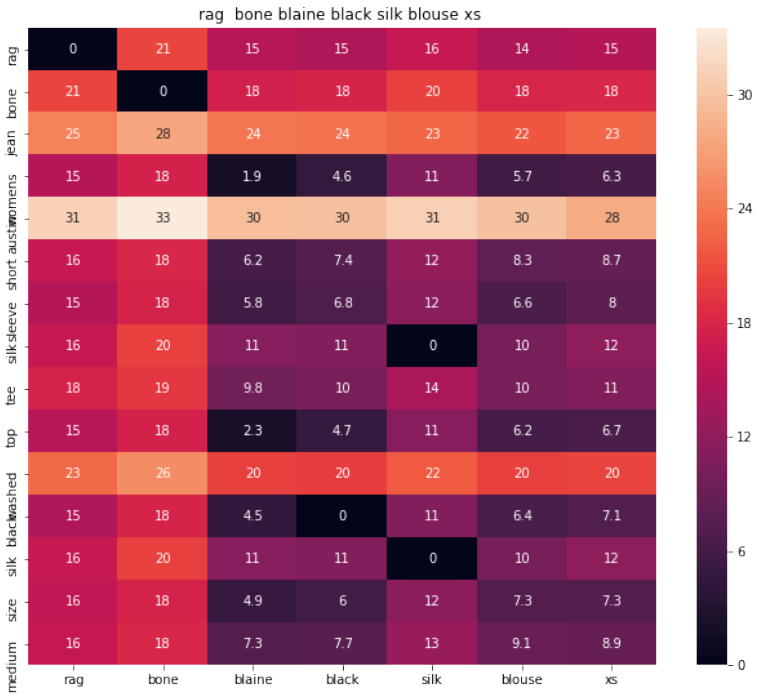


ASIN : B06XDYB1TX
Brand : rag & bone
euclidean distance from input : 0.001953125
==========

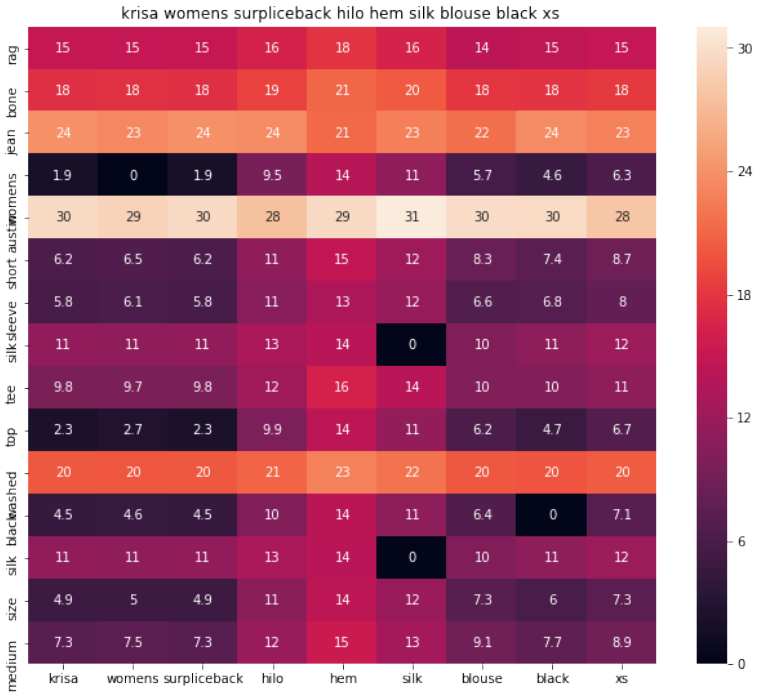


ASIN : B06XCZFCWM
Brand : rag & bone
euclidean distance from input : 3.3960376
==========

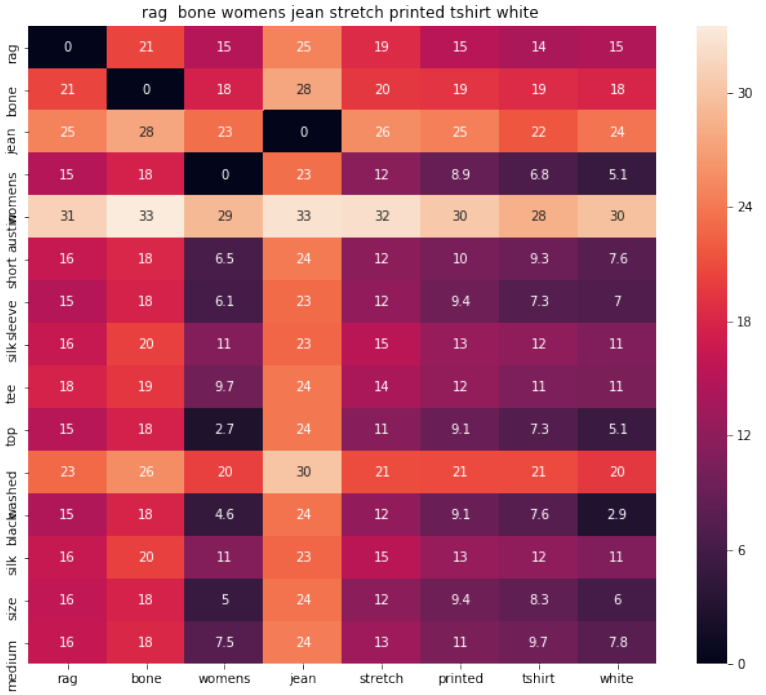
ASIN : B01N34YASX
Brand : Rag & Bone/JEAN
euclidean distance from input : 3.3979177
==========



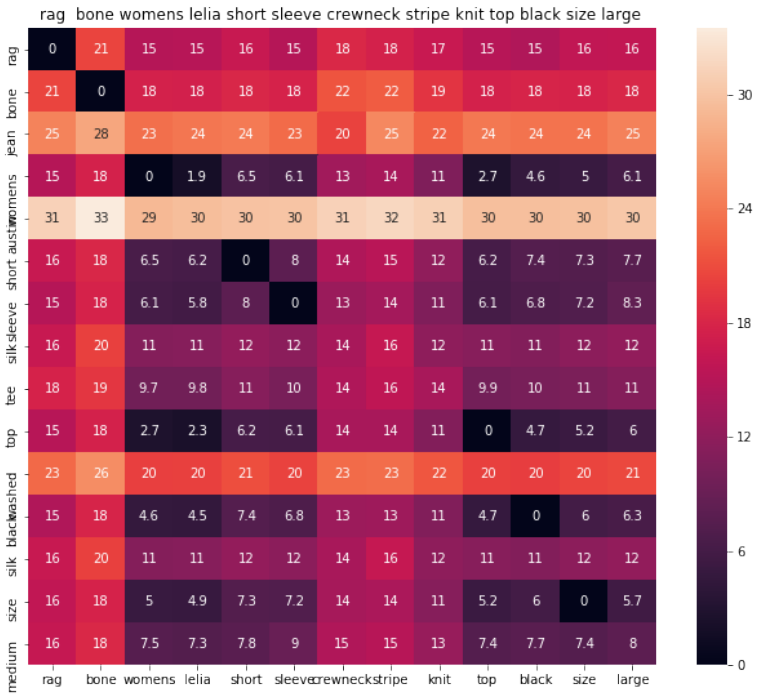
ASIN : B072LDKGBB
Brand : rag & bone
euclidean distance from input : 3.4275532
==========



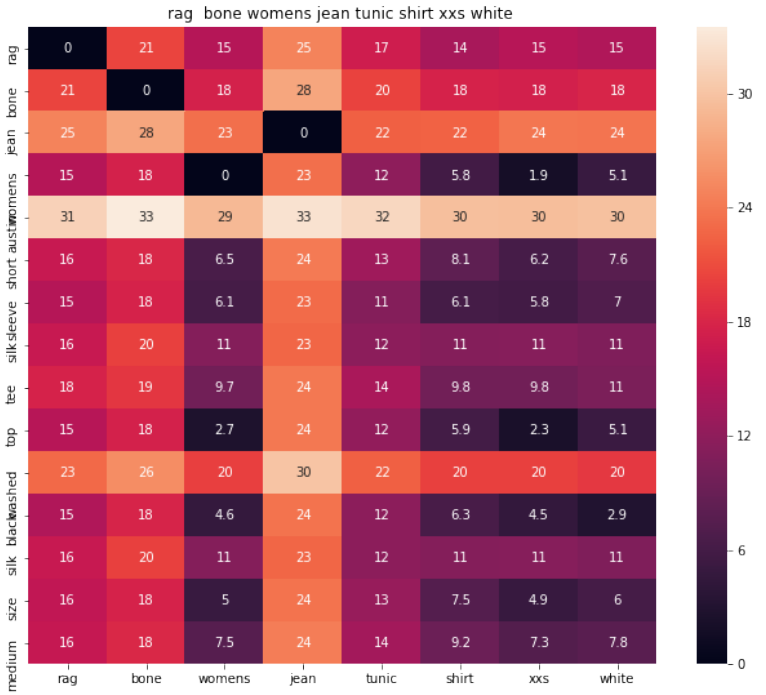
ASIN : B0719R5YZ5
Brand : Krisa
euclidean distance from input : 3.5234778
==========



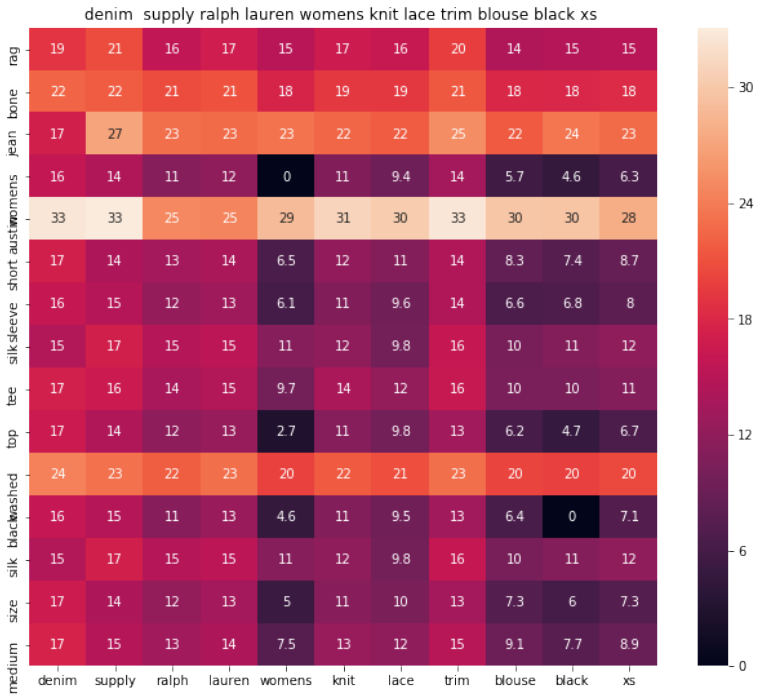
ASIN : B01F49HHVQ
Brand : rag & bone
euclidean distance from input : 3.5468042
==========



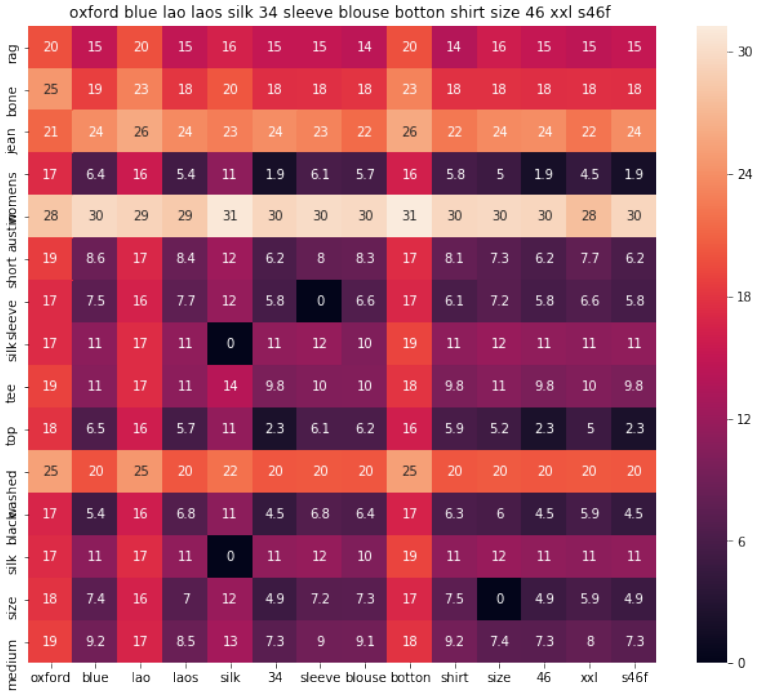
ASIN : B06XDW4KZH
Brand : rag & bone
euclidean distance from input : 3.5594802
==========



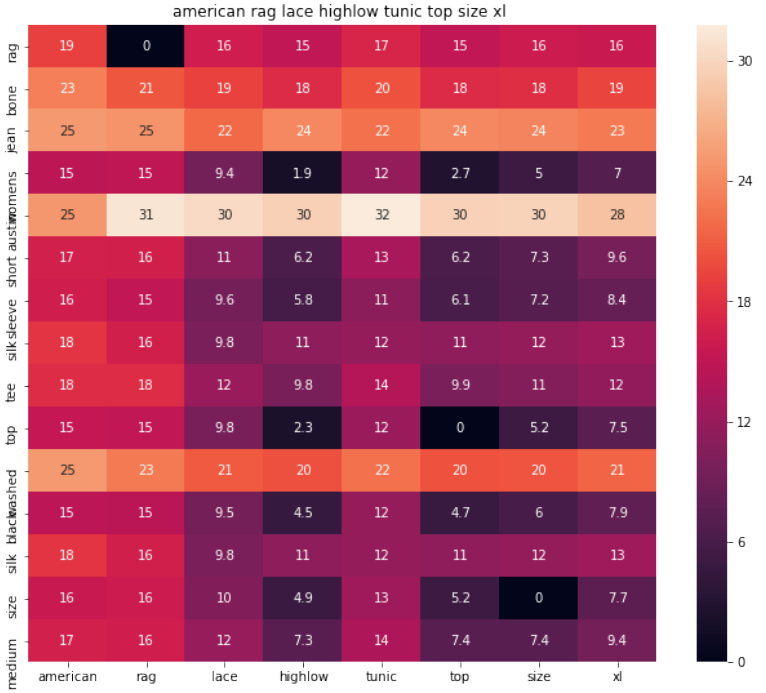
ASIN : B06XCSHG55
Brand : rag & bone
euclidean distance from input : 3.5883954
==========



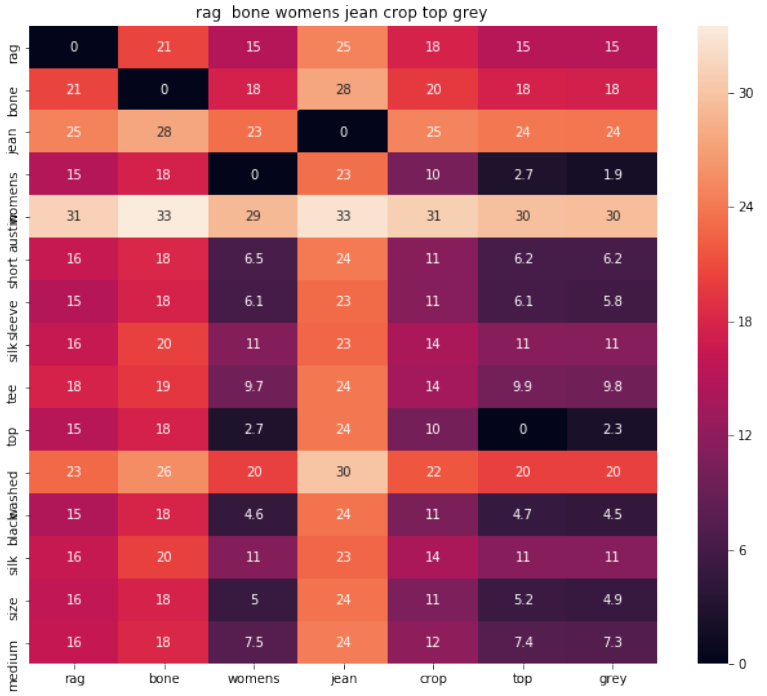
ASIN : B073HV3FFQ
Brand : Denim & Supply
euclidean distance from input : 3.6181843
==========



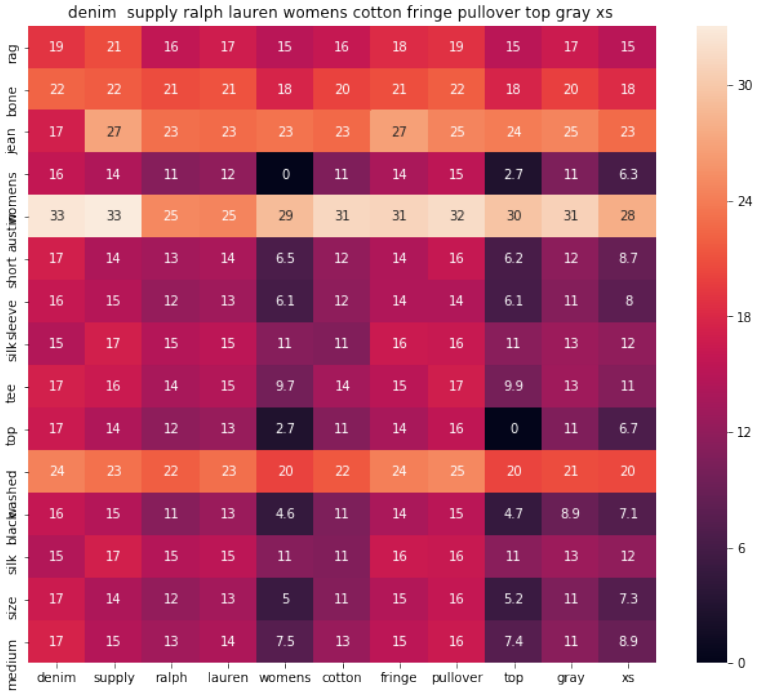
ASIN : B074FRG8D8
Brand : Nanon
euclidean distance from input : 3.654277
==========



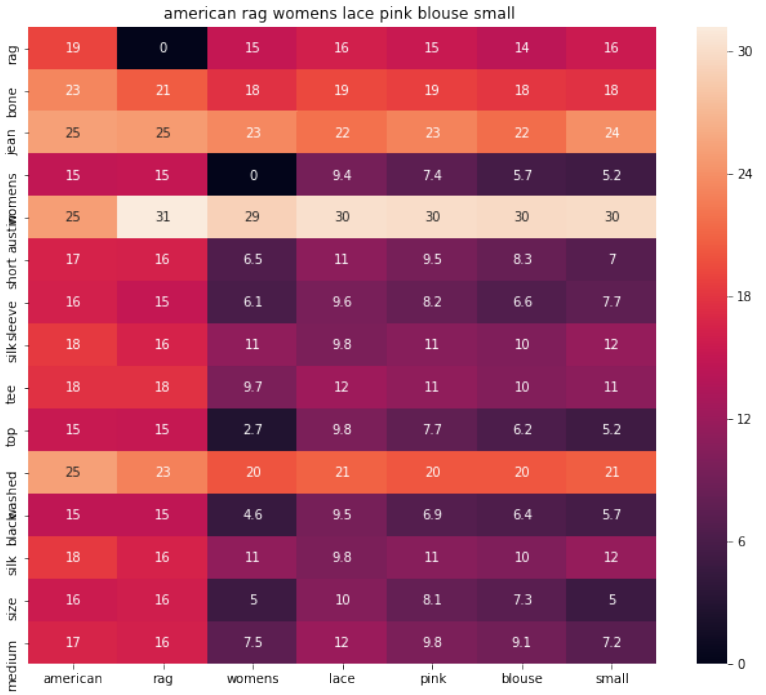
ASIN : B06XY5Z1JB
Brand : American Rag
euclidean distance from input : 3.6587949
==========



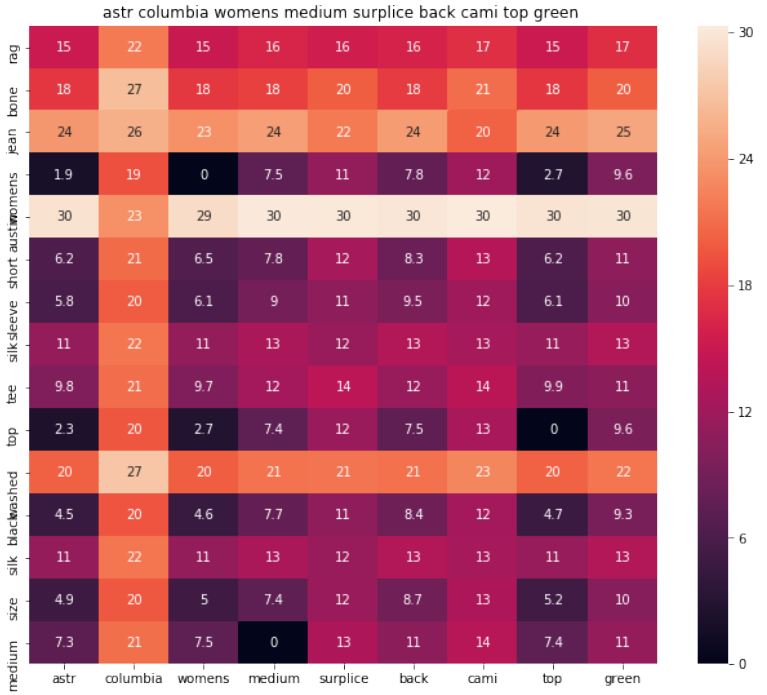
ASIN : B06XD37PKL
Brand : rag & bone
euclidean distance from input : 3.6722715
==========



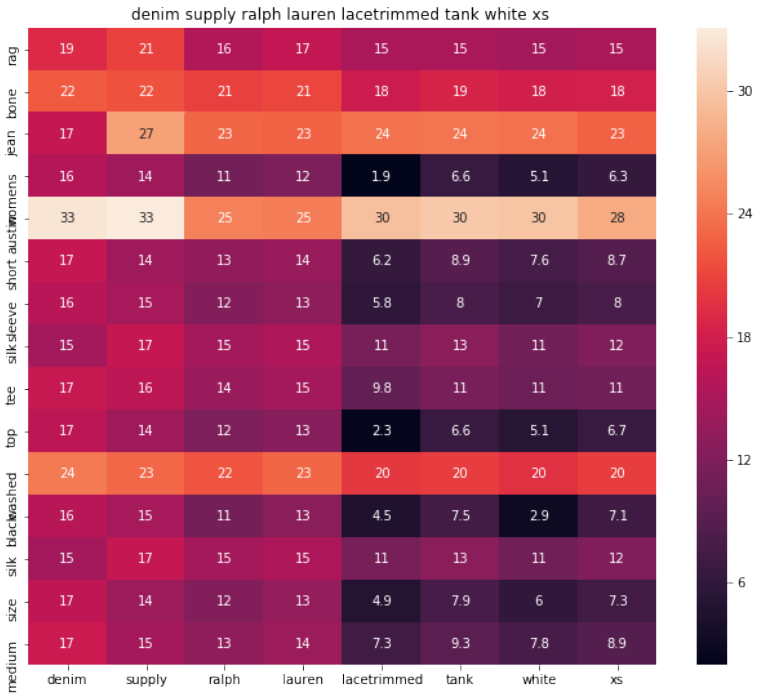
ASIN : B01C0BMJBO
Brand : Denim & Supply
euclidean distance from input : 3.6744053
==========



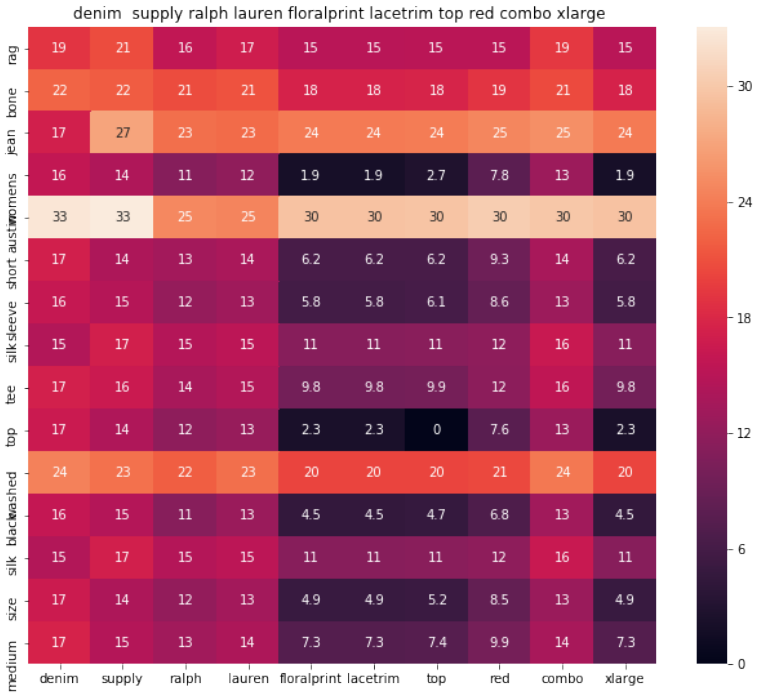
ASIN : B01MYLTZJT
Brand : American Rag
euclidean distance from input : 3.7340558
==========



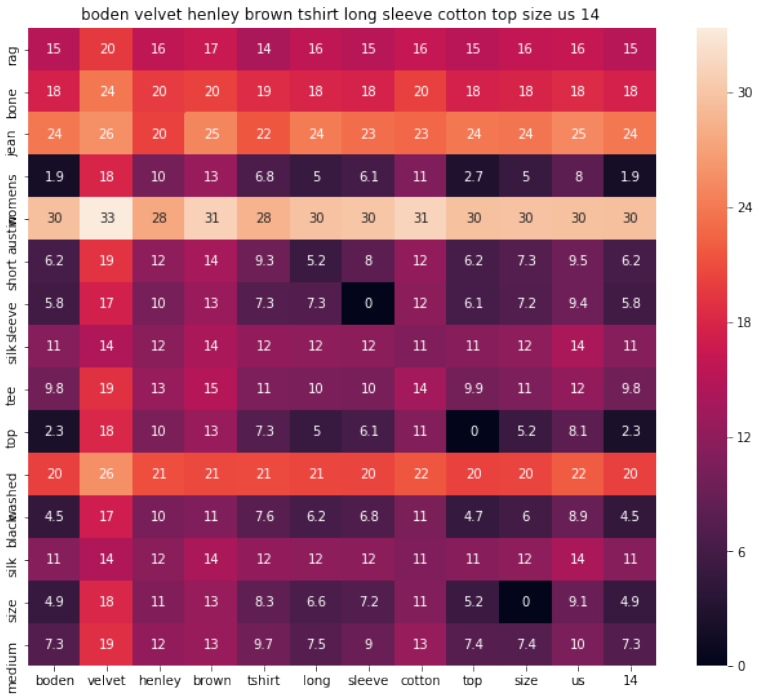
ASIN : B073P8C2M1
Brand : ASTR
euclidean distance from input : 3.7445295
==========



ASIN : B01N5TTQN0
Brand : Denim Supply ralph Lauren
euclidean distance from input : 3.7508209
==========



ASIN : B01MQWZVG4
Brand : Polo Ralph Lauren
euclidean distance from input : 3.7508419
==========



ASIN : B072DRJ5PK
Brand : BODEN
euclidean distance from input : 3.7553468
==========

ASIN : B00WT79CAQ
Brand : ARMANI JEANS
euclidean distance from input : 3.7587252
==========

```
ASIN : B00Z55WXFU
Brand : Denim & Supply
euclidean distance from input : 3.7611432
```

6.3 [5.3] Using VGG16 ConvNet ,Keras and Tensorflow for Image

```
In [2]: # 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 = 16435
        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 //
            bottleneck_features_train = bottleneck_features_train.reshape((16435,25088))

            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()
```

```
Found 16435 images belonging to 1 classes.
```

```
In [3]: #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_t

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

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

        get_similar_products_cnn(13577, 20)
```

```
Product Title:  lacoste womens womens navy sleeveles pique polo size 463xl navy
Euclidean Distance from input image: 0.054126587
Amazon Url: www.amzon.com/dp/B01M3VNRG1
```

Product Title: lacoste womens womens white pique polo size 46xxxl white
Euclidean Distance from input image: 20.777796
Amazon Url: www.amzon.com/dp/B01M2AAJNP

Product Title: alc womens nina top 4 blue
Euclidean Distance from input image: 33.190117
Amazon Url: www.amzon.com/dp/B01MPX3OQJ

Product Title: alc womens margaret silkunderlayer top 4 black
Euclidean Distance from input image: 33.683064
Amazon Url: www.amzon.com/dp/B01M7U5M9S

Product Title: belle badgley mischka womens silkblend sweater l black
Euclidean Distance from input image: 34.59951
Amazon Url: www.amzon.com/dp/B01N3W4IDC

Product Title: alice olivia womens eldridge tank black
Euclidean Distance from input image: 34.601215
Amazon Url: www.amzon.com/dp/B071G2YTJ4

Product Title: st john womens silkblend top p
Euclidean Distance from input image: 34.716682
Amazon Url: www.amzon.com/dp/B0753QB584

Product Title: six crisp days womens side slit top xss
Euclidean Distance from input image: 35.042706
Amazon Url: www.amzon.com/dp/B06XCS9KLC

Product Title: splendid womens double strap cami xl blue
Euclidean Distance from input image: 35.76045
Amazon Url: www.amzon.com/dp/B0746STR6M

Product Title: trina turk womens hani silkblend top xs blue
Euclidean Distance from input image: 36.125244
Amazon Url: www.amzon.com/dp/B07575F1NR

Product Title: koral womens activewear move crop top xs
Euclidean Distance from input image: 36.15733
Amazon Url: www.amzon.com/dp/B071XD14DR

Product Title: esley womens scalloped blouse l
Euclidean Distance from input image: 36.16053
Amazon Url: www.amzon.com/dp/B01M716OVC

Product Title: rag bone womens twist back tank top sz l navy black heather 270874dh
Euclidean Distance from input image: 36.512947
Amazon Url: www.amzon.com/dp/B073WMBN94

Product Title: sandro womens epply lace top 2 white
Euclidean Distance from input image: 36.69804
Amazon Url: www.amzon.com/dp/B06XKVT9ZT

Product Title: sadie sage womens aline top black
Euclidean Distance from input image: 37.03688
Amazon Url: www.amzon.com/dp/B074MJJ2X2

Product Title: bar iii womens solid racerback casual top black
Euclidean Distance from input image: 37.04073
Amazon Url: www.amzon.com/dp/B0724ZCX9F

Product Title: susana monaco womens harper top green
Euclidean Distance from input image: 37.213726
Amazon Url: www.amzon.com/dp/B0753P4BC6

Product Title: lacoste womens womens navy cotton polo size 42xl blue
Euclidean Distance from input image: 37.301964
Amazon Url: www.amzon.com/dp/B01MG65GD0

Product Title: joan vass womens tank 0
Euclidean Distance from input image: 37.329704
Amazon Url: www.amzon.com/dp/B06XXFBSP9

```
Product Title:  inc womens cutout split neck casual top blue l
Euclidean Distance from input image: 37.338505
Amazon Url: www.amzon.com/dp/B01NAST7MR
```

6.4 [5.4] Required Heat-Map and Weighted Euclidean Distance Method

```
In [14]: def heat_map_w2v_brand(sentance1, sentance2, url, doc_id1, doc_id2, df_id1, df_id2, mo

             # 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 tw o values, 1. avg 2. weighted

             #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/
             s1_vec = get_word_vec(sentance1, doc_id1, model)
             #s2_vec = np.array(#number_of_words_title2 * 300), each row is a vector(weighted/
             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'],
                        [data['asin'].loc[df_id1],brands[doc_id1], colors[doc_id1]], # input ap
                        [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2],]] # recomm

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

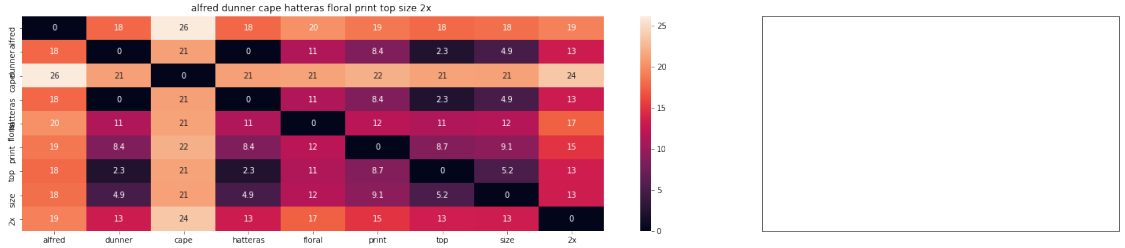
```
        # we create a table with the data_matrix
        table = ff.create_table(data_matrix, index=True, colorscale=colorscale)
        # plot it with plotly
        plotly.offline.iplot(table, filename='simple_table')

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

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

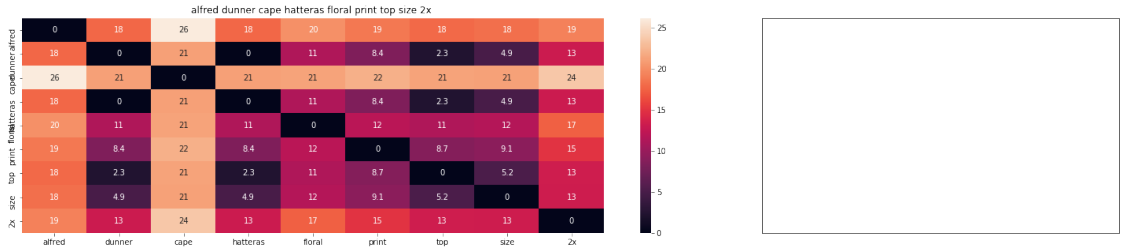
In [15]:
```
def idf_w2v_brand_image(doc_id, wt, wb,wc,wv,num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand and color features

    # pairwise_dist will store the distance from given input apparel to all remaining
    # the metric we used here is cosine, the coside distance is mesured as K(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].resh
    brand_feat_dist = pairwise_distances(brand_features, brand_features[doc_id])
    color_feat_dist = pairwise_distances(color_features, color_features[doc_id])
    visual_dist = pairwise_distances(bottleneck_features_train, bottleneck_features_tr
    pairwise_dist   = (wt * idf_w2v_dist +  wb * brand_feat_dist+wc*color_feat_dist+wv

    # np.argsort will return indices of 9 smallest distances
```

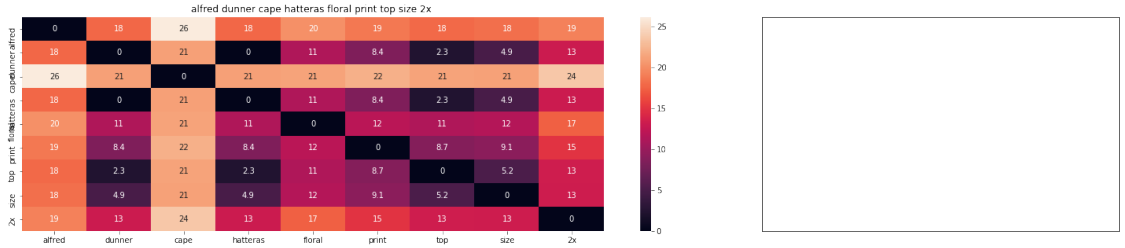
```
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

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


# in the give heat map, each cell contains the euclidean distance between words i, j
```

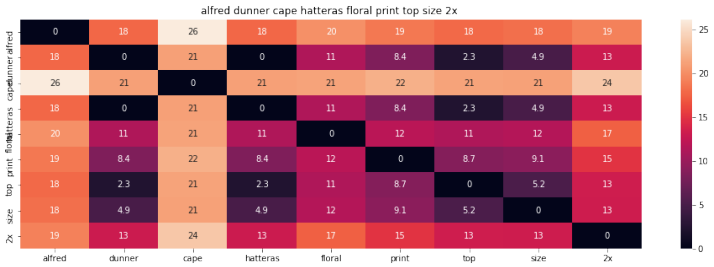
A sample for Title preference

```
In [23]: idf_w2v_brand_image(15698, 30, 5,1,4,25)
```

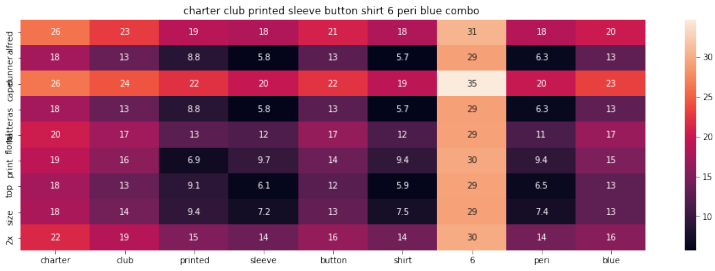




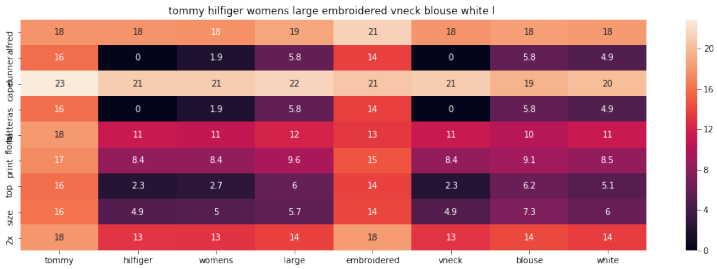




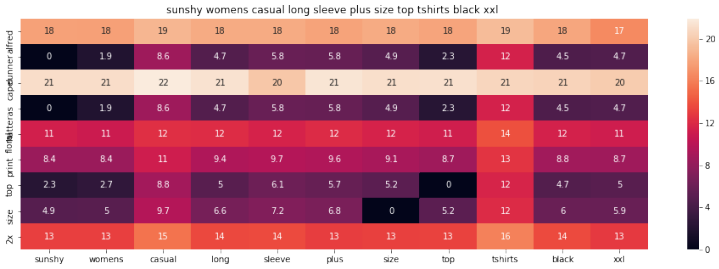
ASIN : B01JKGBK4Y
Brand : Alfred Dunner
euclidean distance from input : 0.0020716018974781037
==========



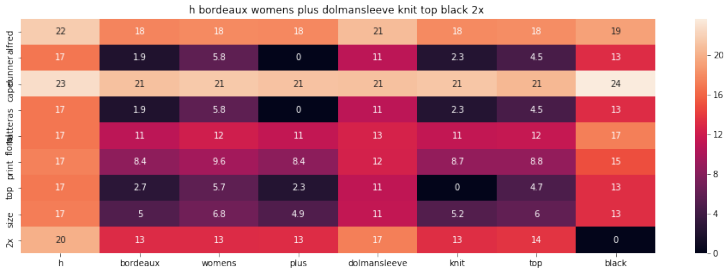
ASIN : B0758N8Q3R
Brand : Charter Club
euclidean distance from input : 4.005362701416016
==========



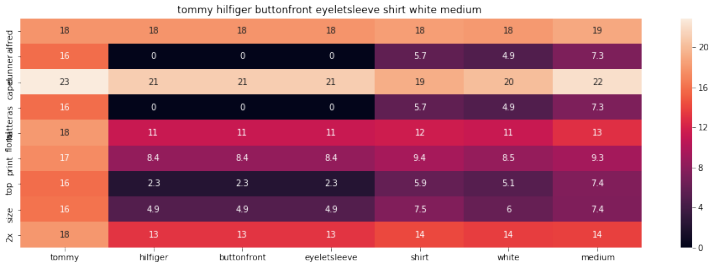
ASIN : B071F5G7D6
Brand : Tommy Hilfiger
euclidean distance from input : 8.427404937753176
==========



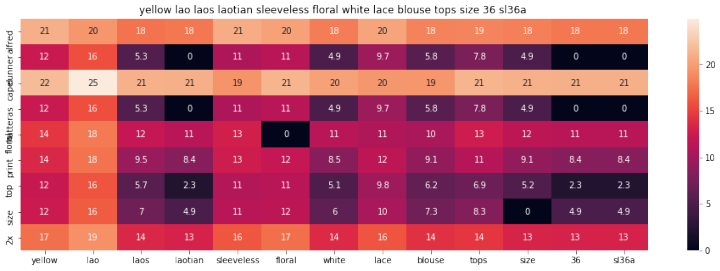
ASIN : B018EB3Q34
Brand : Juntong
euclidean distance from input : 8.552170871218816
==========



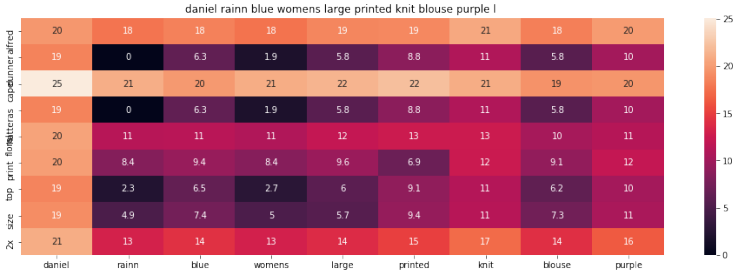
ASIN : B071VV7XSW
Brand : H By Bordeaux
euclidean distance from input : 8.552759513864014
==========



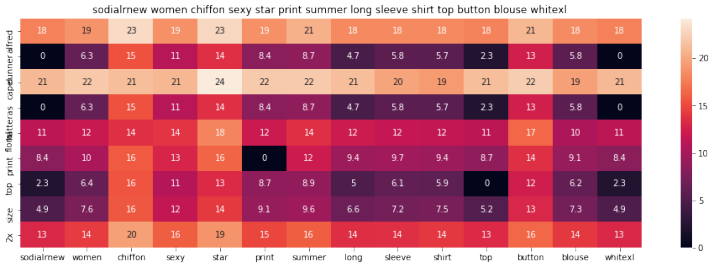
ASIN : B06XXL5H9Z
Brand : Tommy Hilfiger
euclidean distance from input : 8.562561378488038
==========



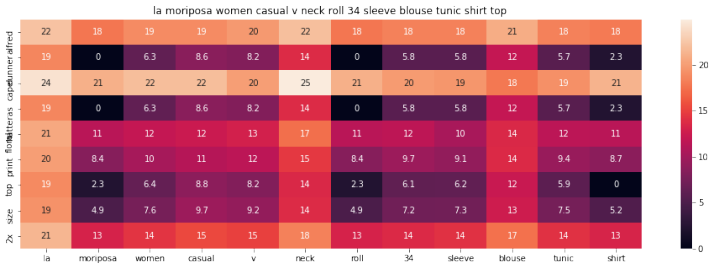
ASIN : B074J781N9
Brand : Nanon
euclidean distance from input : 8.610631297549382
==========



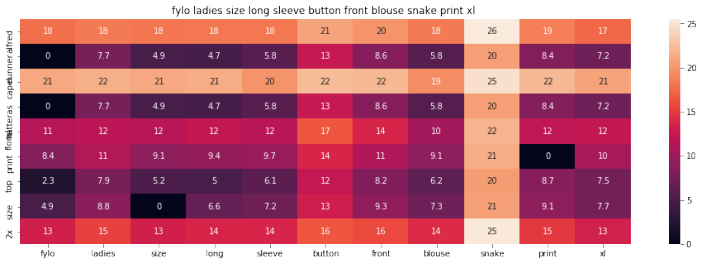
ASIN : B074VD5TNB
Brand : Daniel Rainn
euclidean distance from input : 8.640774116525147
==========



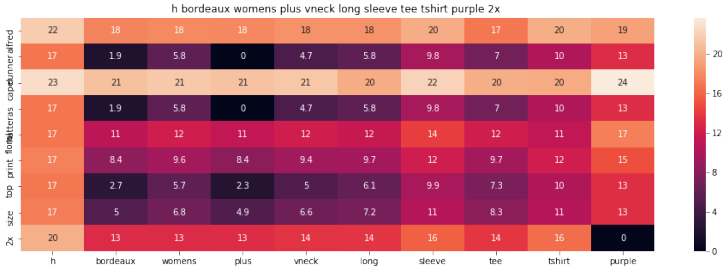
ASIN : B013OD2T26
Brand : SODIAL(R)
euclidean distance from input : 8.651548694094794
==========



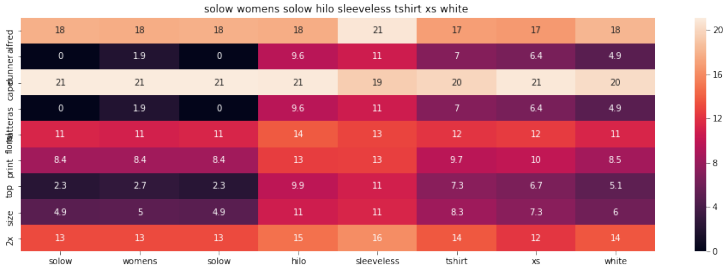
ASIN : B01K1O3X0I
Brand : La moriposa
euclidean distance from input : 8.65254474640796
==========



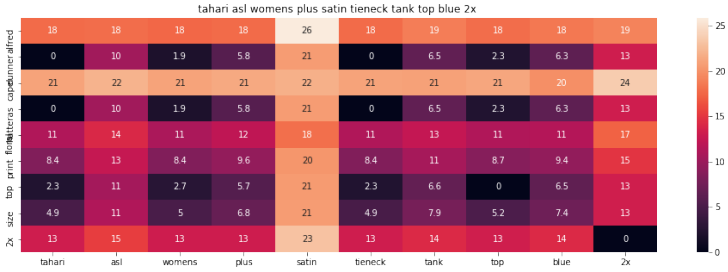
ASIN : B06Y2KFPQV
Brand : Fylo
euclidean distance from input : 8.655061184062578
==========



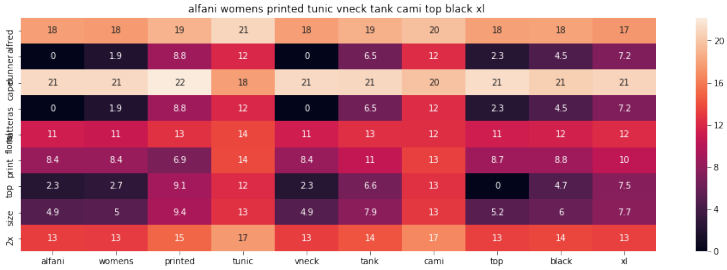
ASIN : B071ZDSGZ8
Brand : H By Bordeaux
euclidean distance from input : 8.65816684723804
==========



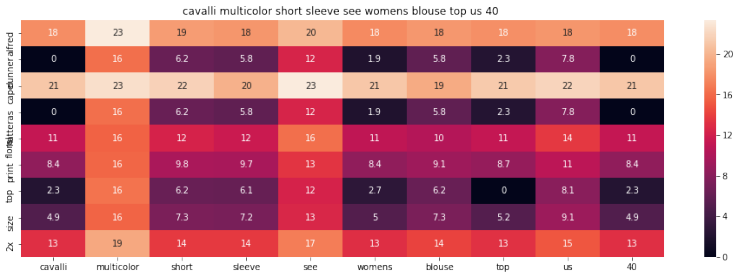
ASIN : B01M292SR6
Brand : SOLOW
euclidean distance from input : 8.665025256594792
==========



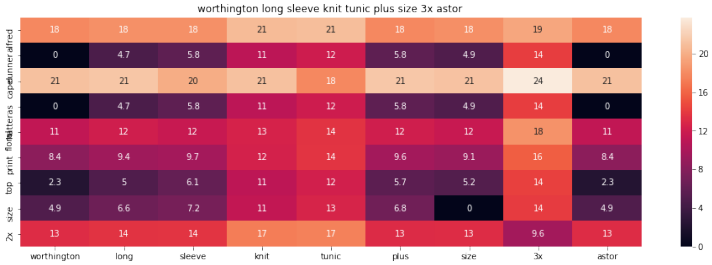
ASIN : B06WLMWCS7
Brand : Tahari by Arthur S. Levine
euclidean distance from input : 8.668673591208982
==========



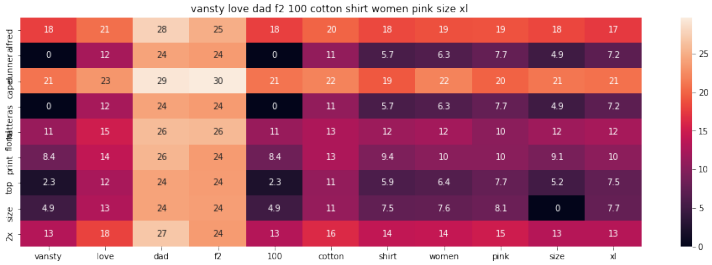
ASIN : B0721QB1JW
Brand : Alfani
euclidean distance from input : 8.67163612695612
==========



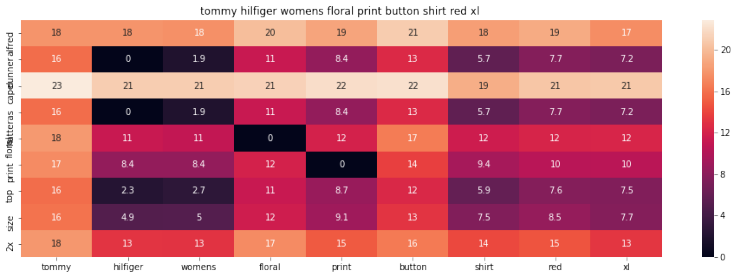
ASIN : B01HSKDJXY
Brand : Just Cavalli
euclidean distance from input : 8.675534057617188
==========



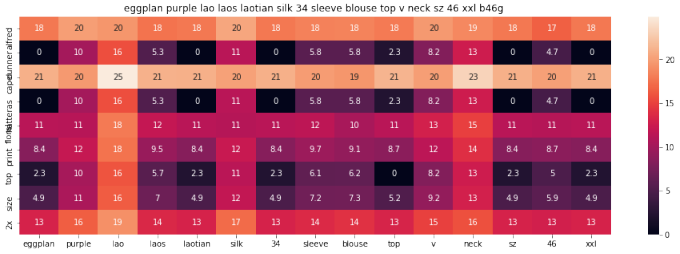
ASIN : B06Y4JS88M
Brand : Worthington
euclidean distance from input : 8.68925968499152
==========



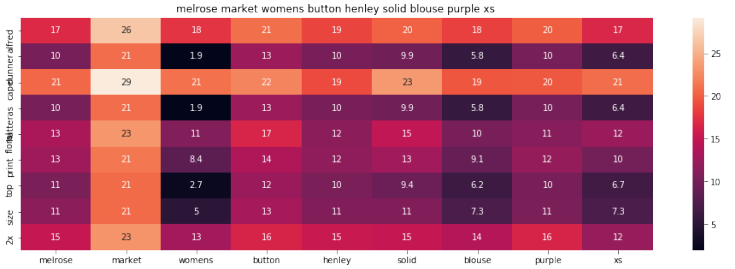
ASIN : B01EJS5AIK
Brand : Vansty
euclidean distance from input : 8.697496531924383
==========



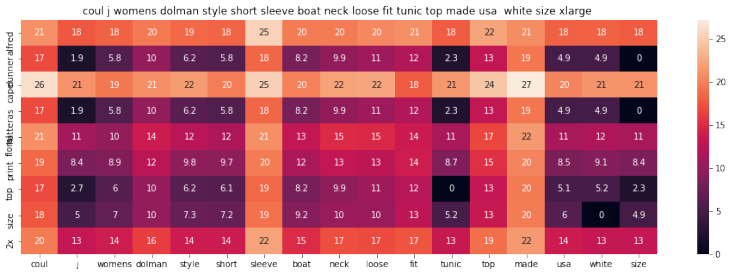
ASIN : B0759T6696
Brand : Tommy Hilfiger
euclidean distance from input : 8.698355255135988
==========



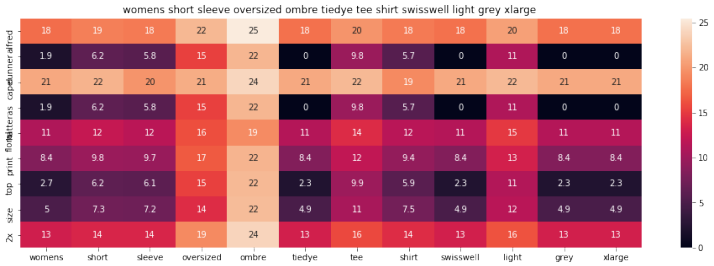
ASIN : B0722KD8RZ
Brand : Nanon
euclidean distance from input : 8.701326023234941
==========



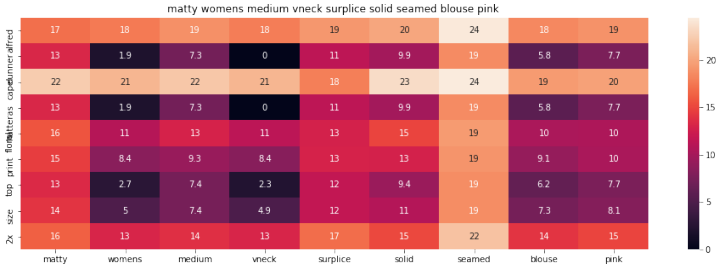
ASIN : B06W9HRMYX
Brand : Melrose And Market
euclidean distance from input : 8.704899274784399
==========



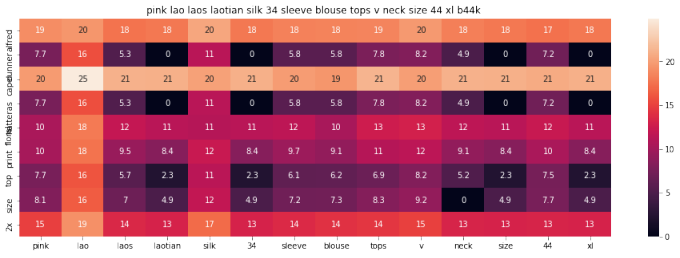
ASIN : B06XKY1HKZ
Brand : coul J
euclidean distance from input : 8.706984828433173
==========



ASIN : B071WBHSGP
Brand : SWISSWELL
euclidean distance from input : 8.712929187907793
==========



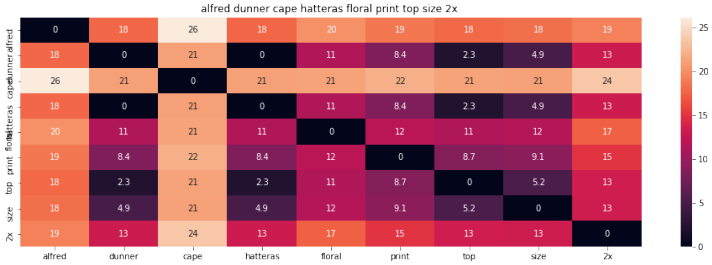
ASIN : B072LTP7Z2
Brand : Matty M
euclidean distance from input : 8.726094745120184
==========



ASIN : B072DXG86D
Brand : Nanon
euclidean distance from input : 8.741099093875066
==========

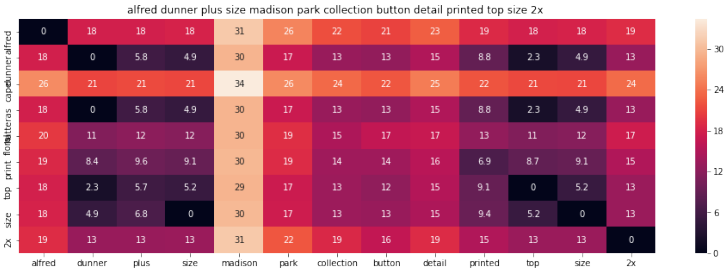
A sample for Brand Preference

In [26]: idf_w2v_brand_image(15698, 5,30,1,4,25)

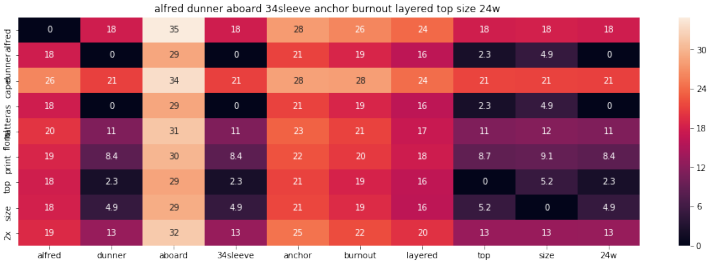


ASIN : B01JKGBK4Y
Brand : Alfred Dunner
euclidean distance from input : 0.00034526698291301725
==========

ASIN : B0758N8Q3R
Brand : Charter Club
euclidean distance from input : 2.1675604343414308
==========

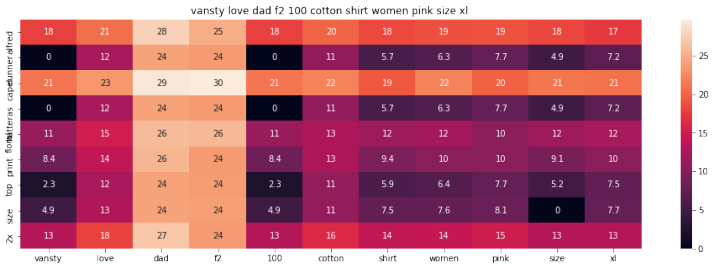


ASIN : B071WDLTHV
Brand : Alfred dunner
euclidean distance from input : 6.472673130035401
==========

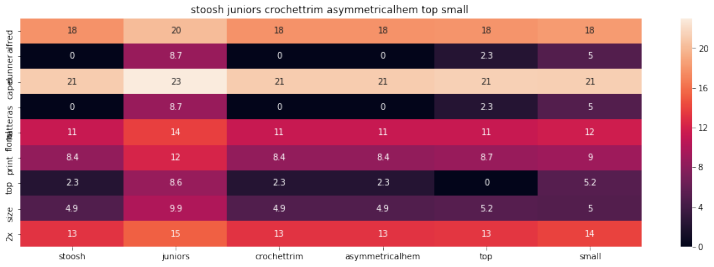


ASIN : B01N2H16MW
Brand : Alfred Dunner

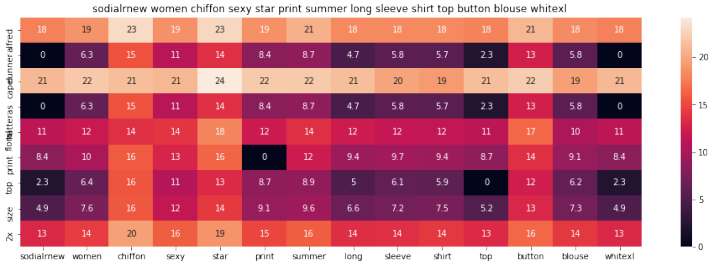
euclidean distance from input : 6.553946208953858



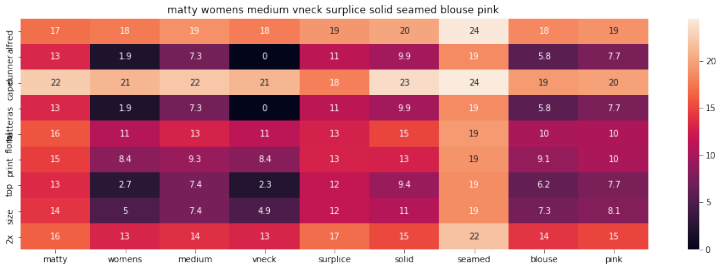
ASIN : B01EJS5AIK
Brand : Vansty
euclidean distance from input : 6.958462658603173



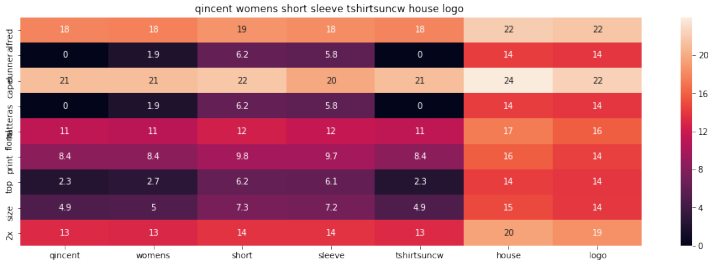
ASIN : B01F5S8R2E
Brand : Stoosh
euclidean distance from input : 7.019558671367589



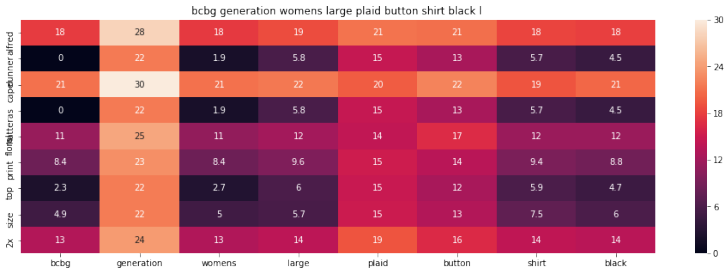
ASIN : B013OD2T26
Brand : SODIAL(R)
euclidean distance from input : 7.135910645205956
==========



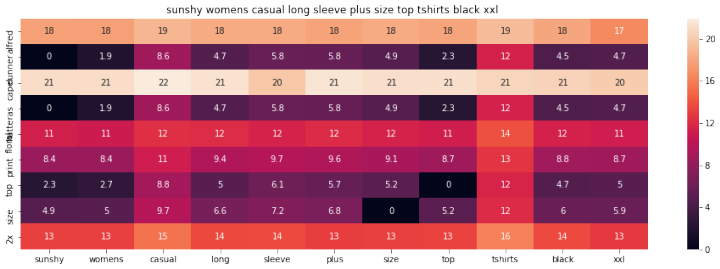
ASIN : B072LTP7Z2
Brand : Matty M
euclidean distance from input : 7.1785858542513665
==========



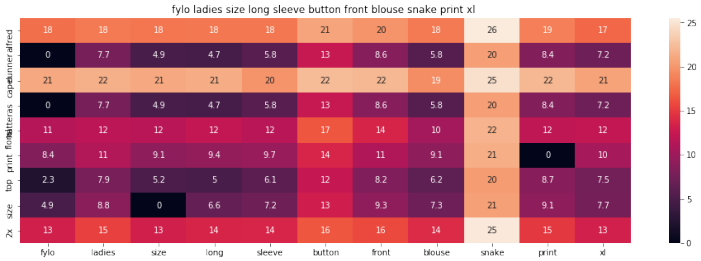
ASIN : B01AT4PCU4
Brand : Qincent
euclidean distance from input : 7.178859940249902
==========



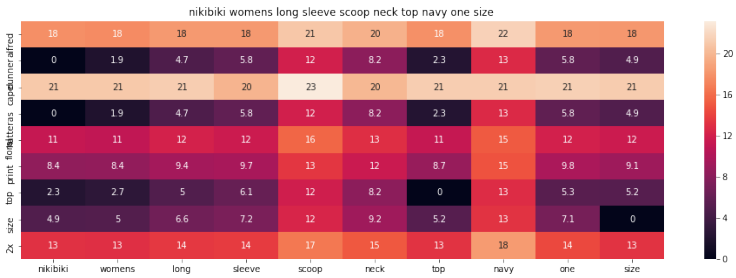
ASIN : B072XL7MTT
Brand : BCBGeneration
euclidean distance from input : 7.216386690814477
==========



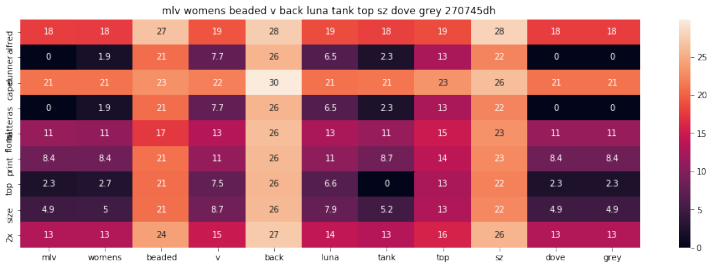
ASIN : B018EB3Q34
Brand : Juntong
euclidean distance from input : 7.221864166934472
==========



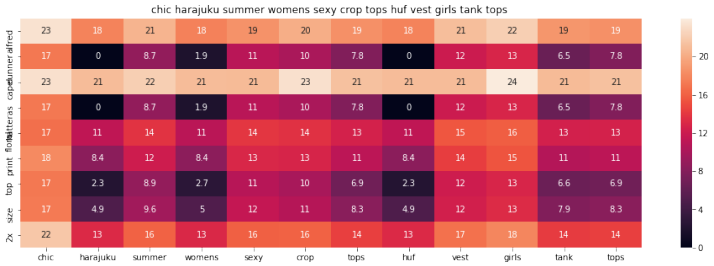
ASIN : B06Y2KFPQV
Brand : Fylo
euclidean distance from input : 7.244662383449499
==========



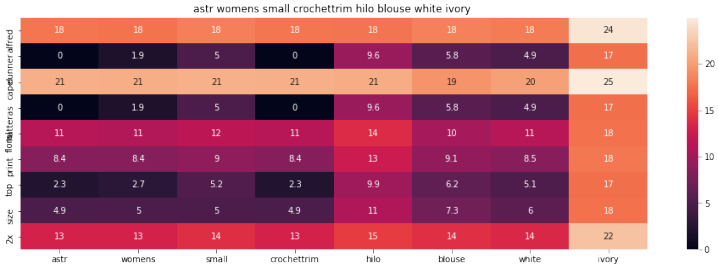
ASIN : B015IXOQYO
Brand : Nikibiki
euclidean distance from input : 7.261809435565453
==========



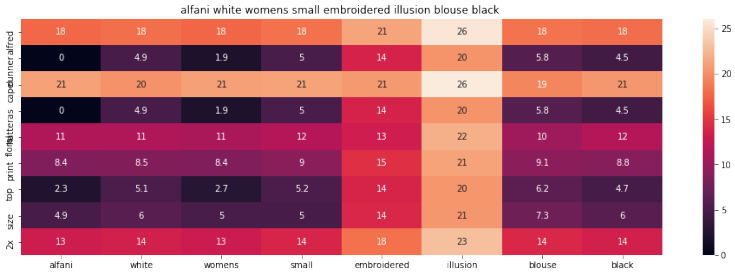
ASIN : B071XDBQWZ
Brand : MLV
euclidean distance from input : 7.265607932460729
==========



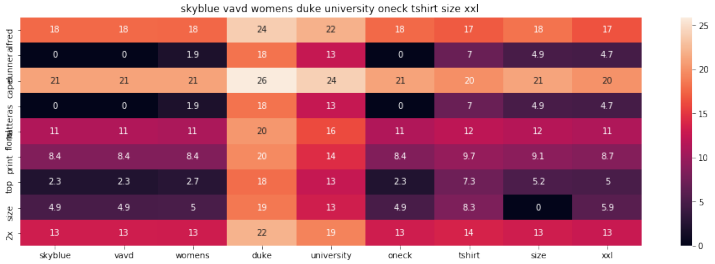
ASIN : B011OU5EUU
Brand : Chiclook Cool
euclidean distance from input : 7.274550924310182
==========



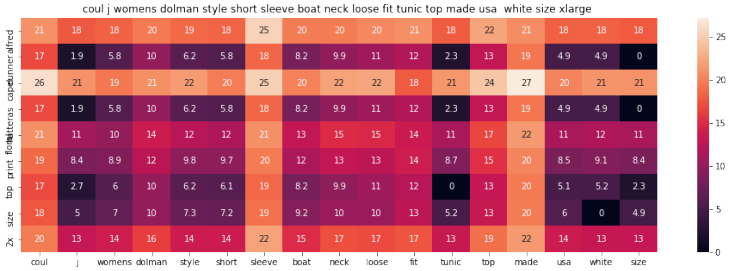
ASIN : B074X932QS
Brand : ASTR
euclidean distance from input : 7.2764935048942005
==========



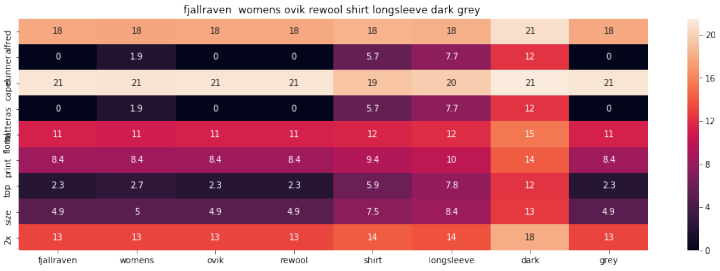
ASIN : B072FTMQ3S
Brand : Alfani
euclidean distance from input : 7.28078044958828
==========



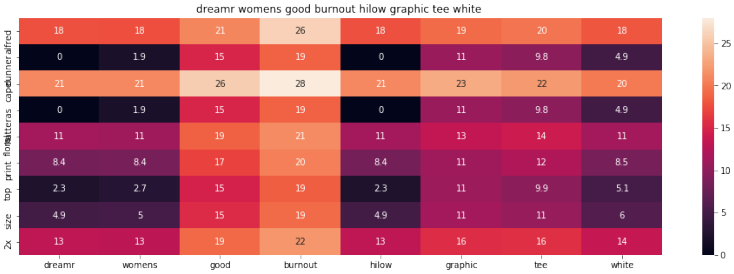
ASIN : B013PEJH6K
Brand : VAVD
euclidean distance from input : 7.282155123431664
==========



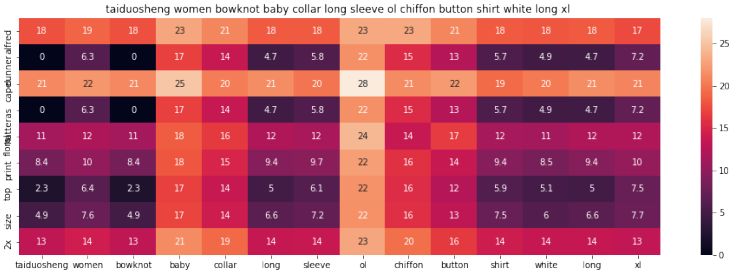
ASIN : B06XKY1HKZ
Brand : coul J
euclidean distance from input : 7.2869095713686765
==========



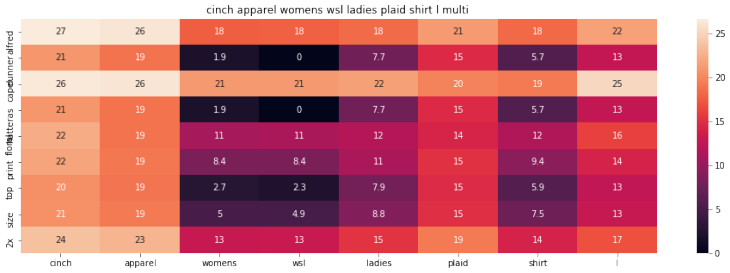
ASIN : B01N7RCYQ4
Brand : Fjallraven
euclidean distance from input : 7.288287309062902
==========



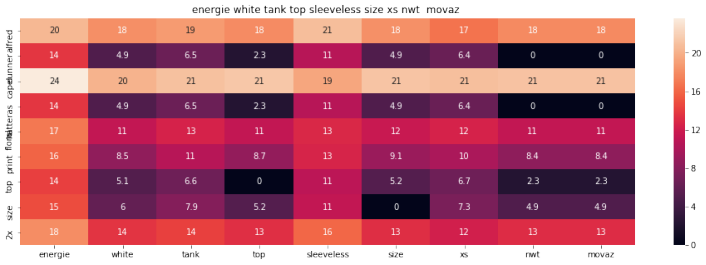
ASIN : B015X60ND4
Brand : Dreamr
euclidean distance from input : 7.3024656207156
==========



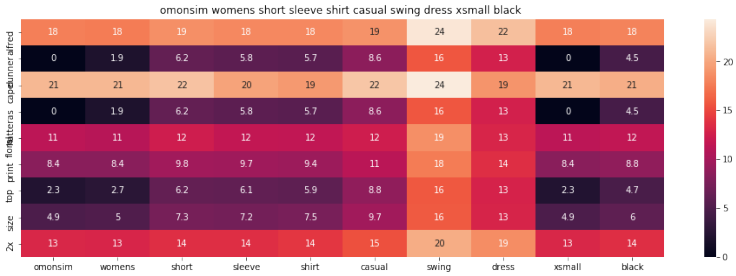
ASIN : B06XW87PMQ
Brand : Taiduosheng
euclidean distance from input : 7.303694716174585
==========



ASIN : B01M7NMEIS
Brand : Cinch
euclidean distance from input : 7.311870079706199
==========



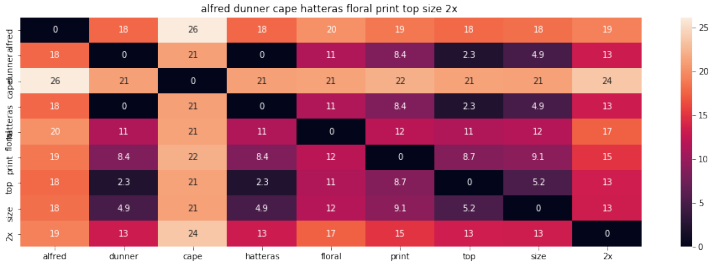
ASIN : B00Z8RY6EG
Brand : Energie
euclidean distance from input : 7.312457218844872
==========



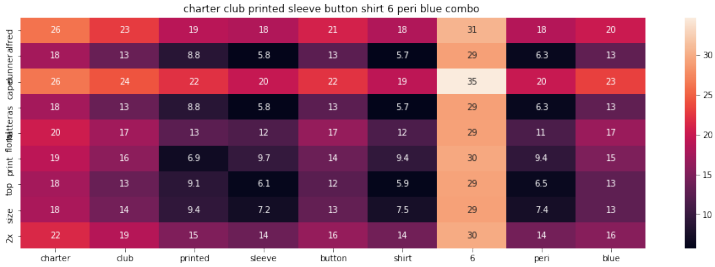
ASIN : B01GLF6Q3C
Brand : OMONSIM
euclidean distance from input : 7.318924084384423
==========

In [ ]: ##### A sample for Color Preference

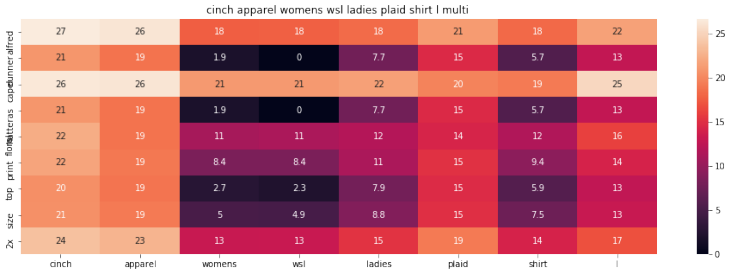
In [27]: idf_w2v_brand_image(15698, 5, 5,30,4,25)



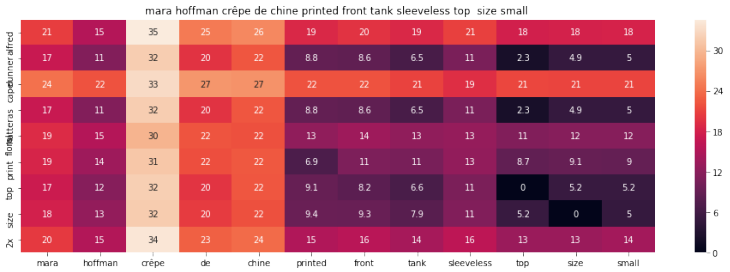
ASIN : B01JKGBK4Y
Brand : Alfred Dunner
euclidean distance from input : 0.0003138790753754702
==========



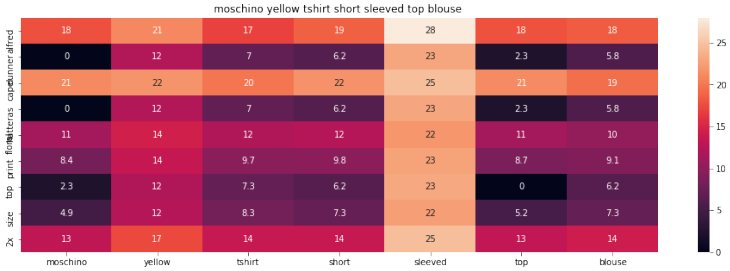
ASIN : B0758N8Q3R
Brand : Charter Club
euclidean distance from input : 2.1523276675831187
==========



ASIN : B01M7NMEIS
Brand : Cinch
euclidean distance from input : 5.663034840886954
==========

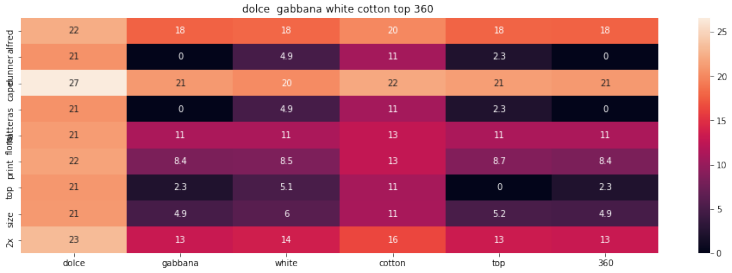


ASIN : B01ICBGPG6
Brand : Mara Hoffman
euclidean distance from input : 5.6957104856317695
==========

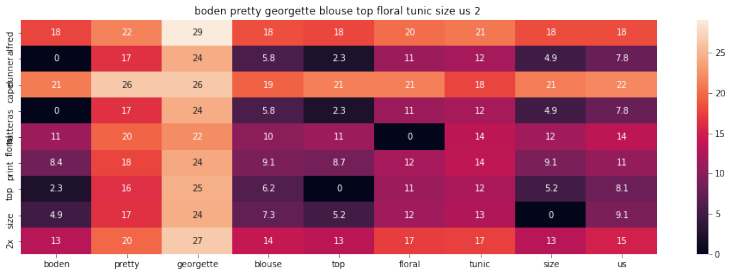


ASIN : B074G4VFHW
Brand : MOSCHINO

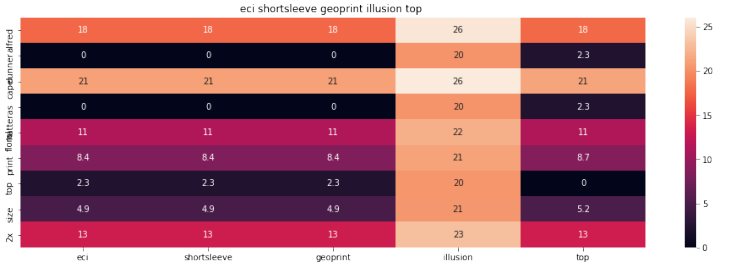
euclidean distance from input : 5.801692454121041



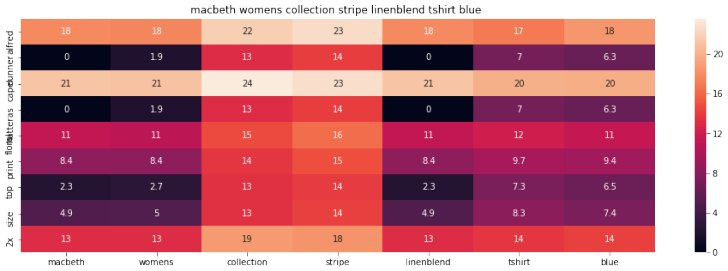
ASIN : B01MRZHU66
Brand : Dolce & Gabbana
euclidean distance from input : 5.805427637967196



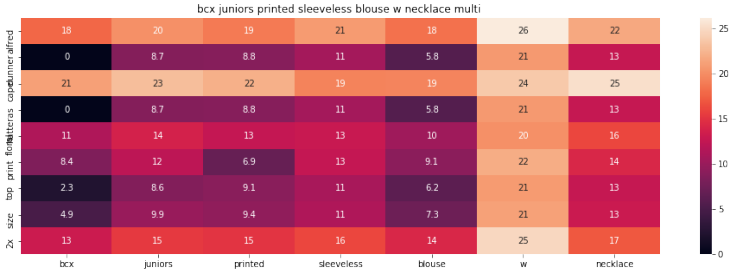
ASIN : B071R8YQQ6
Brand : BODEN
euclidean distance from input : 5.814084801630252



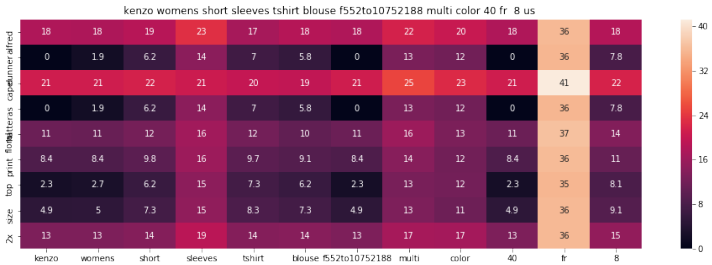
ASIN : B01BH7OMH6
Brand : ECI
euclidean distance from input : 5.815894615389795
==========



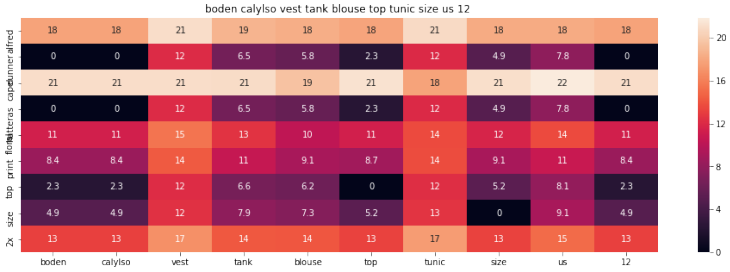
ASIN : B0722DZ925
Brand : Macbeth
euclidean distance from input : 5.820610664930922
==========



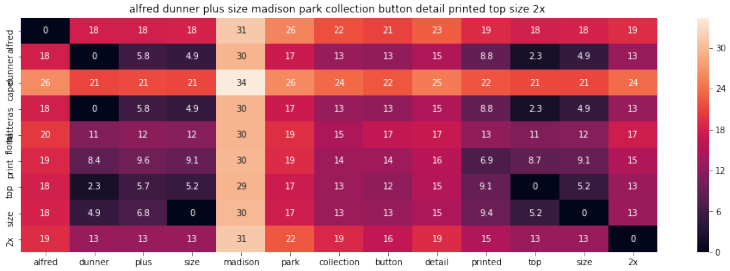
ASIN : B074ZR44C7
Brand : BCX
euclidean distance from input : 5.833726157664964
==========



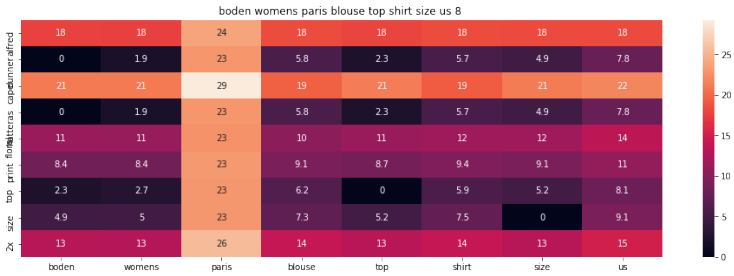
ASIN : B06W9JR55J
Brand : Kenzo
euclidean distance from input : 5.833994313543204
==========



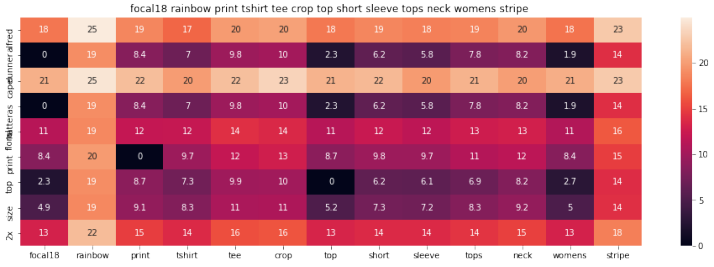
ASIN : B071DKSG78
Brand : BODEN
euclidean distance from input : 5.842969689325565
==========



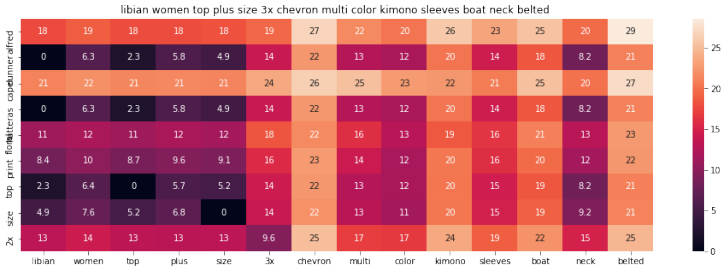
ASIN : B071WDLTHV
Brand : Alfred dunner
euclidean distance from input : 5.884248300032183
==========



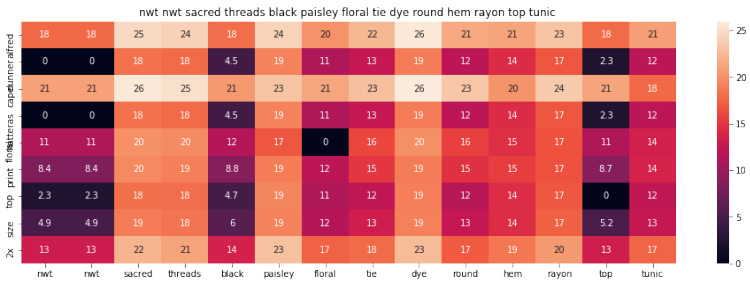
ASIN : B0727PRH1V
Brand : BODEN
euclidean distance from input : 5.899384943745065
==========



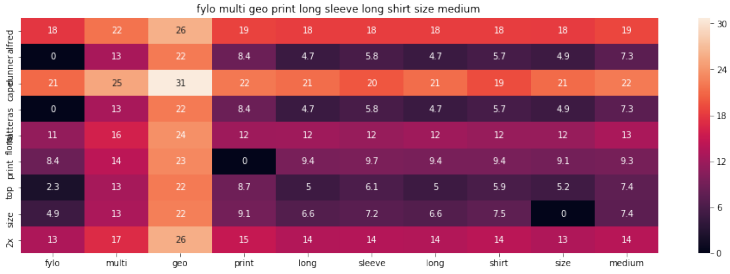
ASIN : B01N19SJ9F
Brand : Focal18
euclidean distance from input : 5.90293673034119
==========



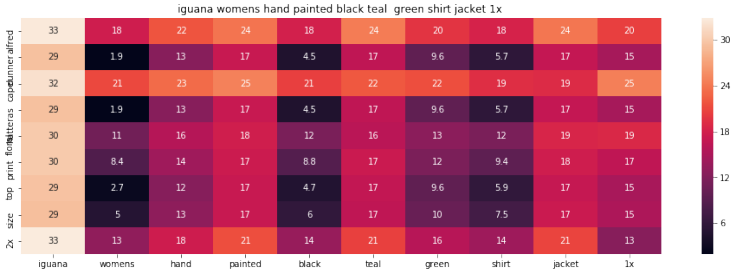
ASIN : B074S9Q7VM
Brand : Libian
euclidean distance from input : 5.913707658117179
==========



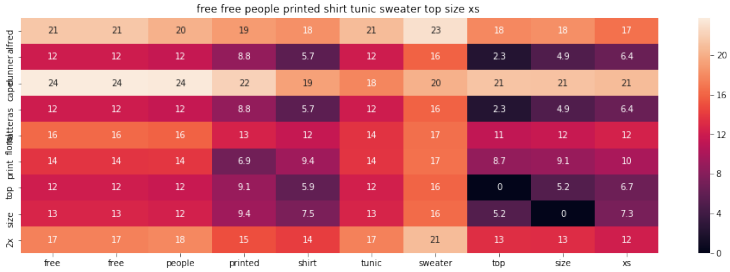
ASIN : B0756MMPML
Brand : None
euclidean distance from input : 5.917400100014427
==========



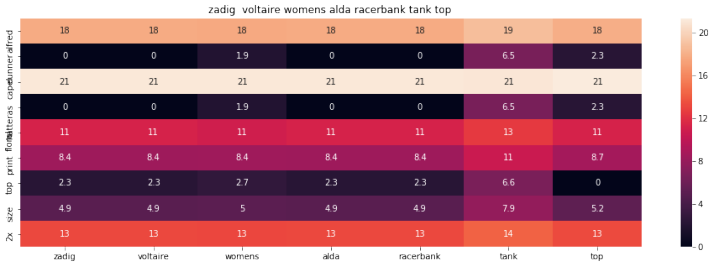
ASIN : B06VX8J4TM
Brand : Fylo
euclidean distance from input : 5.936724934534305
==========



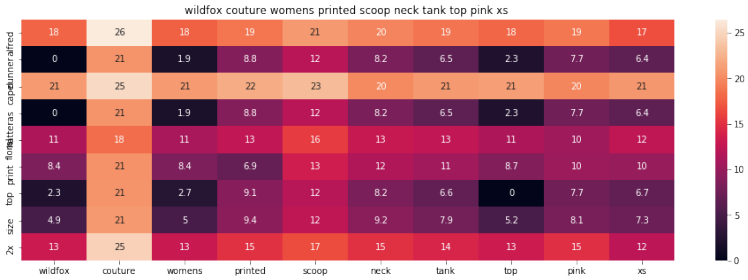
ASIN : B00W5WGAAA
Brand : Iguana
euclidean distance from input : 5.938627124656042
==========



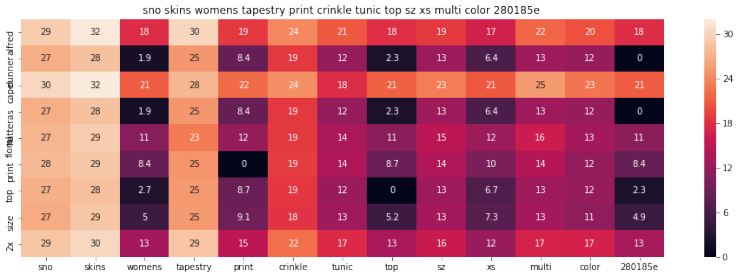
ASIN : B073SYZZWP
Brand : Free People
euclidean distance from input : 5.949575337496671
==========



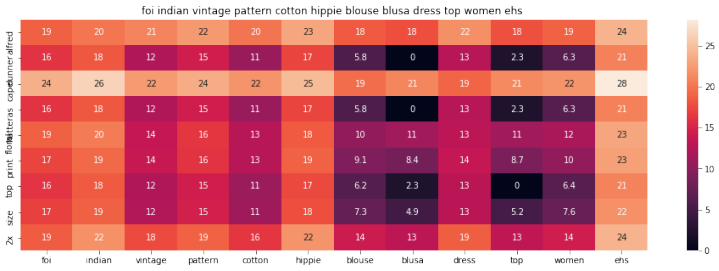
ASIN : B0753QLDKB
Brand : Zadig & Voltaire
euclidean distance from input : 5.971980224956166
==========



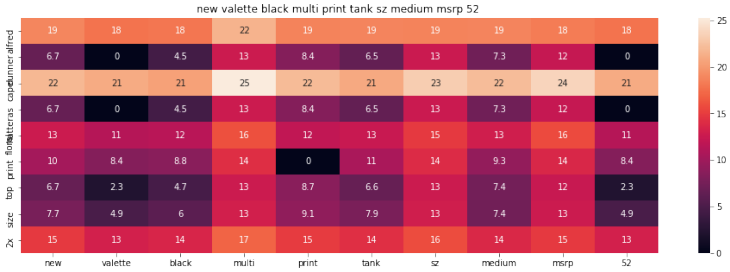
ASIN : B01DV6SQNC
Brand : Wildfox
euclidean distance from input : 5.972294645699299
==========



ASIN : B071W651CD
Brand : SNO SKINS
euclidean distance from input : 5.974771152843129
==========



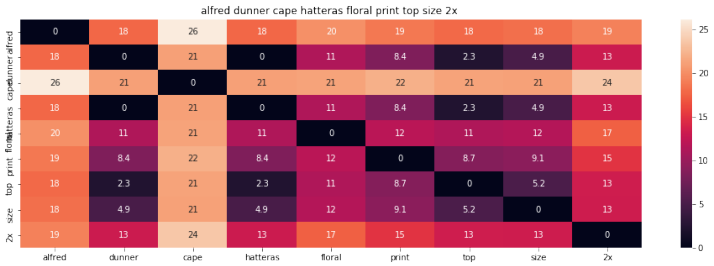
ASIN : B014KOOI6C
Brand : Fashion of India
euclidean distance from input : 5.980206057762538
==========



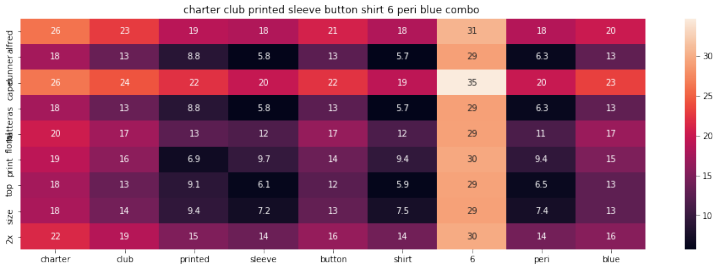
ASIN : B0757YRT8H
Brand : Valette
euclidean distance from input : 5.980428013757938
==========

A sample for Visual Preferences

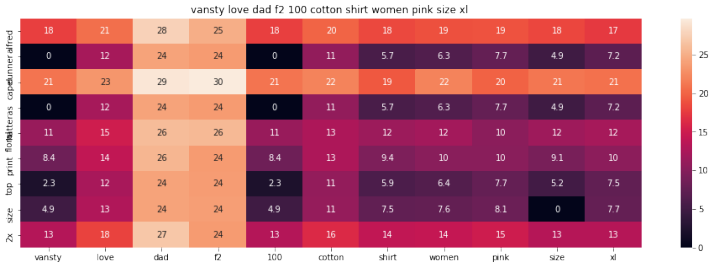
In [28]: idf_w2v_brand_image(15698, 5, 5,1,30,25)



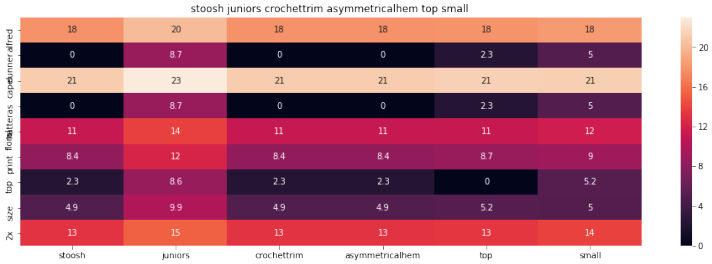
ASIN : B01JKGBK4Y
Brand : Alfred Dunner
euclidean distance from input : 0.00033684583698830956
==========



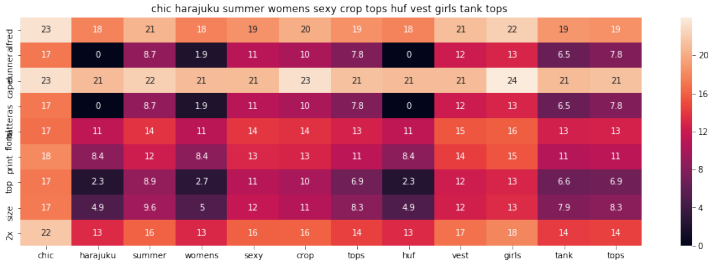
ASIN : B0758N8Q3R
Brand : Charter Club
euclidean distance from input : 0.8951809115526153
==========



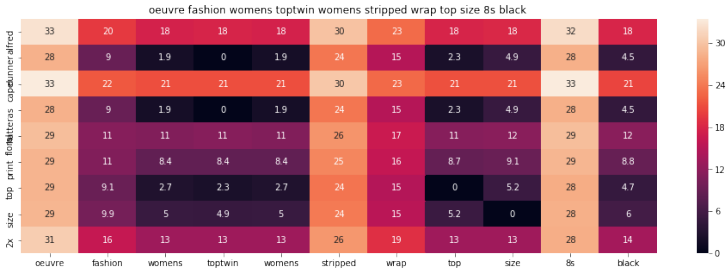
ASIN : B01EJS5AIK
Brand : Vansty
euclidean distance from input : 37.818874939205955
==========



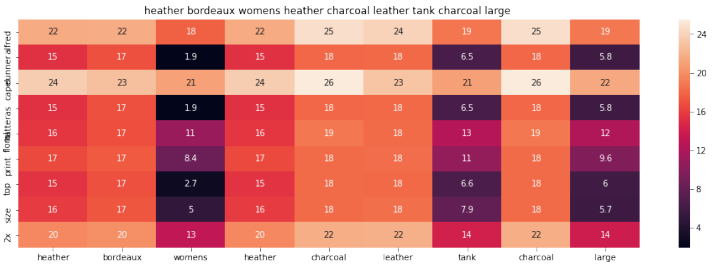
ASIN : B01F5S8R2E
Brand : Stoosh
euclidean distance from input : 37.96036021175813
==========



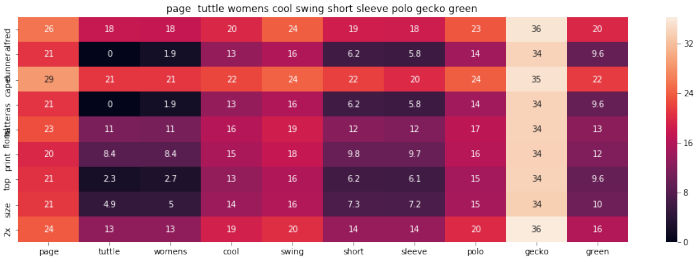
ASIN : B011OU5EUU
Brand : Chiclook Cool
euclidean distance from input : 38.53212450540075
==========



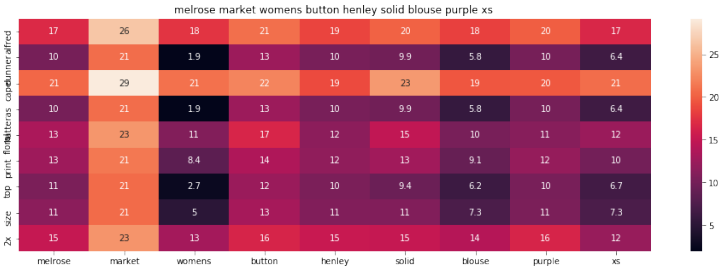
ASIN : B06XJ21PKK
Brand : OEUVRE FASHION
euclidean distance from input : 38.93012723475194
==========



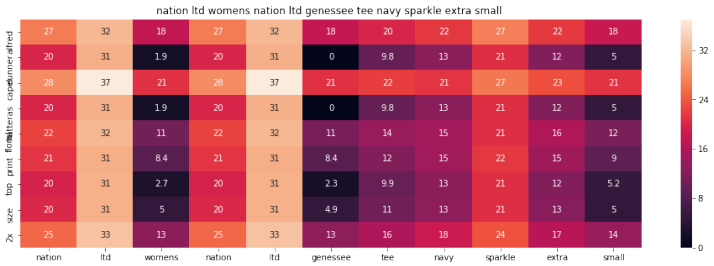
ASIN : B071VFDGG3
Brand : Heather by Bordeaux
euclidean distance from input : 38.9374839614834
==========



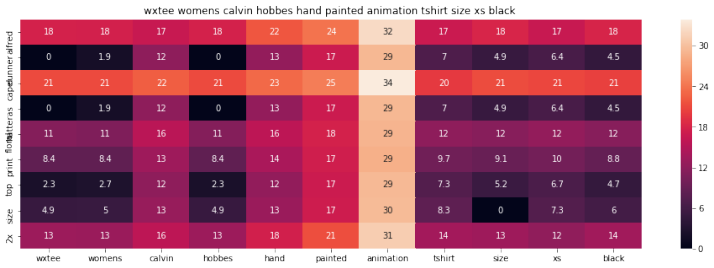
ASIN : B005BX28LA
Brand : PAGE & TUTTLE
euclidean distance from input : 39.03392356564585
==========



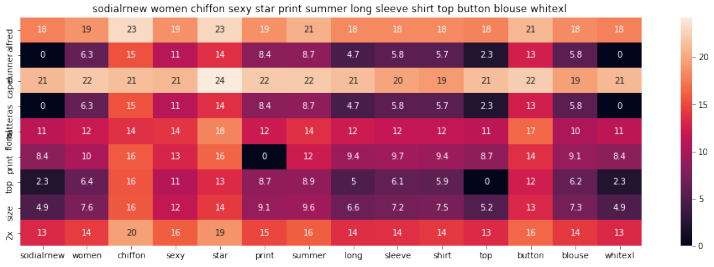
ASIN : B06W9HRMYX
Brand : Melrose And Market
euclidean distance from input : 39.188361900102876
==========



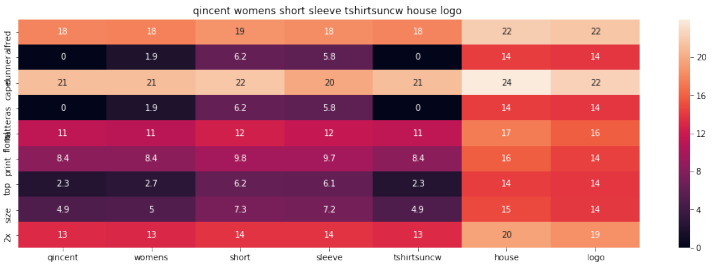
ASIN : B01N6V7HY2
Brand : Nation LTD
euclidean distance from input : 39.19494310024535
==========



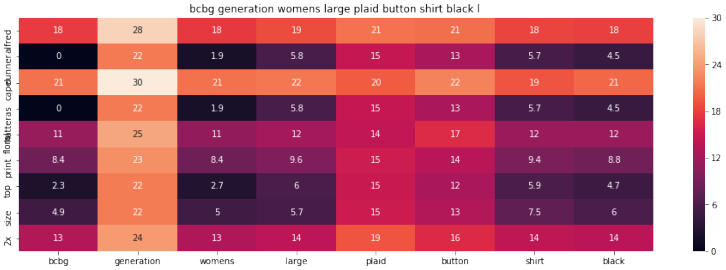
ASIN : B012GAD8H8
Brand : WTEE Geek Design
euclidean distance from input : 39.265499870445744
==========



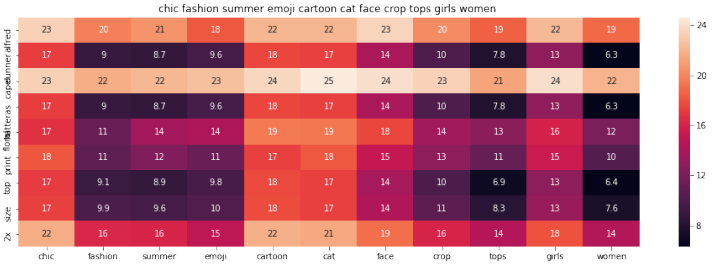
ASIN : B013OD2T26
Brand : SODIAL(R)
euclidean distance from input : 39.400604083790796
==========



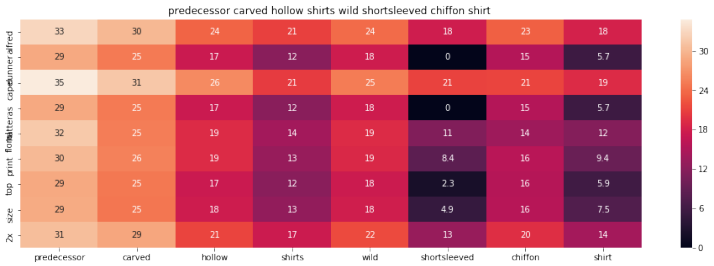
ASIN : B01AT4PCU4
Brand : Qincent
euclidean distance from input : 39.435946788749824
==========



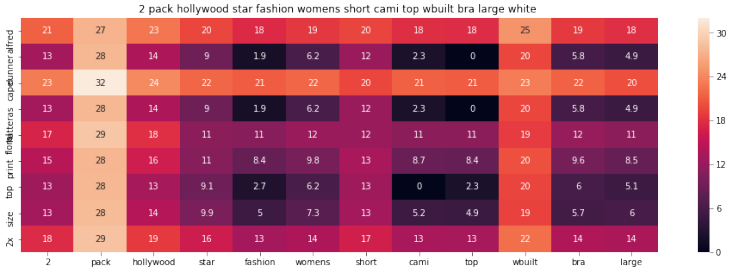
ASIN : B072XL7MTT
Brand : BCBGeneration
euclidean distance from input : 39.501458027034055
==========



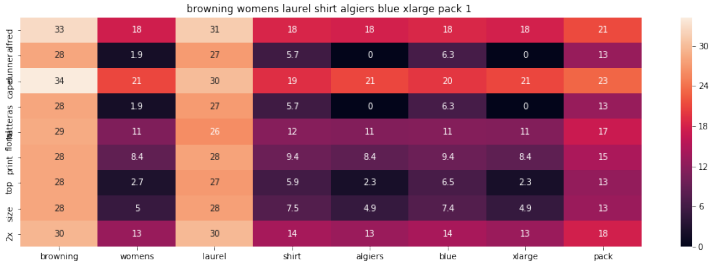
ASIN : B011UEYDHK
Brand : Chiclook Cool
euclidean distance from input : 39.50627583760468
==========



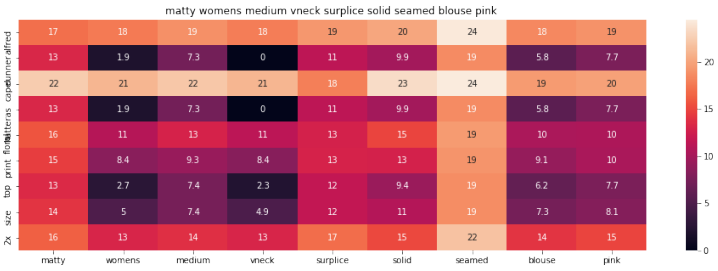
ASIN : B0754LDZBT
Brand : Rbwinner
euclidean distance from input : 39.507343977150896
==========



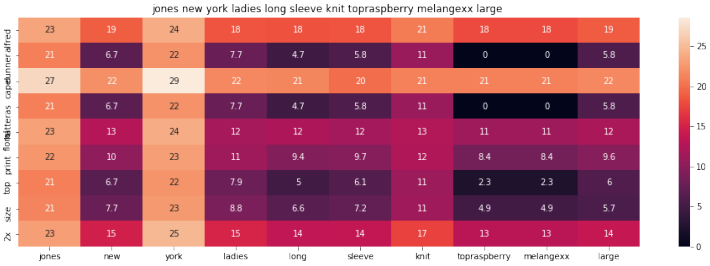
ASIN : B00M73W574
Brand : Hollywood Star Fashion
euclidean distance from input : 39.59001002610028
==========



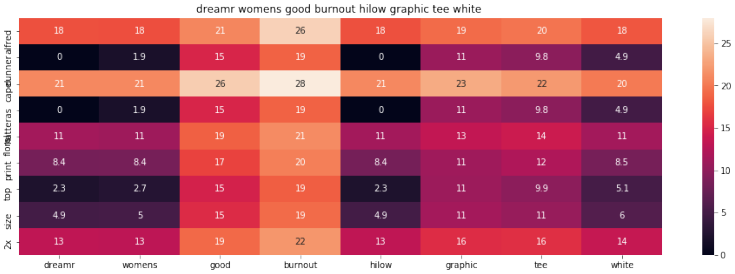
ASIN : B01KAYFT7O
Brand : SPG Outdoors
euclidean distance from input : 39.65087725261589
==========



ASIN : B072LTP7Z2
Brand : Matty M
euclidean distance from input : 39.67244183001555
==========



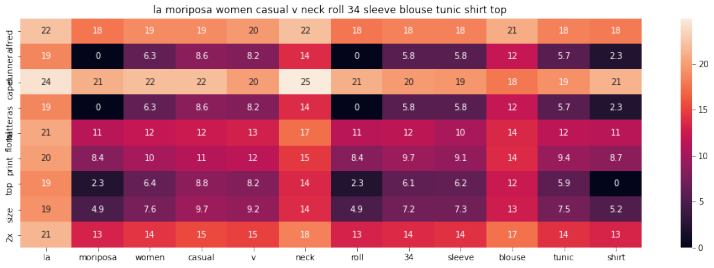
ASIN : B06XGB5Y5G
Brand : Jones New York
euclidean distance from input : 39.786161480374524
==========



ASIN : B015X60ND4
Brand : Dreamr

euclidean distance from input : 39.79659268770836

==========

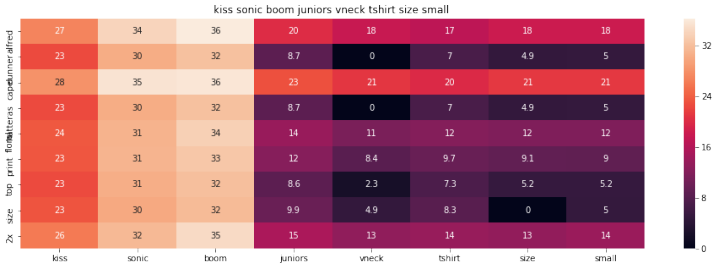


ASIN : B01K1O3X0I

Brand : La moriposa

euclidean distance from input : 39.80264287809928

==========

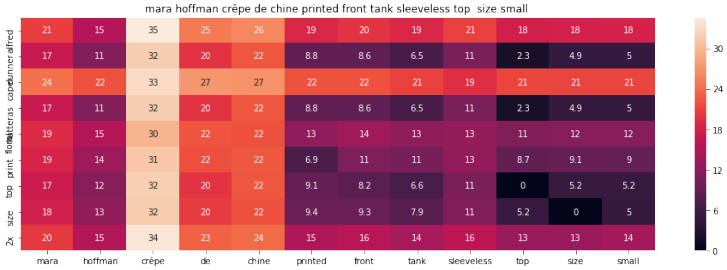


ASIN : B01FLH1UM8

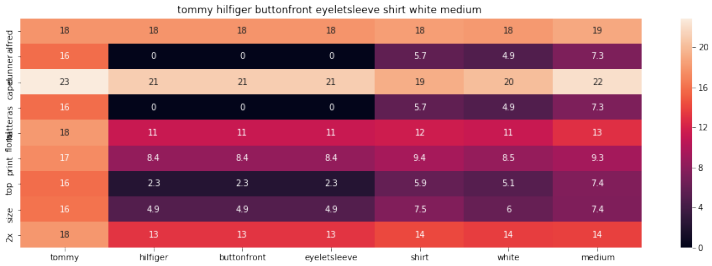
Brand : Kiss

euclidean distance from input : 39.85298326395467

==========



```
ASIN : B01ICBGPG6
Brand : Mara Hoffman
euclidean distance from input : 39.92889874155929
```



```
ASIN : B06XXL5H9Z
Brand : Tommy Hilfiger
euclidean distance from input : 39.96275972506776
```

7 Note-This overall project is done under appliedaicourse.com