Amazon Apparel Recommendations

[4.2] Data and Code:

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

[4.3] Overview of the data

In [1]:
```
#import all the necessary packages.

from PIL import Image
import requests
from io import BytesIO
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import warnings
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
import math
import time
import re
import os
import seaborn as sns
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import pairwise_distances
from matplotlib import gridspec
from scipy.sparse import hstack
import plotly
import plotly.figure_factory as ff
from plotly.graph_objs import Scatter, Layout

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

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

In [6]:
```
print ('Number of data points : ', data.shape[0], \
       'Number of features/variables:', data.shape[1])
```

```
Number of data points :  183138 Number of features/variables: 19
```

Terminology:

What is a dataset?
Rows and columns
Data-point
Feature/variable

In [7]:
```
# each product/item has 19 features in the raw dataset.
data.columns # prints column-names or feature-names.
```

Out[7]:
```
Index(['asin', 'author', 'availability', 'availability_type', 'brand', 'colo
r',
       'editorial_reivew', 'editorial_review', 'formatted_price',
       'large_image_url', 'manufacturer', 'medium_image_url', 'model',
       'product_type_name', 'publisher', 'reviews', 'sku', 'small_image_url',
       'title'],
      dtype='object')
```

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

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

In [8]:
```
data = data[['asin', 'brand', 'color', 'medium_image_url', 'product_type_name'
, 'title', 'formatted_price']]
```

In [9]:
```
print ('Number of data points : ', data.shape[0], \
       'Number of features:', data.shape[1])
data.head() # prints the top rows in the table.
```

```
Number of data points :  183138 Number of features: 7
```

Out[9]:

	asin	brand	color	medium_image_url	product_type_name	t
0	B016I2TS4W	FNC7C	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Minions Como Superher Ironman Long Slee R...
1	B01N49AI08	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG Cloth Womens Tunic
2	B01JDPCOHO	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG Cloth Womens Won Top
3	B01N19U5H5	Focal18	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Focal18 Sailor Co Bubble Sleeve Blouse Sh
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	Featherlit Ladies' Lo Sleeve St Resistan.

[5.1] Missing data for various features.

Basic stats for the feature: product_type_name

In [10]:
```
# We have total 72 unique type of product_type_names
print(data['product_type_name'].describe())

# 91.62% (167794/183138) of the products are shirts,
```

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

In [11]:
```
# names of different product types
print(data['product_type_name'].unique())
```

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

In [12]:
```
# find the 10 most frequent product_type_names.
product_type_count = Counter(list(data['product_type_name']))
product_type_count.most_common(10)
```

Out[12]:
```
[('SHIRT', 167794),
 ('APPAREL', 3549),
 ('BOOKS_1973_AND_LATER', 3336),
 ('DRESS', 1584),
 ('SPORTING_GOODS', 1281),
 ('SWEATER', 837),
 ('OUTERWEAR', 796),
 ('OUTDOOR_RECREATION_PRODUCT', 729),
 ('ACCESSORY', 636),
 ('UNDERWEAR', 425)]
```

Basic stats for the feature: brand

In [13]:
```
# there are 10577 unique brands
print(data['brand'].describe())

# 183138 - 182987 = 151 missing values.
```

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

In [14]:
```
brand_count = Counter(list(data['brand']))
brand_count.most_common(10)
```

Out[14]:
```
[('Zago', 223),
 ('XQS', 222),
 ('Yayun', 215),
 ('YUNY', 198),
 ('XiaoTianXin-women clothes', 193),
 ('Generic', 192),
 ('Boohoo', 190),
 ('Alion', 188),
 ('Abetteric', 187),
 ('TheMogan', 187)]
```

Basic stats for the feature: color

In [15]:
```
print(data['color'].describe())


# we have 7380 unique colors
# 7.2% of products are black in color
# 64956 of 183138 products have brand information. That's approx 35.4%.
```

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

In [16]:
```
color_count = Counter(list(data['color']))
color_count.most_common(10)
```

Out[16]:
```
[(None, 118182),
 ('Black', 13207),
 ('White', 8616),
 ('Blue', 3570),
 ('Red', 2289),
 ('Pink', 1842),
 ('Grey', 1499),
 ('*', 1388),
 ('Green', 1258),
 ('Multi', 1203)]
```

Basic stats for the feature: formatted_price

In [17]:
```
print(data['formatted_price'].describe())

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

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

In [18]:
```
price_count = Counter(list(data['formatted_price']))
price_count.most_common(10)
```

Out[18]:
```
[(None, 154743),
 ('$19.99', 945),
 ('$9.99', 749),
 ('$9.50', 601),
 ('$14.99', 472),
 ('$7.50', 463),
 ('$24.99', 414),
 ('$29.99', 370),
 ('$8.99', 343),
 ('$9.01', 336)]
```

Basic stats for the feature: title

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

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

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

In [20]:
```
data.to_pickle('pickels/180k_apparel_data')
```

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

In [21]:
```
# consider products which have price information
# data['formatted_price'].isnull() => gives the information
#about the dataframe row's which have null values price == None|Null
data = data.loc[~data['formatted_price'].isnull()]
print('Number of data points After eliminating price=NULL :', data.shape[0])
```

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

In [22]:
```
# consider products which have color information
# data['color'].isnull() => gives the information about the dataframe row's wh
ich have null values price == None|Null
data =data.loc[~data['color'].isnull()]
print('Number of data points After eliminating color=NULL :', data.shape[0])
```

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

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

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

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

In [23]:
```
data.to_pickle('pickels/28k_apparel_data')
```

In [52]:
```
# You can download all these 28k images using this code below.
# You do NOT need to run this code and hence it is commented.


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

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


'''
```

Out[52]:
```
"\nfrom PIL import Image\nimport requests\nfrom io import BytesIO\n\nfor inde
x, row in images.iterrows():\n        url = row['large_image_url']\n        r
esponse = requests.get(url)\n        img = Image.open(BytesIO(response.conten
t))\n        img.save('workshop/images/28k_images/'+row['asin']+'.jpeg')\n\n
\n"
```

[5.2] Remove near duplicate items

[5.2.1] Understand about duplicates.

In [97]:
```
# read data from pickle file from previous stage
data = pd.read_pickle('pickels/28k_apparel_data')

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

```
2325
```

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

 :B00AQ4GMCK	 :B00AQ4GMTS
 :B00AQ4GMLQ	 :B00AQ4GN3I

These shirts exactly same except in color

 :B00G278GZ6	 :B00G278W6O
 :B00G278Z2A	 :B00G2786X8

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

[5.2.2] Remove duplicates : Part 1

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

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

Out[103]:

	asin	brand	color	medium_image_url	product_type_name	
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	Featherli Ladies' L Sleeve S Resistan
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	Women's Unique 100% Co T - Spec Olympic.
11	B001LOUGE4	Fitness Etc.	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	Ladies Cotton T 2x1 Ribb Tank Top
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	FeatherL Ladies' Moisture Free Me Sport S..
21	B014ICEDNA	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	Superna Chibis S Dean An Castiel Short...

In [104]:
```
# Remove All products with very few words in title
data_sorted = data[data['title'].apply(lambda x: len(x.split())>4)]
print("After removal of products with short description:", data_sorted.shape[0
])
```

```
After removal of products with short description: 27949
```

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

Out[105]:

	asin	brand	color	medium_image_url	product_type_name	
61973	B06Y1KZ2WB	Éclair	Black/Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	É W P T B B
133820	B010RV33VE	xiaoming	Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	xi W S L L sl
81461	B01DDSDLNS	xiaoming	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	xi W W L S S B
75995	B00X5LYO9Y	xiaoming	Red Anchors	https://images-na.ssl-images-amazon.com/images...	SHIRT	xi S Ta P S A
151570	B00WPJG35K	xiaoming	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	xi S S L Ta K W

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

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

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

Title 3:
61. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print
Head Shirt for woman Neon Wolf t-shirt
62. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print
Head Shirt for woman Neon Wolf t-shirt
63. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print
Head Shirt for woman Neon Wolf t-shirt
64. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print
Head Shirt for woman Neon Wolf t-shirt
```

In [106]:
```
indices = []
for i,row in data_sorted.iterrows():
    indices.append(i)
```

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

    previous_i = i

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

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

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

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

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

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

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

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

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

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

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

We removed the dupliactes which differ only at the end.

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

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

In [110]:
```
data.to_pickle('pickels/17k_apperal_data')
```

[5.2.3] Remove duplicates : Part 2

In the previous cell, we sorted whole data in alphabetical order of titles.Then, w
e 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-L
arge
115042. UltraClub Ladies Classic Wrinkle-Free Long-Sleeve Oxford Light Blue XXL

TItles-2
75004. EVALY Women's Cool University Of UTAH 3/4 Sleeve Raglan Tee
109225. EVALY Women's Unique University Of UTAH 3/4 Sleeve Raglan Tees
120832. EVALY Women's New University Of UTAH 3/4-Sleeve Raglan Tshirt

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

In [66]:
```
# This code snippet takes significant amount of time.
# O(n^2) time.
# Takes about an hour to run on a decent computer.

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

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

        b = data['title'].loc[j].split()
        # store the list of words of jth string in b, ex: b = ['tokidoki', 'Th
e', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'X-Large']

        length = max(len(a),len(b))

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

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

        # if the number of words in which both strings differ are < 3 , we are
considering it as those two apperals are same, hence we are ignoring them
        if (length - count) < 3:
            indices.remove(j)
```

In [71]:
```
# from whole previous products we will consider only
# the products that are found in previous cell
data = data.loc[data['asin'].isin(stage2_dedupe_asins)]
```

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

```
Number of data points after stage two of dedupe:  16042
```

In [75]:
```
data.to_pickle('pickels/16k_apperal_data')
# Storing these products in a pickle file
# candidates who wants to download these files instead
# of 180K they can download and use them from the Google Drive folder.
```

6. Text pre-processing

In [3]:
```
data = pd.read_pickle('pickels/16k_apperal_data')

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

In [4]:
```
# we use the list of stop words that are downloaded from nltk lib.
stop_words = set(stopwords.words('english'))
print ('list of stop words:', stop_words)

def nlp_preprocessing(total_text, index, column):
    if type(total_text) is not int:
        string = ""
        for words in total_text.split():
            # remove the special chars in review like '"#$@!%^&*()_+-~?>< etc.
            word = ("".join(e for e in words if e.isalnum()))
            # Conver all letters to lower-case
            word = word.lower()
            # stop-word removal
            if not word in stop_words:
                string += word + " "
        data[column][index] = string
```

```
list of stop words: {'such', 'and', 'hers', 'up', 'she', 'd', 'further', 'al
l', 'than', 'under', 'is', 'off', 'both', 'most', 'few', 'should', 're', 'ver
y', 'just', 'then', 'didn', 'myself', 'in', 'too', 's', 'shouldn', 'herself',
'because', 'how', 'itself', 'what', 'shan', 'weren', 'doing', 'them', 'could
n', 'their', 'so', 'ain', 'haven', 'yourself', 'now', 'll', 'isn', 'about',
'over', 'into', 'before', 'during', 'on', 'as', 'aren', 'against', 'above',
'down', 'they', 'below', 'me', 'again', 'for', 'why', 'been', 'yourselves',
'more', 'her', 'that', 'can', 'am', 'was', 'themselves', 'mightn', 'does', 't
hose', 'only', 'hasn', 'any', 'ma', 'are', 'nor', 'out', 'you', 'ourselves',
'the', 'an', 'has', 'where', 'i', 'while', 'ours', 'its', 'your', 'had', 'wer
e', 'being', 'no', 'or', 'needn', 've', 'y', 'a', 'each', 'have', 'through',
'when', 'mustn', 'by', 'won', 'from', 'own', 'will', 'there', 't', 'him', 'th
ese', 'doesn', 'theirs', 'my', 'did', 'of', 'who', 'until', 'wouldn', 'we',
'do', 'having', 'yours', 'other', 'wasn', 'it', 'with', 'once', 'here', 'do
n', 'o', 'whom', 'this', 'if', 'but', 'hadn', 'our', 'some', 'm', 'not', 'bet
ween', 'himself', 'same', 'at', 'be', 'he', 'after', 'which', 'to', 'his'}
```

In [5]:
```
start_time = time.clock()
# we take each title and we text-preprocess it.
for index, row in data.iterrows():
    nlp_preprocessing(row['title'], index, 'title')
# we print the time it took to preprocess whole titles
print(time.clock() - start_time, "seconds")
```

```
3.5727220000000006 seconds
```

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

Out[6]:

	asin	brand	color	medium_image_url	product_type_name	
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	featherli ladies lo sleeve s resistant
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	womens unique 1 cotton special olympics wor...
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	featherli ladies moisture free mes sport sh
27	B014ICEJ1Q	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	superna chibis sa dean ca neck tsh
46	B01NACPBG2	Fifth Degree	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	fifth deg womens gold foil graphic jun...

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

Stemming

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


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

```
argu
fish
```

[8] Text based product similarity

In [8]:
```
data = pd.read_pickle('pickels/16k_apperal_data_preprocessed')
data.head()
```

Out[8]:

	asin	brand	color	medium_image_url	product_type_name	
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	featherli ladies lo sleeve s resistant
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	womens unique 1 cotton special olympics wor...
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	featherli ladies moisture free mes sport sh
27	B014ICEJ1Q	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	superna chibis sa dean ca neck tsh
46	B01NACPBG2	Fifth Degree	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	fifth deg womens gold foil graphic jun...

In [9]:
```
# Utility Functions which we will use through the rest of the workshop.


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

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

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

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

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

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

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

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

    # doc_id : index of the title1
    # vec1 : input apparels's vector, it is of a dict type {word:count}
```

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

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

    # we set the values of non intersecting words to zero, this is just to sho
w the difference in heatmap
    for i in vec2:
        if i not in intersection:
            vec2[i]=0

    # for labeling heatmap, keys contains list of all words in title2
    keys = list(vec2.keys())
    #  if ith word in intersection(lis of words of title1 and list of words of
title2): values(i)=count of that word in title2 else values(i)=0
    values = [vec2[x] for x in vec2.keys()]

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

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

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


# this function gets a list of wrods along with the frequency of each
# word given "text"
def text_to_vector(text):
    word = re.compile(r'\w+')
    words = word.findall(text)
    # words stores list of all words in given string, you can try 'words = tex
t.split()' this will also gives same result
    return Counter(words) # Counter counts the occurence of each word in list,
it returns dict type object {word1:count}



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

    # vector1 = dict{word11:#count, word12:#count, etc.}
    vector1 = text_to_vector(text1)
    #print(vector1)
    # vector1 = dict{word21:#count, word22:#count, etc.}
    vector2 = text_to_vector(text2)
    #print(vector2)

    plot_heatmap_image(doc_id, vector1, vector2, url, text2, model)
```

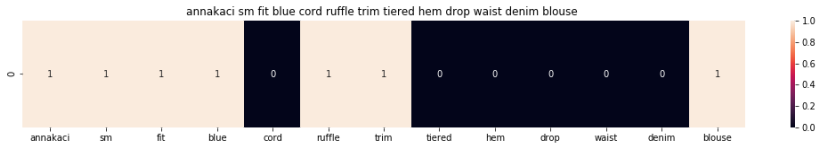
[8.2] Bag of Words (BoW) on product titles.

In [10]:
```
from sklearn.feature_extraction.text import CountVectorizer
title_vectorizer = CountVectorizer()
title_features   = title_vectorizer.fit_transform(data['title'])
title_features.get_shape() # get number of rows and columns in feature matrix.
# title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(corpus) returns
# the a sparase matrix of dimensions #data_points * #words_in_corpus

# What is a sparse vector?

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

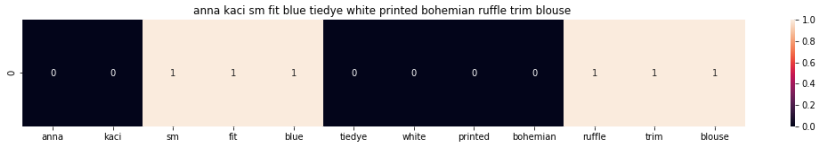
Out[10]: (16042, 12609)

In [11]:
```
def bag_of_words_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

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

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

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

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

#call the bag-of-words model for a product to get similar products.
bag_of_words_model(931, 25) # change the index if you want to.
# In the output heat map each value represents the count value
# of the label word, the color represents the intersection
# with inputs title.

#try 12566
#try 931
```

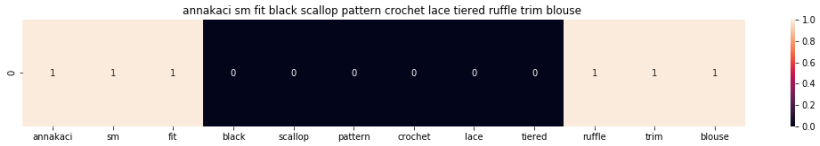
ASIN : B00KLHUIBS
Brand: Anna-Kaci
Title: annakaci sm fit blue green polka dot tie front ruffle trim blouse
Euclidean similarity with the query image : 0.0
==========



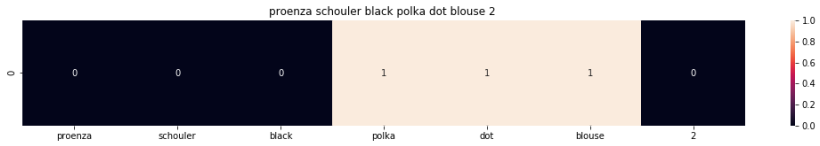
ASIN : B0759G15ZX
Brand: Anna-Kaci
Title: annakaci sm fit blue cord ruffle trim tiered hem drop waist denim blou
se
Euclidean similarity with the query image : 3.3166247903554
==========



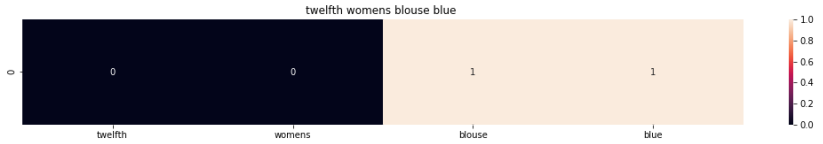
ASIN : B00YQ8S4K0
Brand: Anna-Kaci
Title: anna kaci sm fit blue tiedye white printed bohemian ruffle trim blouse
Euclidean similarity with the query image : 3.4641016151377544
==========



ASIN : B00O194W8W
Brand: Anna-Kaci
Title: annakaci sm fit black scallop pattern crochet lace tiered ruffle trim
blouse
Euclidean similarity with the query image : 3.4641016151377544
==========

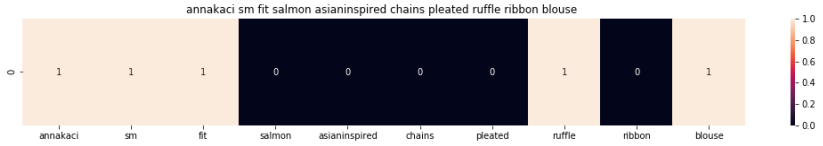


ASIN : B074TLHLMN
Brand: Proenza Schouler
Title: proenza schouler black polka dot blouse 2
Euclidean similarity with the query image : 3.4641016151377544
==========

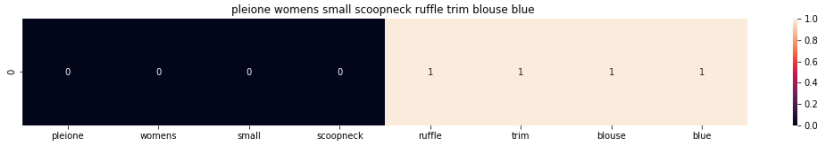


ASIN : B074F5BP5F
Brand: On Twelfth
Title: twelfth womens blouse blue
Euclidean similarity with the query image : 3.4641016151377544
==========

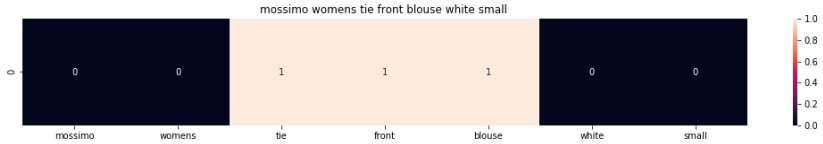
ASIN : B016P80OKQ
Brand: Studio M
Title: studio blue blouse size
Euclidean similarity with the query image : 3.4641016151377544
==========



ASIN : B007KSG42S
Brand: Anna-Kaci
Title: annakaci sm fit salmon asianinspired chains pleated ruffle ribbon blou
se
Euclidean similarity with the query image : 3.4641016151377544
==========

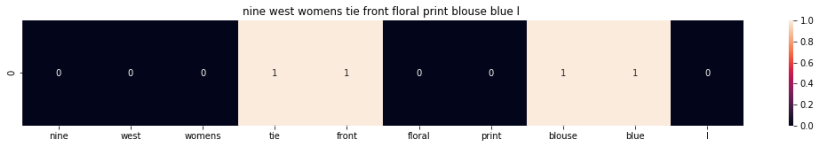


ASIN : B072VHTT1D
Brand: Pleione
Title: pleione womens small scoopneck ruffle trim blouse blue
Euclidean similarity with the query image : 3.4641016151377544
==========

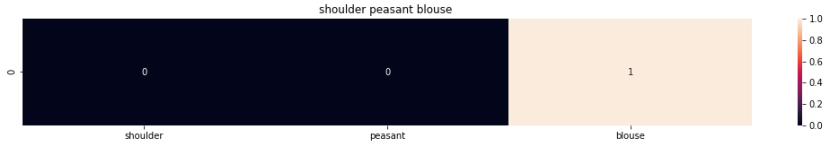


ASIN : B07111HHX6
Brand: Mossimo
Title: mossimo womens tie front blouse white small
Euclidean similarity with the query image : 3.605551275463989
==========

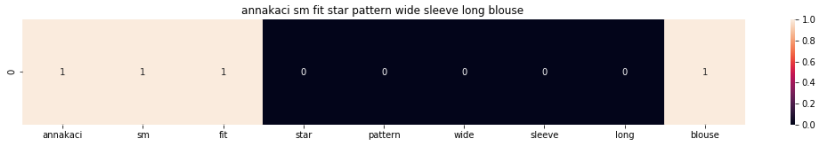
ASIN : B01N3SAT1F
Brand: Ganesh
Title: ganesh womens silkblend blouse 2 blue
Euclidean similarity with the query image : 3.605551275463989
==========



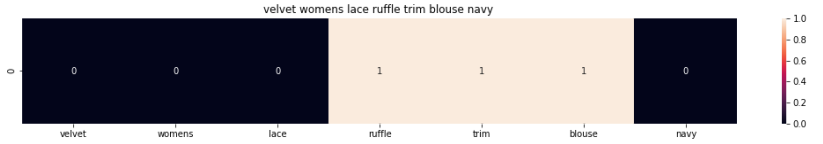
ASIN : B06WW5C6NJ
Brand: Nine West
Title: nine west womens tie front floral print blouse blue l
Euclidean similarity with the query image : 3.605551275463989
==========



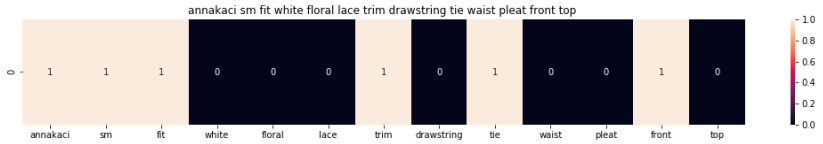
ASIN : B01E1QD5PK
Brand: CHASER
Title: shoulder peasant blouse
Euclidean similarity with the query image : 3.605551275463989
==========



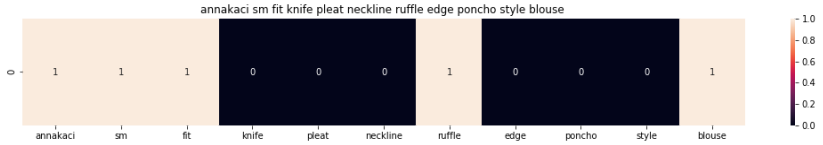
ASIN : B00G5RYY18
Brand: Anna-Kaci
Title: annakaci sm fit star pattern wide sleeve long blouse
Euclidean similarity with the query image : 3.605551275463989
==========



ASIN : B06XCZGQLP
Brand: Velvet by Graham & Spencer
Title: velvet womens lace ruffle trim blouse navy
Euclidean similarity with the query image : 3.605551275463989
==========

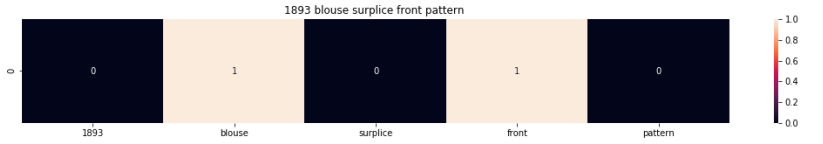


ASIN : B00DW1NKSS
Brand: Anna-Kaci
Title: annakaci sm fit white floral lace trim drawstring tie waist pleat fron
t top
Euclidean similarity with the query image : 3.605551275463989
==========

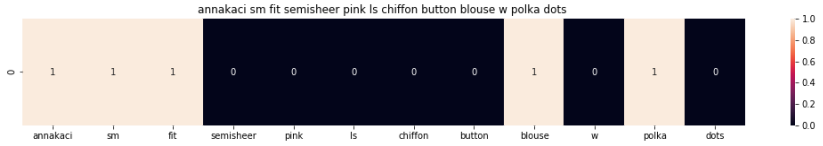


ASIN : B00HCNNOJW
Brand: Anna-Kaci
Title: annakaci sm fit knife pleat neckline ruffle edge poncho style blouse
Euclidean similarity with the query image : 3.605551275463989
==========

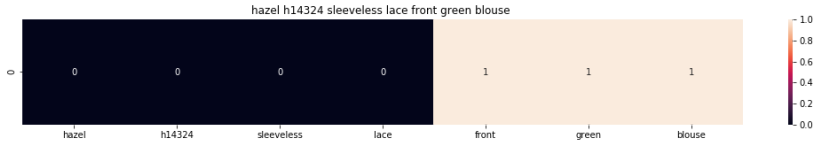
ASIN : B071NDX99J
Brand: CBK
Title: cbk blouse fanny women blue
Euclidean similarity with the query image : 3.605551275463989
==========



ASIN : B00886YXL0
Brand: Ageless Patterns
Title: 1893 blouse surplice front pattern
Euclidean similarity with the query image : 3.605551275463989
==========

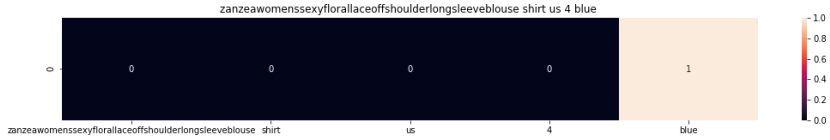


ASIN : B008Z5ST3C
Brand: Anna-Kaci
Title: annakaci sm fit semisheer pink ls chiffon button blouse w polka dots
Euclidean similarity with the query image : 3.605551275463989
==========



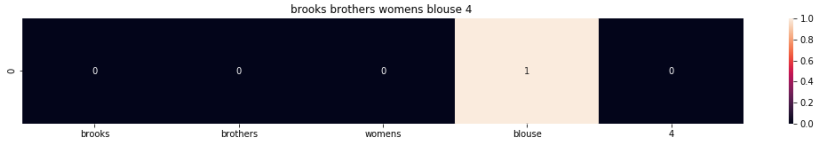
ASIN : B01EAV57MA
Brand: Hazel H14324 Sleeveless Lace Front Green Blouse
Title: hazel h14324 sleeveless lace front green blouse
Euclidean similarity with the query image : 3.605551275463989
==========

ASIN : B074R9MRKY
Brand: Cynthia Rowley
Title: silk georgette blouse l white
Euclidean similarity with the query image : 3.7416573867739413
==========



ASIN : B01192N6GC
Brand: ZANZEA
Title: zanzeawomenssexyflorallaceoffshoulderlongsleeveblouse shirt us 4 blue
Euclidean similarity with the query image : 3.7416573867739413
==========

ASIN : B01M716OVC
Brand: Esley
Title: esley womens scalloped blouse l
Euclidean similarity with the query image : 3.7416573867739413
==========

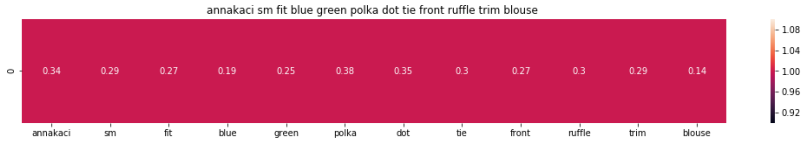


ASIN : B01M9FHVL3
Brand: Brooks Brothers
Title: brooks brothers womens blouse 4
Euclidean similarity with the query image : 3.7416573867739413
==========

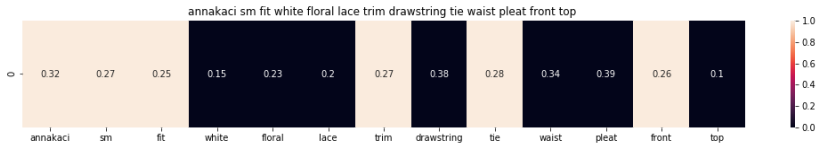
[8.5] TF-IDF based product similarity

In [12]:
```
tfidf_title_vectorizer = TfidfVectorizer(min_df = 0)
tfidf_title_features = tfidf_title_vectorizer.fit_transform(data['title'])
# tfidf_title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(courpus) returns the a sparase matrix of dim
ensions #data_points * #words_in_corpus
# tfidf_title_features[doc_id, index_of_word_in_corpus] = tfidf values of the
 word in given doc
```

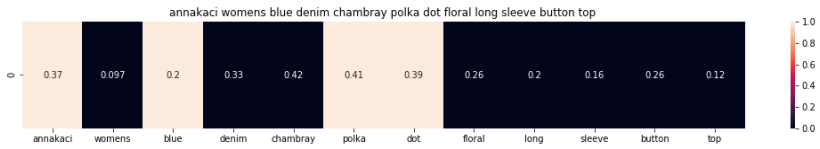
In [13]:
```
def tfidf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

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

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

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

    for i in range(0,len(indices)):
        # we will pass 1. doc_id, 2. title1, 3. title2, url, model
        get_result(indices[i], data['title'].loc[df_indices[0]], data['title']
.loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]], 'tfidf')
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print('BRAND :',data['brand'].loc[df_indices[i]])
        print ('Eucliden distance from the given image :', pdists[i])
        print('='*125)
tfidf_model(931, 25)
# in the output heat map each value represents the tfidf values of the label w
ord, the color represents the intersection with inputs title
```



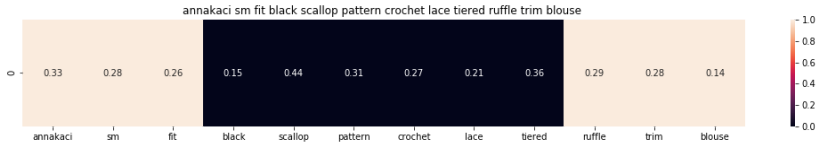
ASIN : B00KLHUIBS
BRAND : Anna-Kaci
Eucliden distance from the given image : 0.0
==========



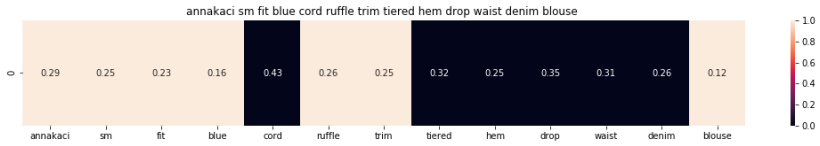
ASIN : B00DW1NKSS
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.0095030470167985
==========



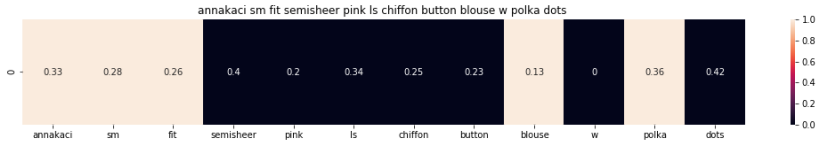
ASIN : B008SMIFN6
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.0417985120306876
==========



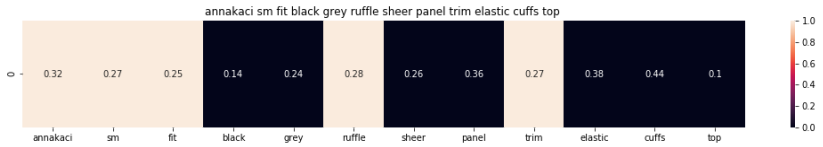
ASIN : B00O194W8W
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.0459904789057266
==========



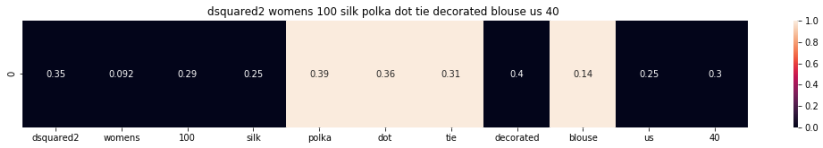
ASIN : B0759G15ZX
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.0670382891349643
==========



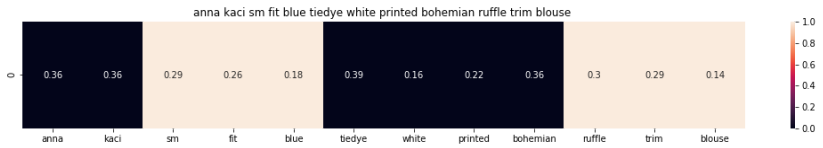
ASIN : B008Z5ST3C
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.079194342831273
==========
==========



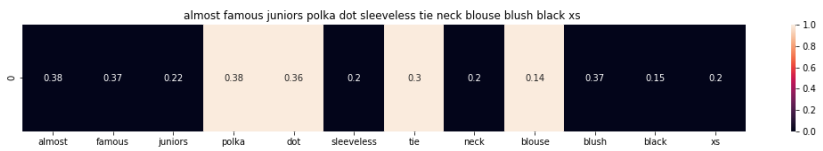
ASIN : B00OIBU11K
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.0810642547494658
==========
==========



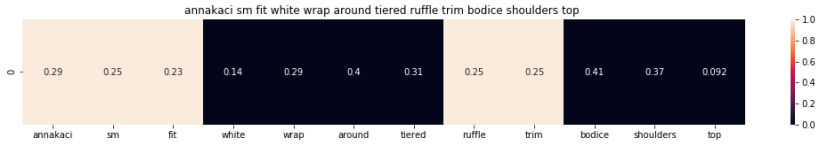
ASIN : B01AYBH28M
BRAND : DSQUARED2
Eucliden distance from the given image : 1.1040520792000035
==========
==========



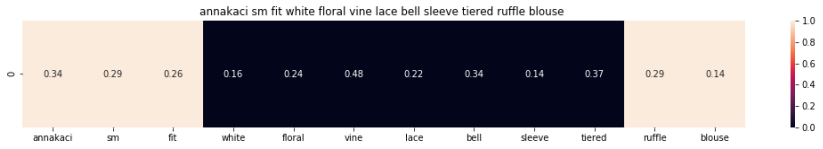
ASIN : B00YQ8S4K0
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.1106318669558235
==========
==========



ASIN : B0745J9HNS
BRAND : Almost Famous
Eucliden distance from the given image : 1.1114784382103975
==========
==========

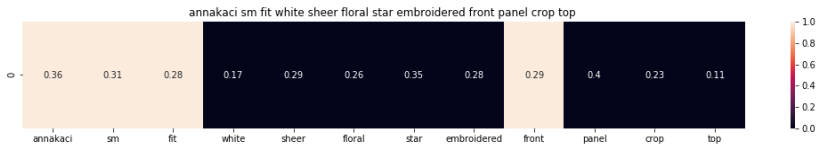


ASIN : B00LMKGFS8
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.113963394862801
==========
==========

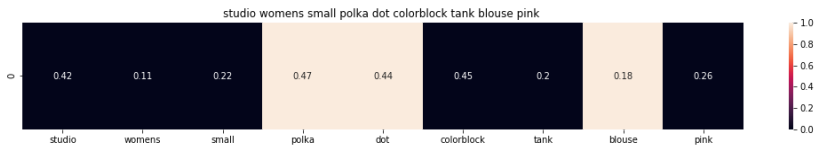


ASIN : B00DVOAWM8
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.115670616192694
==========
==========

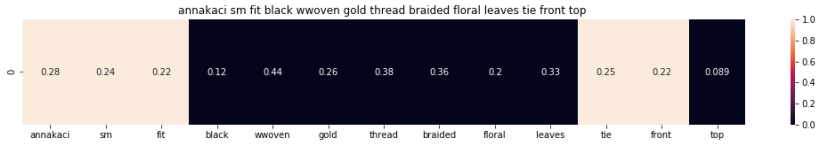
ASIN : B00OPFYBXI
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.1246975675554731
==========
==========



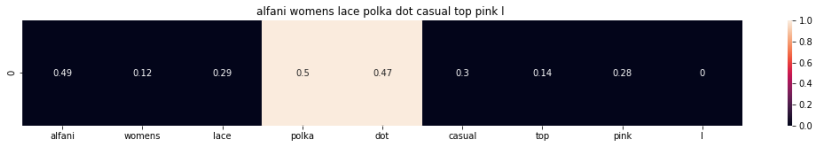
ASIN : B00NAB1R8A
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.124773884923174
==========
==========



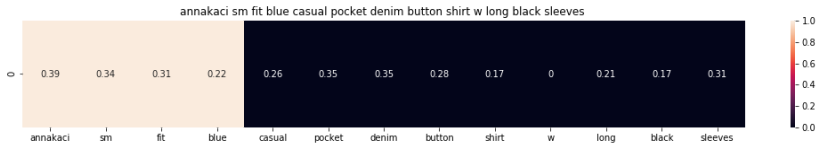
ASIN : B0721KC2HT
BRAND : Studio M
Eucliden distance from the given image : 1.129816389682597
==========
==========



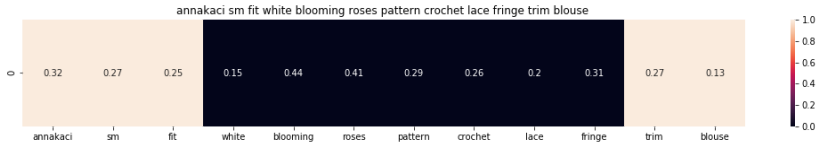
ASIN : B00E7Z8DWQ
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.13146940297404
==========
==========



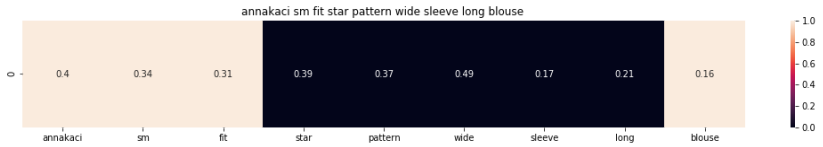
ASIN : B07125NJJ2
BRAND : Alfani
Eucliden distance from the given image : 1.1325545037426967
==========
==========



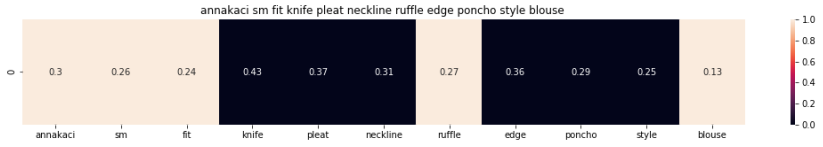
ASIN : B0097LQOOY
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.134493010198643
==========
==========



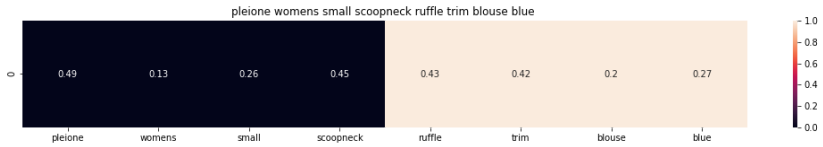
ASIN : B00RDLAR2A
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.1394888920467072
==========
==========



ASIN : B00G5RYY18
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.1474660726278398
==========
==========

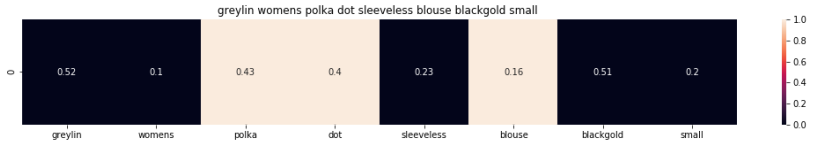


ASIN : B00HCNNOJW
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.1475606961167006
==========
==========

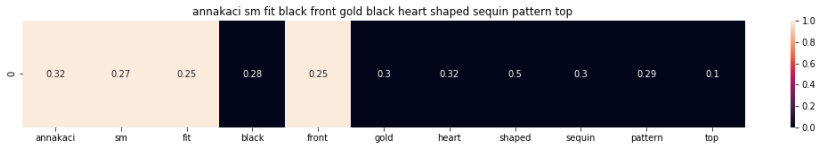


ASIN : B072VHTT1D
BRAND : Pleione
Eucliden distance from the given image : 1.1598897534751225
==========
==========

ASIN : B00HM9OD8W
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.1609978164371932
==========
==========



ASIN : B0725BZ69R
BRAND : Greylin
Eucliden distance from the given image : 1.1618892780591146
==========
==========



ASIN : B00SIALRLA
BRAND : Anna-Kaci
Eucliden distance from the given image : 1.1627646013394934
==========
==========

[8.5] IDF based product similarity

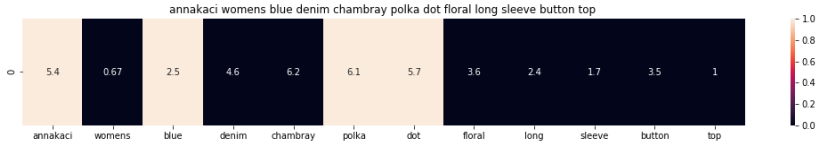
In [14]:
```
idf_title_vectorizer = CountVectorizer()
idf_title_features = idf_title_vectorizer.fit_transform(data['title'])

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

In [15]:
```
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)))
```

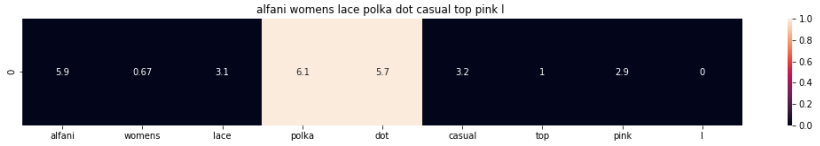
In [16]:
```
# we need to convert the values into float
idf_title_features  = idf_title_features.astype(np.float)

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

    # to calculate idf_title_features we need to replace the count values with
the idf values of the word
    # idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0]
 will return all documents in which the word i present
    for j in idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzer
o()[0]:

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

In [17]:
```
def idf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

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

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

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

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



idf_model(931,25)
# in the output heat map each value represents the idf values of the label wor
d, the color represents the intersection with inputs title
```

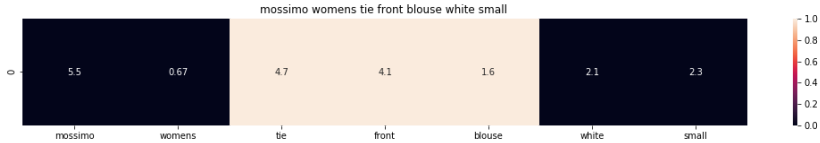
ASIN : B00KLHUIBS
Brand : Anna-Kaci
euclidean distance from the given image : 0.0
==========
==========



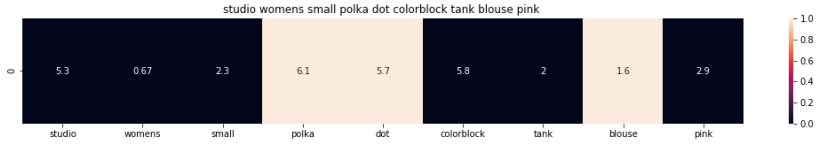
ASIN : B008SMIFN6
Brand : Anna-Kaci
euclidean distance from the given image : 15.046151721911697
==========
==========



ASIN : B07125NJJ2
Brand : Alfani
euclidean distance from the given image : 15.222769212649535
==========
==========

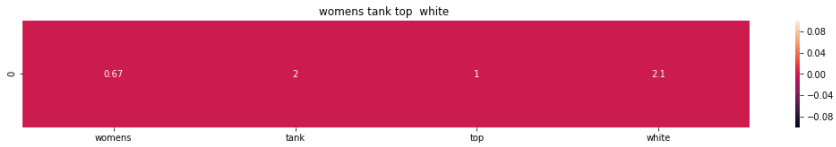


ASIN : B07111HHX6
Brand : Mossimo
euclidean distance from the given image : 15.3415518161122
==========
==========

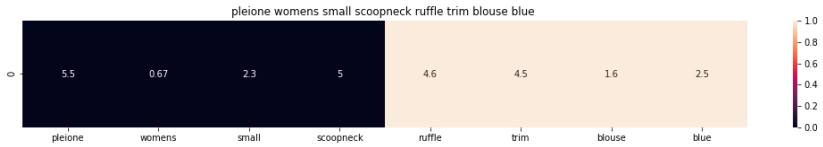


ASIN : B0721KC2HT
Brand : Studio M
euclidean distance from the given image : 15.627349276848992
==========
==========

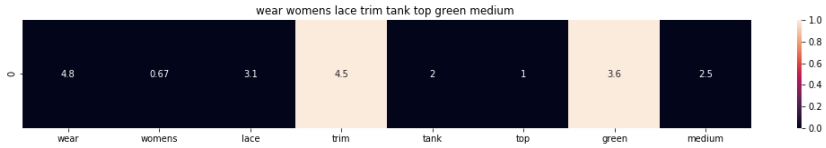
ASIN : B01KXEE20Q
Brand : ouwoow
euclidean distance from the given image : 15.685249209812426
==========
==========



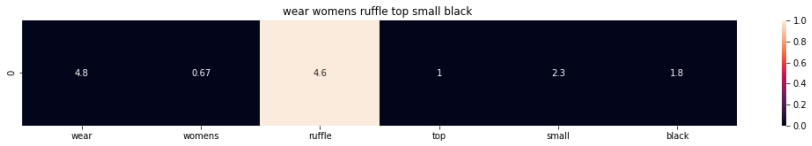
ASIN : B00JPOZ9GM
Brand : Sofra
euclidean distance from the given image : 15.698624105009333
==========
==========



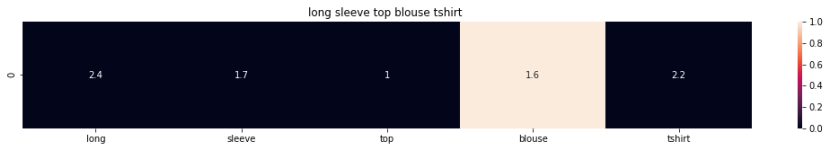
ASIN : B072VHTT1D
Brand : Pleione
euclidean distance from the given image : 15.733159164541727
==========
==========



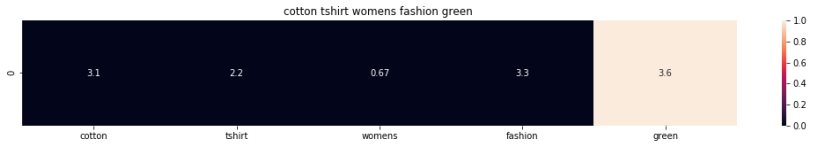
ASIN : B0722TS9DK
Brand : Who What Wear
euclidean distance from the given image : 15.746519642051068
==========
==========



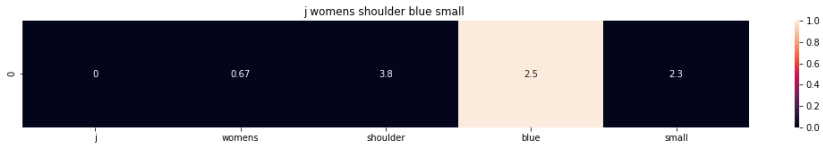
ASIN : B07258X2FK
Brand : Who What Wear
euclidean distance from the given image : 15.76861658770721
==========
==========



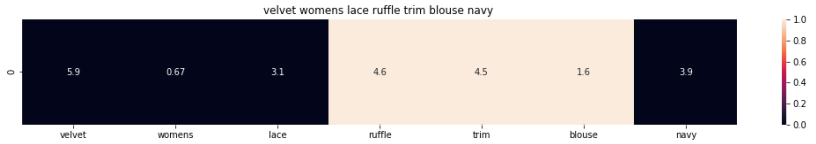
ASIN : B00KF2N5PU
Brand : Vietsbay
euclidean distance from the given image : 15.769570727563098
==========
==========



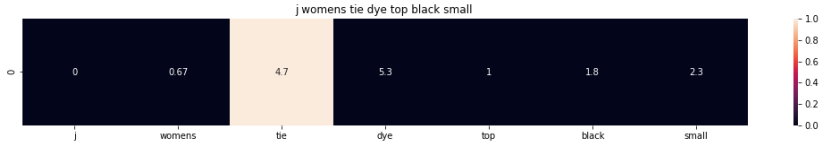
ASIN : B073GJGVBN
Brand : Ivan Levi
euclidean distance from the given image : 15.811562821616887
==========
==========



ASIN : B07583CQFT
Brand : Very J
euclidean distance from the given image : 15.83828054738109
==========
==========



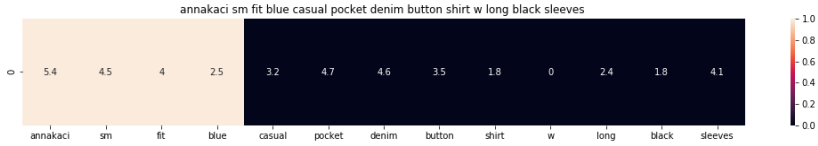
ASIN : B06XCZGQLP
Brand : Velvet by Graham & Spencer
euclidean distance from the given image : 15.878781785172018
==========
==========



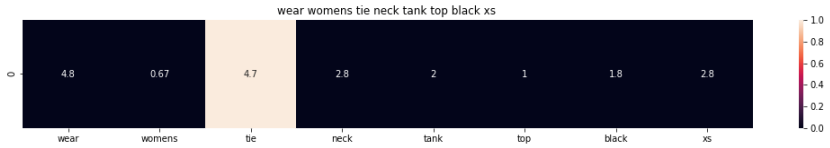
ASIN : B075831F14
Brand : Very J
euclidean distance from the given image : 15.928749759426031
==========
==========

ASIN : B074G5G5RK
Brand : ERMANNO SCERVINO
euclidean distance from the given image : 15.953279501032116
==========
==========

ASIN : B016P80OKQ
Brand : Studio M
euclidean distance from the given image : 16.078823245515473
==========
==========



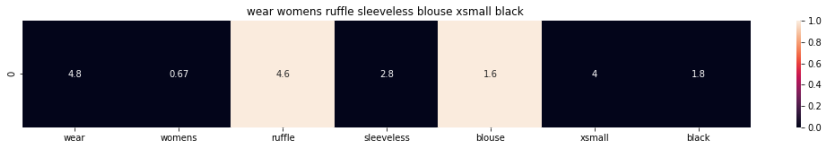
ASIN : B0097LQOOY
Brand : Anna-Kaci
euclidean distance from the given image : 16.11978163937991
==========
==========



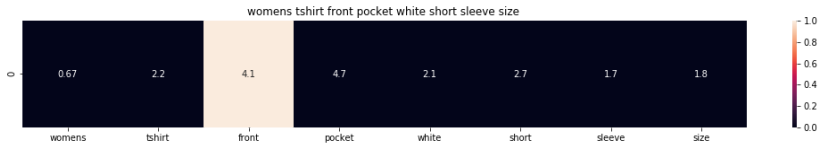
ASIN : B071RYB16J
Brand : Who What Wear
euclidean distance from the given image : 16.18992757930191
==========
==========

ASIN : B074T9KG9Q
Brand : Rain
euclidean distance from the given image : 16.20255923081081
==========
==========

ASIN : B00ZZMYBRG
Brand : HP-LEISURE
euclidean distance from the given image : 16.21635648061094
==========
==========

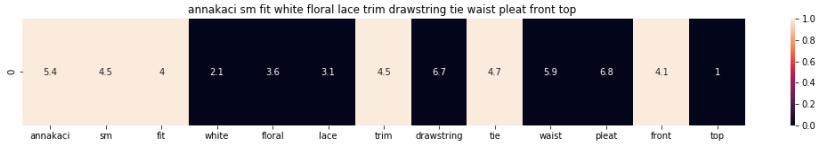


ASIN : B06ZZMQ1YV
Brand : Who What Wear
euclidean distance from the given image : 16.220588637580978
==========
==========



ASIN : B01JR73FSK
Brand : Lofbaz
euclidean distance from the given image : 16.281710938336115
==========
==========

ASIN : B01KM2OLIW
Brand : Voguegirl
euclidean distance from the given image : 16.3374253918477
==========
==========



ASIN : B00DW1NKSS
Brand : Anna-Kaci
euclidean distance from the given image : 16.3433598514525
==========
==========

[9] Text Semantics based product similarity

In [18]:
```
# credits: https://www.kaggle.com/c/word2vec-nlp-tutorial#part-2-word-vectors
# Custom Word2Vec using your own text data.
# Do NOT RUN this code.
# It is meant as a reference to build your own Word2Vec when you have
# lots of data.

'''
# Set values for various parameters
num_features = 300    # Word vector dimensionality
min_word_count = 1    # Minimum word count
num_workers = 4       # Number of threads to run in parallel
context = 10          # Context window size
downsampling = 1e-3   # Downsample setting for frequent words

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

'''
```

Out[18]:
```
'\n# Set values for various parameters\nnum_features = 300    # Word vector d
imensionality                    \nmin_word_count = 1    # Minimum word cou
nt                       \nnum_workers = 4       # Number of threads to run
in parallel\ncontext = 10          # Context window size
\ndownsampling = 1e-3   # Downsample setting for frequent words\n\n# Initiali
ze and train the model (this will take some time)\nfrom gensim.models import
word2vec\nprint ("Training model...")\nmodel = word2vec.Word2Vec(sen_corpus,
workers=num_workers,            size=num_features, min_count = min_word_coun
t,            window = context)\n    \n'
```

In [19]:
```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

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

'''
model = KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bi
n', binary=True)
'''

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

In [20]:
```
# Utility functions

def get_word_vec(sentence, doc_id, m_name):
    # sentence : title of the apparel
    # doc_id: document id in our corpus
    # m_name: model information it will take two values
        # if  m_name == 'avg', we will append the model[i], w2v representation
of word i
        # if m_name == 'weighted', we will multiply each w2v[word] with the id
f(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.voc
abulary_[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 c
orpus, 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/a
vg) in 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
 length 300 corresponds to each word in give title
    # vec2 = np.array(#number_of_words_title2 * 300), each row is a vector of
 length 300 corresponds to each word in give title

    final_dist = []
    # for each vector in vec1 we caluclate the distance(euclidean) to all vect
ors in vec2
    for i in vec1:
        dist = []
        for j in vec2:
            # np.linalg.norm(i-j) will result the euclidean distance between v
ectors 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 t
itle2)
    # final_dist[i,j] = euclidean distance between vectors i, j
    return np.array(final_dist)


def heat_map_w2v(sentence1, sentence2, url, doc_id1, doc_id2, model):
    # sentance1 : title1, input apparel
    # sentance2 : title2, recommended apparel
    # url: apparel image url
    # doc_id1: document id of input apparel
```

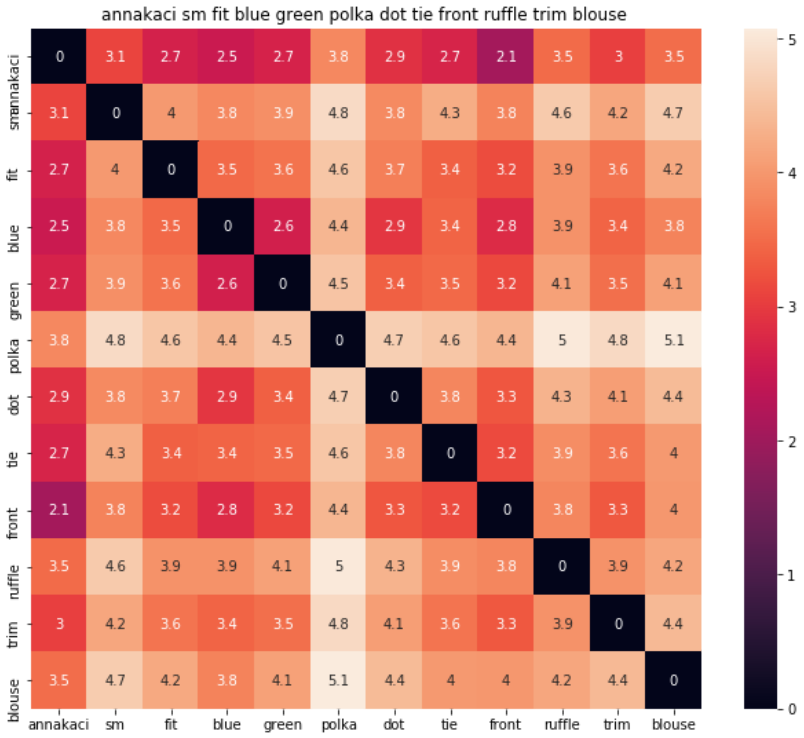
```
    # doc_id2: document id of recommended apparel
    # model: it can have two values, 1. avg 2. weighted

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

    # s1_s2_dist = np.array(#number of words in title1 * #number of words in t
itle2)
    # 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 disp
lays image of apparel
    gs = gridspec.GridSpec(2, 2, width_ratios=[4,1],height_ratios=[2,1])
    fig = plt.figure(figsize=(15,15))

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

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

    plt.show()
```

In [21]:
```
# vocab = stores all the words that are there in google w2v model
# vocab = model.wv.vocab.keys() # if you are using Google word2Vec

vocab = model.keys()
# this function will add the vectors of each word and returns the avg vector o
f given 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 word i
        # if m_name == 'weighted', we will multiply each w2v[word] with the id
f(word)

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

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

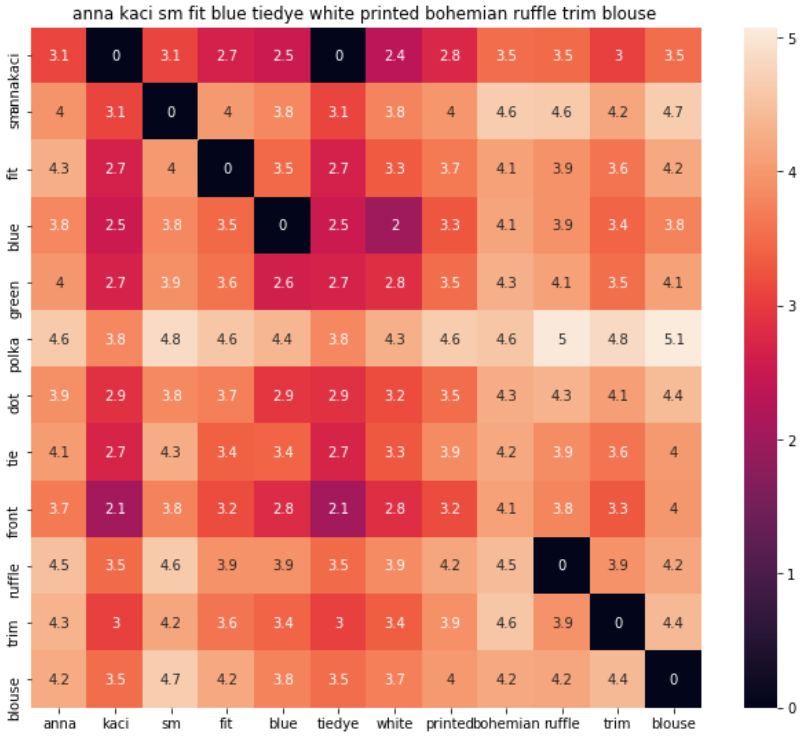
[9.2] Average Word2Vec product similarity.

In [22]:
```
doc_id = 0
w2v_title = []
# for every title we build a avg vector representation
for i in data['title']:
    w2v_title.append(build_avg_vec(i, 300, doc_id,'avg'))
    doc_id += 1

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

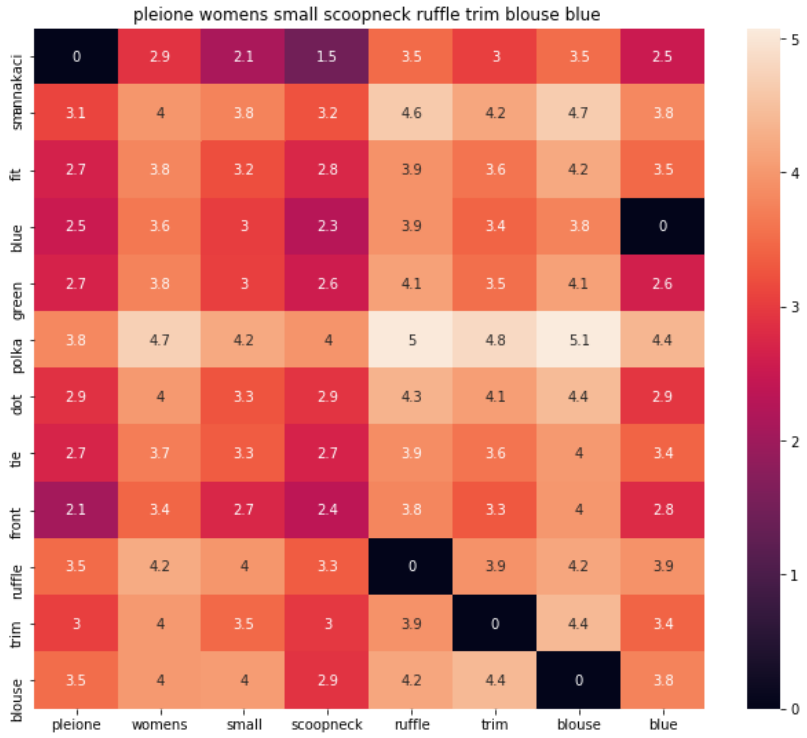
In [23]:
```
def avg_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

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

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

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

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


avg_w2v_model(931, 25)
# in the give heat map, each cell contains the euclidean distance between word
s i, j
```



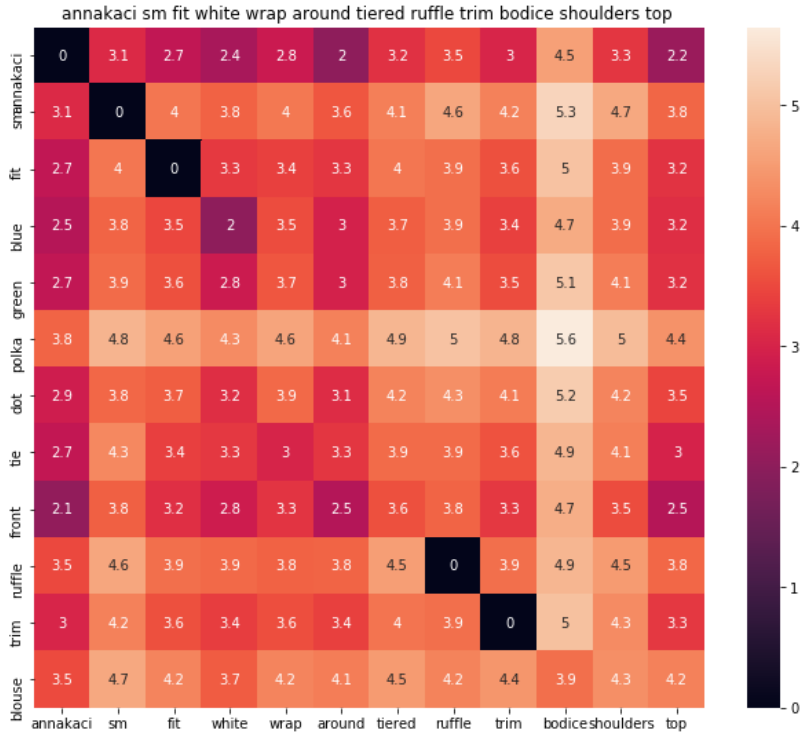
ASIN : B00KLHUIBS
BRAND : Anna-Kaci
euclidean distance from given input image : 0.0
==========
==========



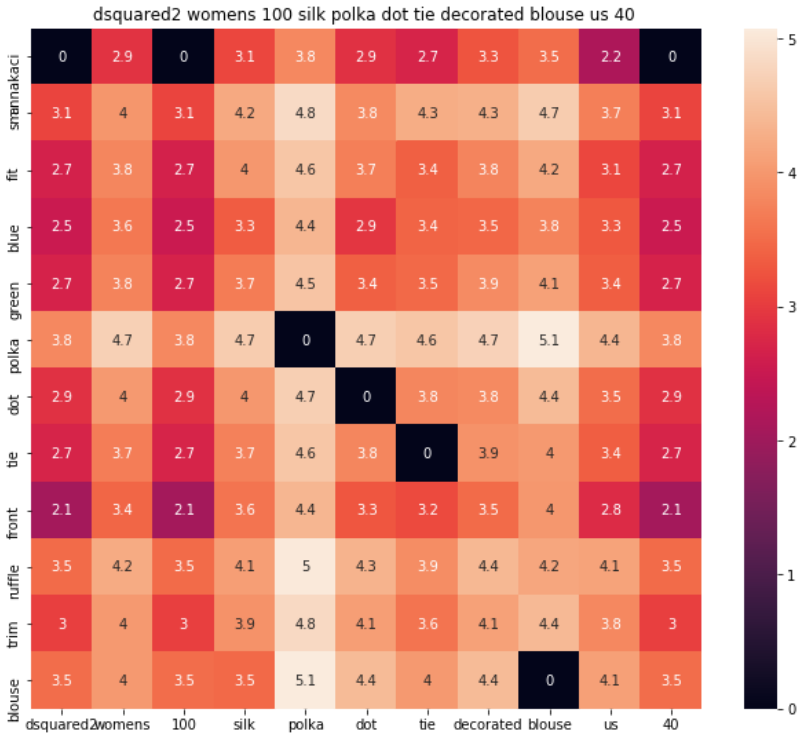
ASIN : B00YQ8S4K0
BRAND : Anna-Kaci
euclidean distance from given input image : 0.669069
==========
==========



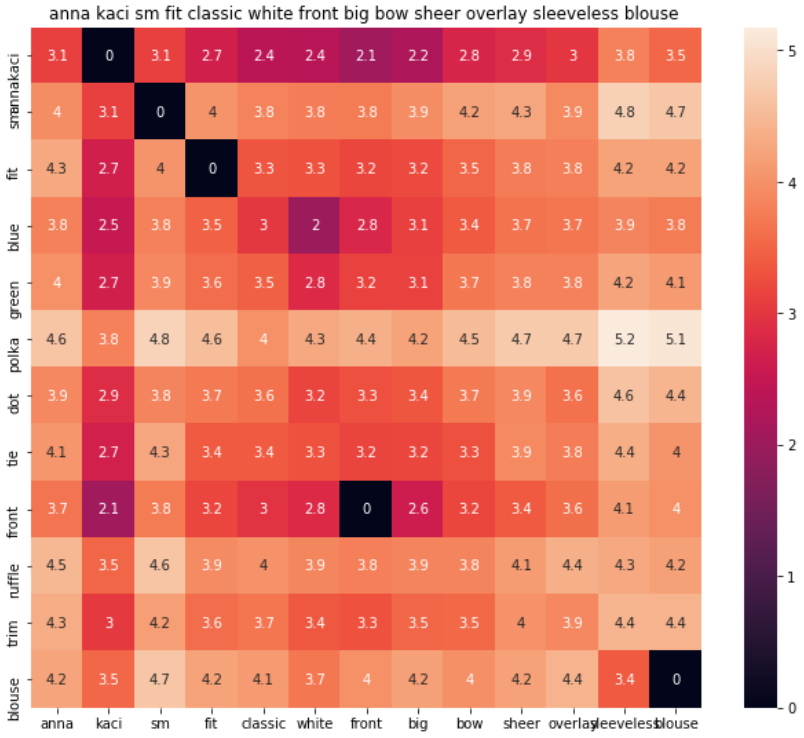
ASIN : B072VHTT1D
BRAND : Pleione
euclidean distance from given input image : 0.79176706
==========
==========



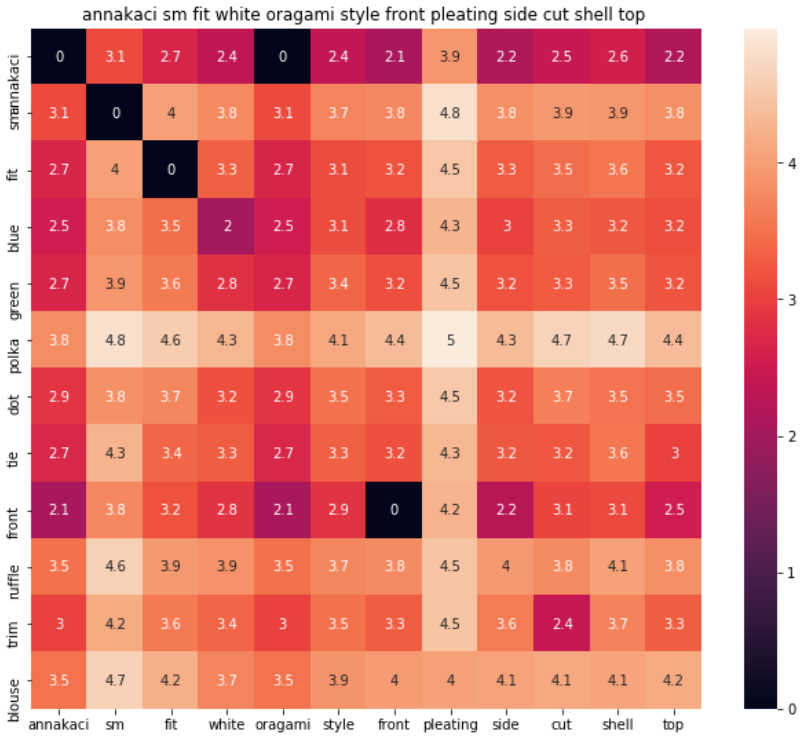
ASIN : B00LMKGFS8
BRAND : Anna-Kaci
euclidean distance from given input image : 0.80942315
==========
==========



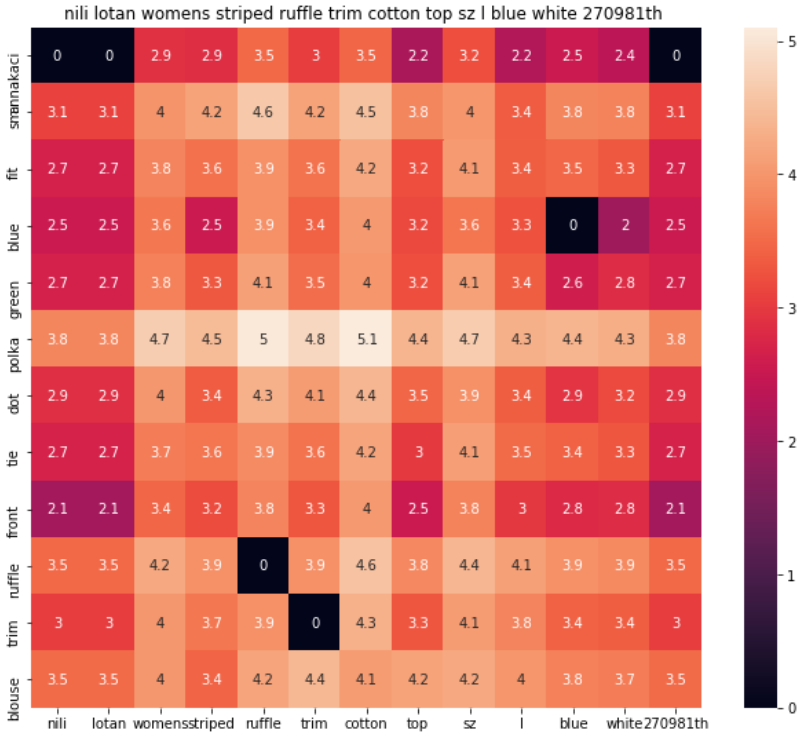
ASIN : B01AYBH28M
BRAND : DSQUARED2
euclidean distance from given input image : 0.8127421
==========
==========



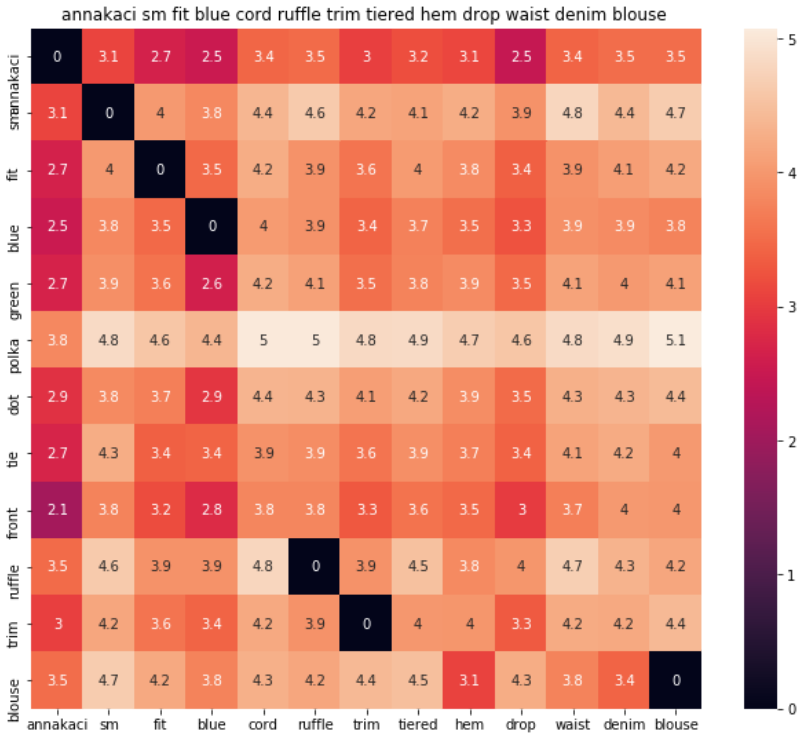
ASIN : B010EH3S02
BRAND : Anna-Kaci
euclidean distance from given input image : 0.8188168
==========
==========



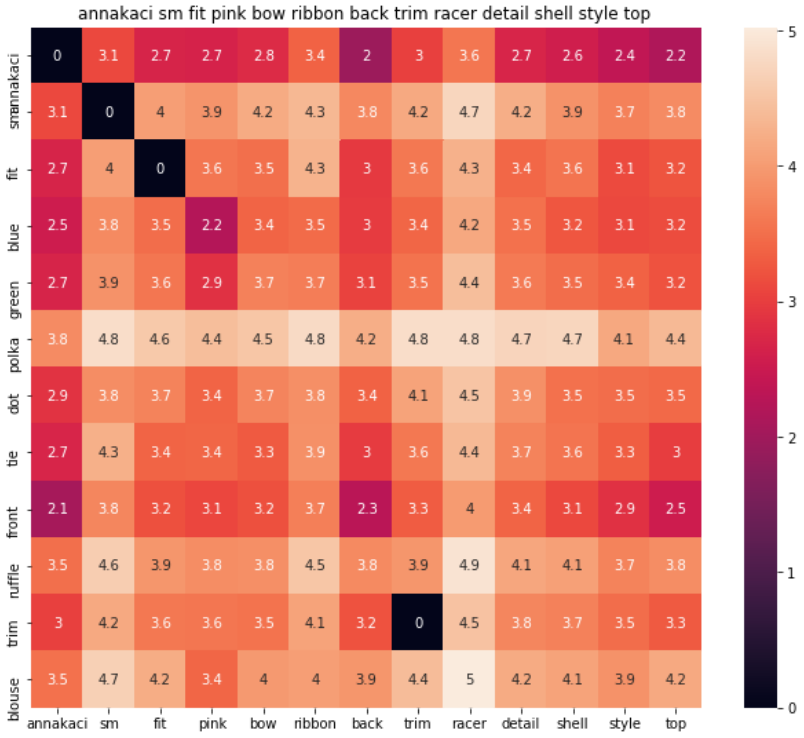
ASIN : B00LU4Z2YY
BRAND : Anna-Kaci
euclidean distance from given input image : 0.819001
==========
==========



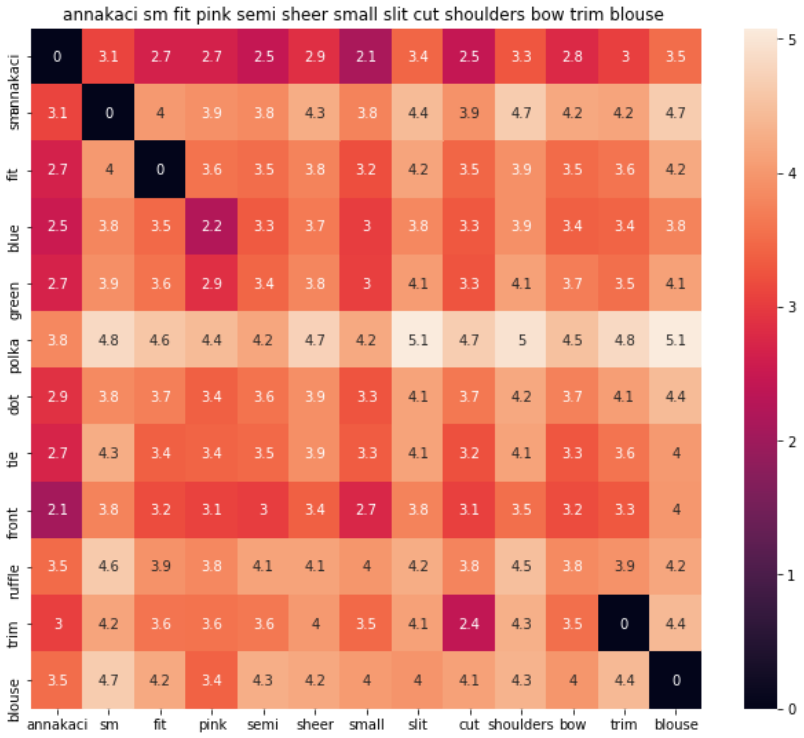
ASIN : B074PFR9CF
BRAND : Nili Lotan
euclidean distance from given input image : 0.821753
==========
==========



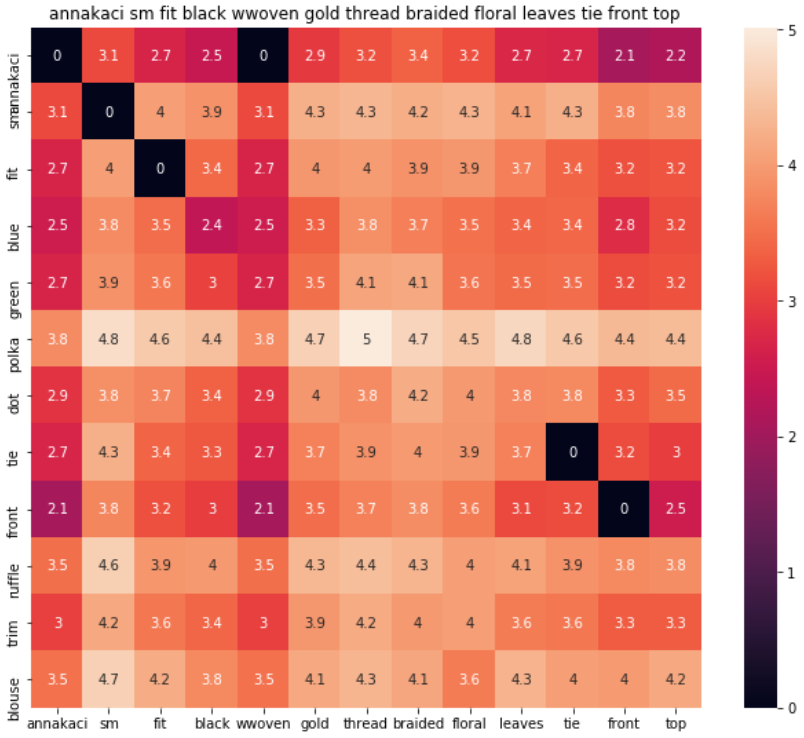
ASIN : B0759G15ZX
BRAND : Anna-Kaci
euclidean distance from given input image : 0.82678145
==========
==========



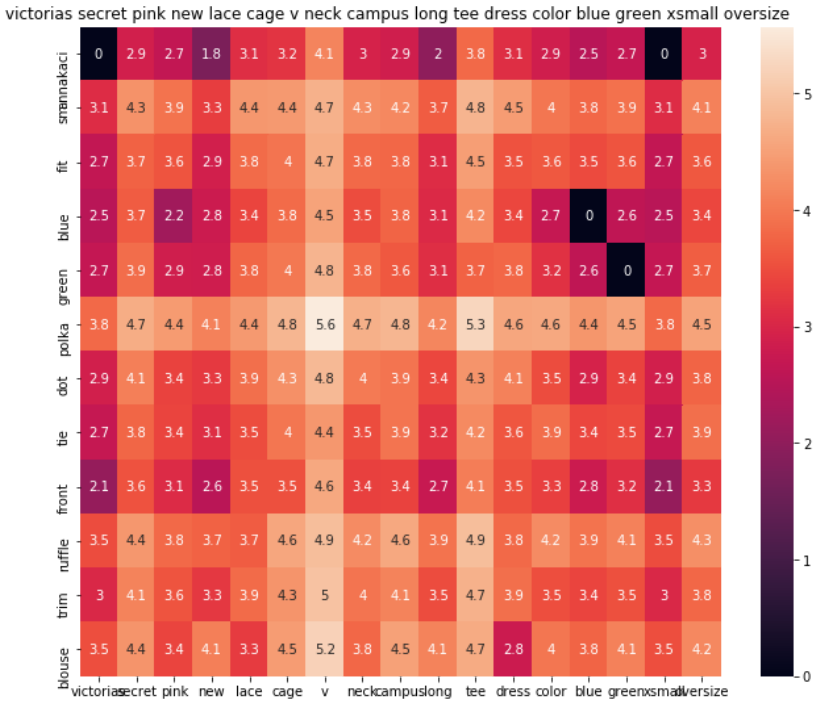
ASIN : B00KOBQEBO
BRAND : Anna-Kaci
euclidean distance from given input image : 0.8268439
==========
==========



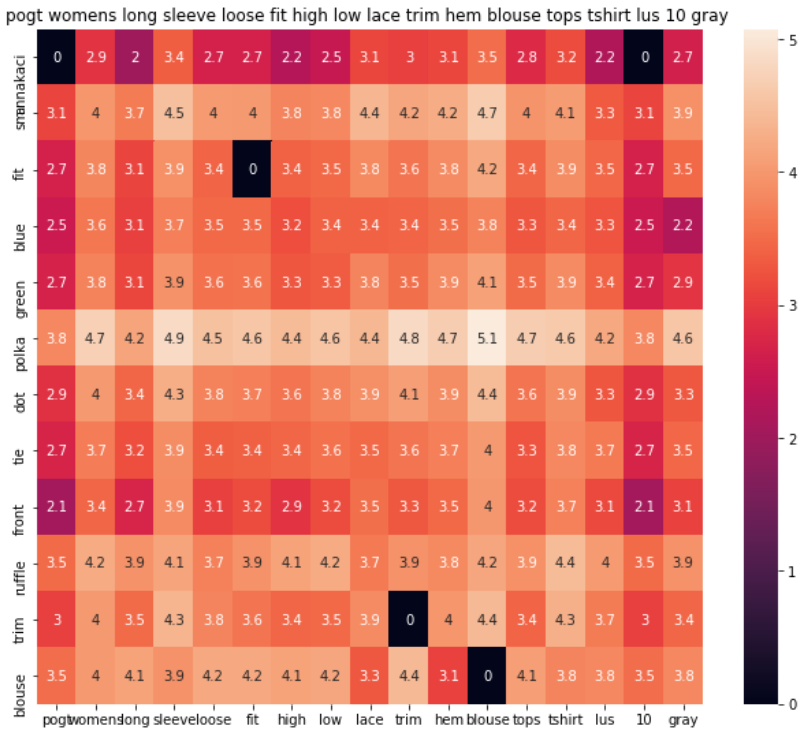
ASIN : B00HM9OD8W
BRAND : Anna-Kaci
euclidean distance from given input image : 0.82737064
==========
==========



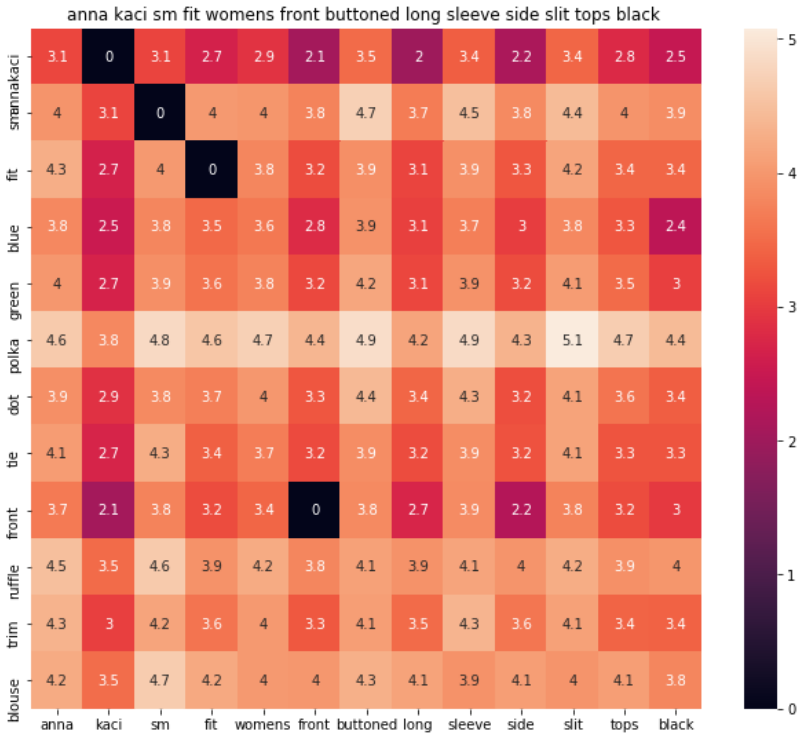
ASIN : B00E7Z8DWQ
BRAND : Anna-Kaci
euclidean distance from given input image : 0.8279004
==========
==========



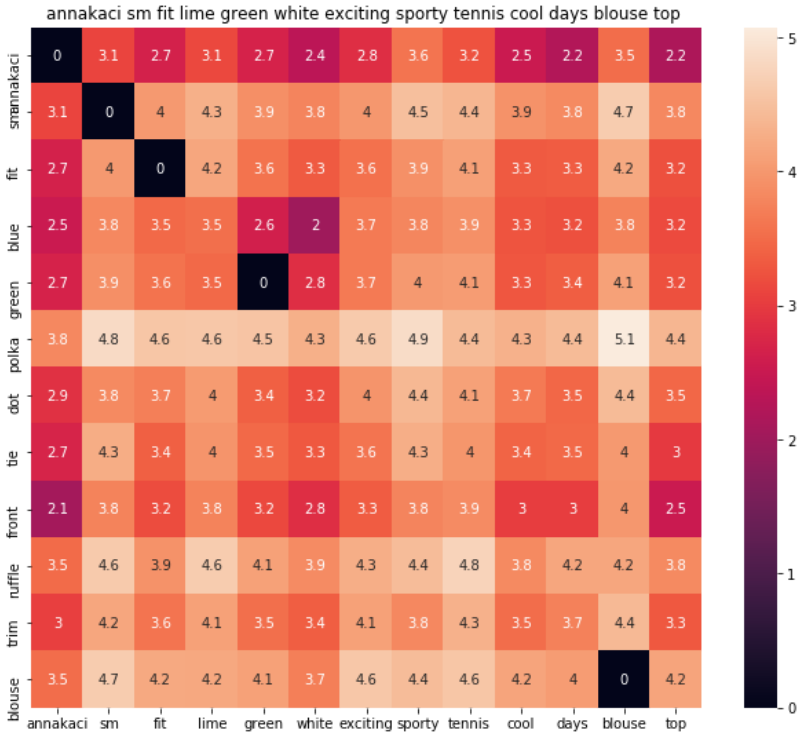
ASIN : B074JKJHNQ
BRAND : VS Pink
euclidean distance from given input image : 0.8330928
==========
==========



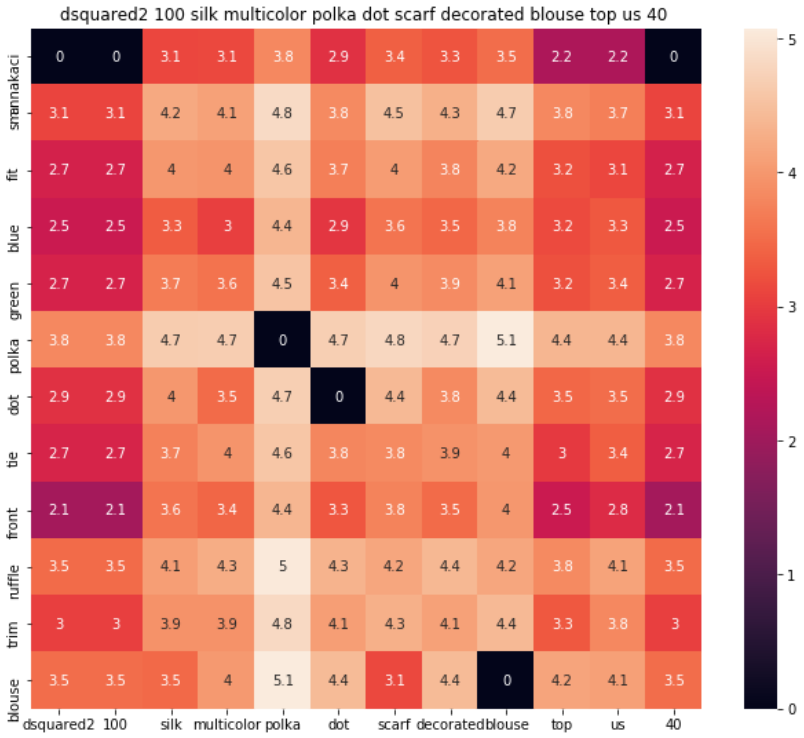
ASIN : B01M19NC0Y
BRAND : POGT
euclidean distance from given input image : 0.83579636
==========
==========



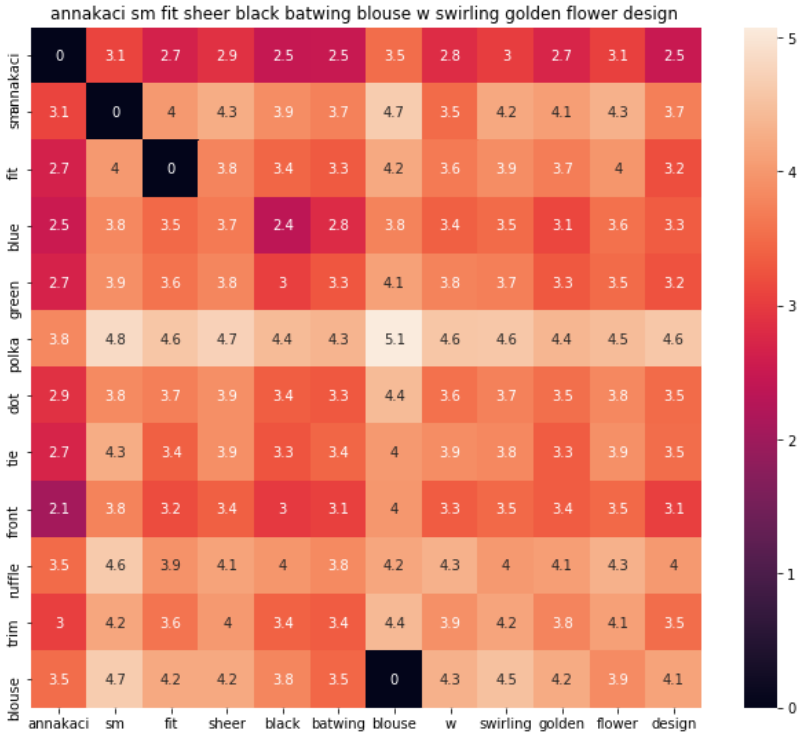
ASIN : B019820A4Q
BRAND : Anna-Kaci
euclidean distance from given input image : 0.8429624
==========
==========



ASIN : B007Y9SZSY
BRAND : Anna-Kaci
euclidean distance from given input image : 0.84324473
==========
==========

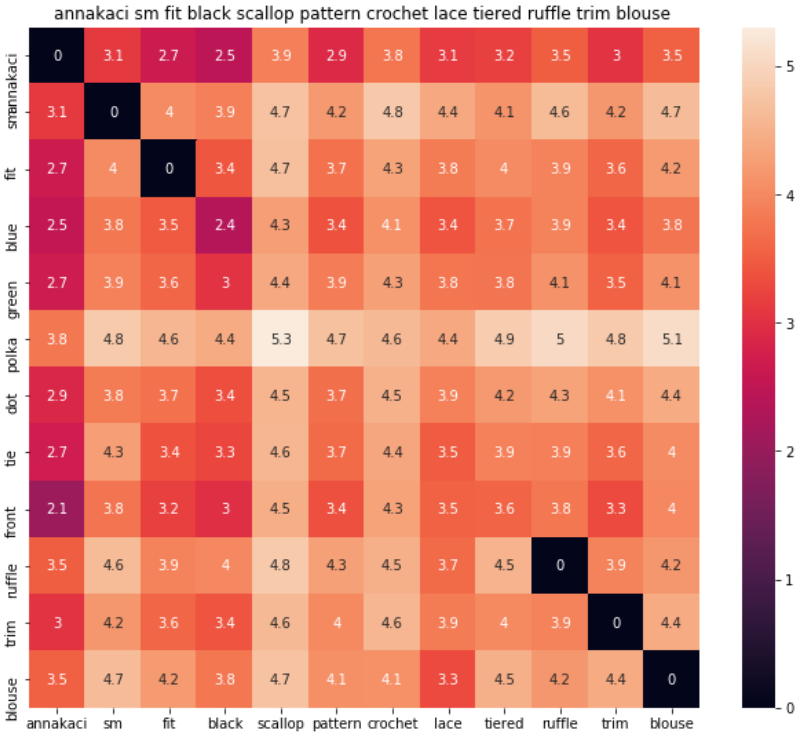


ASIN : B01D6FVFH2
BRAND : DSQUARED2
euclidean distance from given input image : 0.8452494
==========
==========

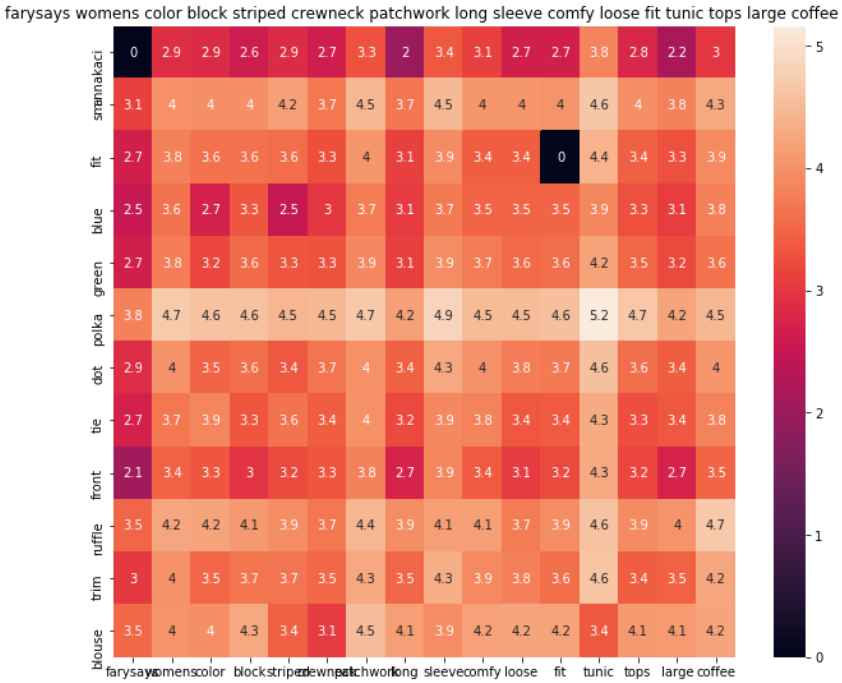


ASIN : B008LE9GA2
BRAND : Anna-Kaci
euclidean distance from given input image : 0.84813684
==========
==========

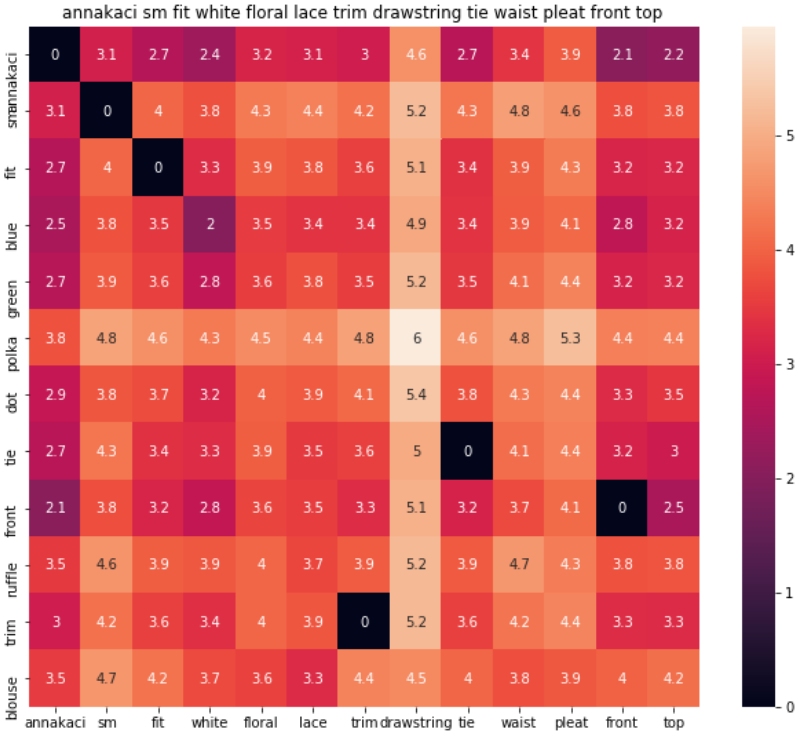
ASIN : B01JCZWOUW
BRAND : No Boundaries
euclidean distance from given input image : 0.8488441
==========
==========



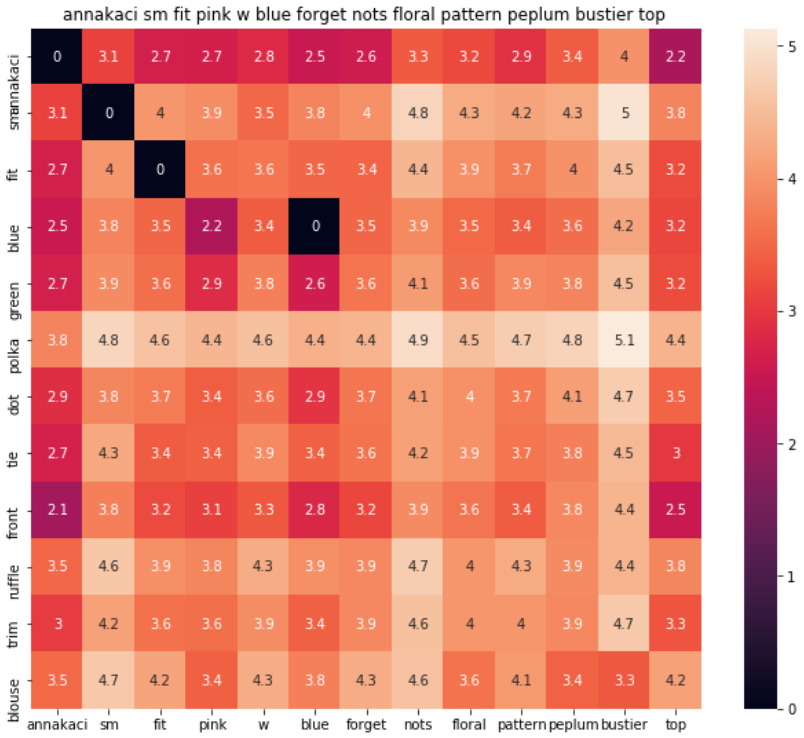
ASIN : B00O194W8W
BRAND : Anna-Kaci
euclidean distance from given input image : 0.8498149
==========
==========



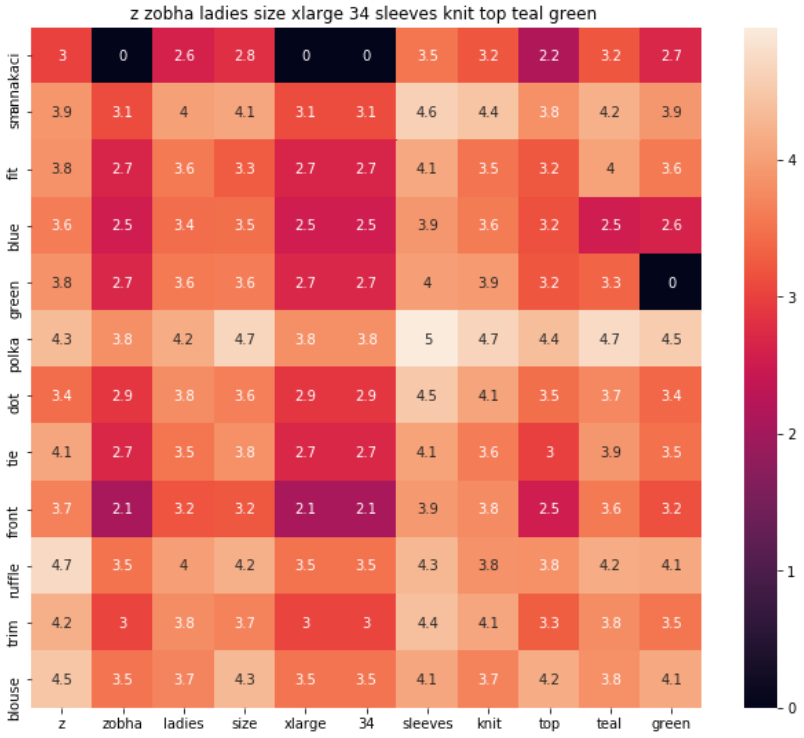
ASIN : B01N1Z7JB7
BRAND : FARYSAYS
euclidean distance from given input image : 0.8523731
==========
==========



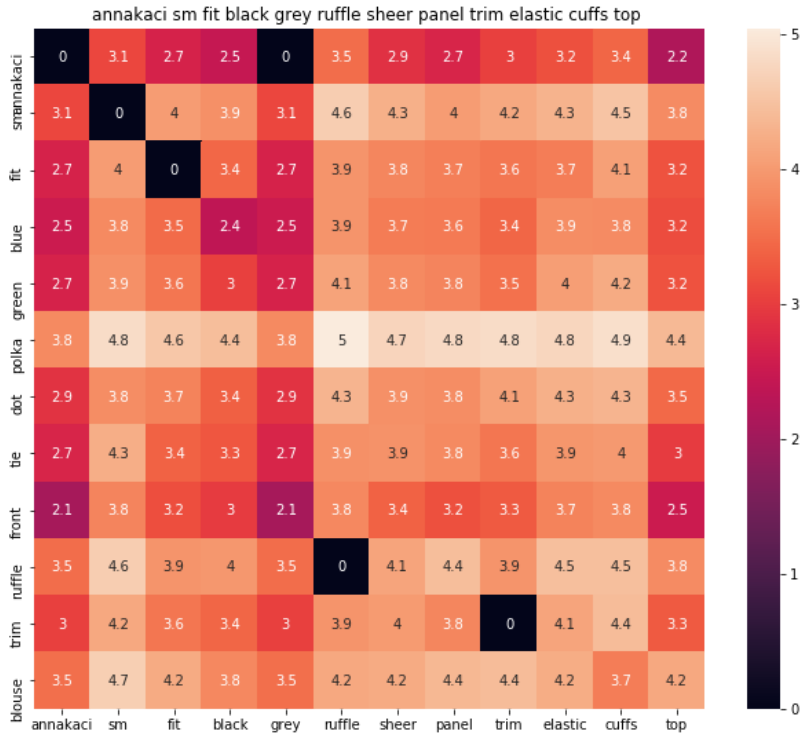
ASIN : B00DW1NKSS
BRAND : Anna-Kaci
euclidean distance from given input image : 0.8536276
==========
==========



ASIN : B00GM4AMU6
BRAND : Anna-Kaci
euclidean distance from given input image : 0.85375154
==========
==========



ASIN : B01FY8UY6C
BRAND : Z by Zobha
euclidean distance from given input image : 0.8559319
==========
==========

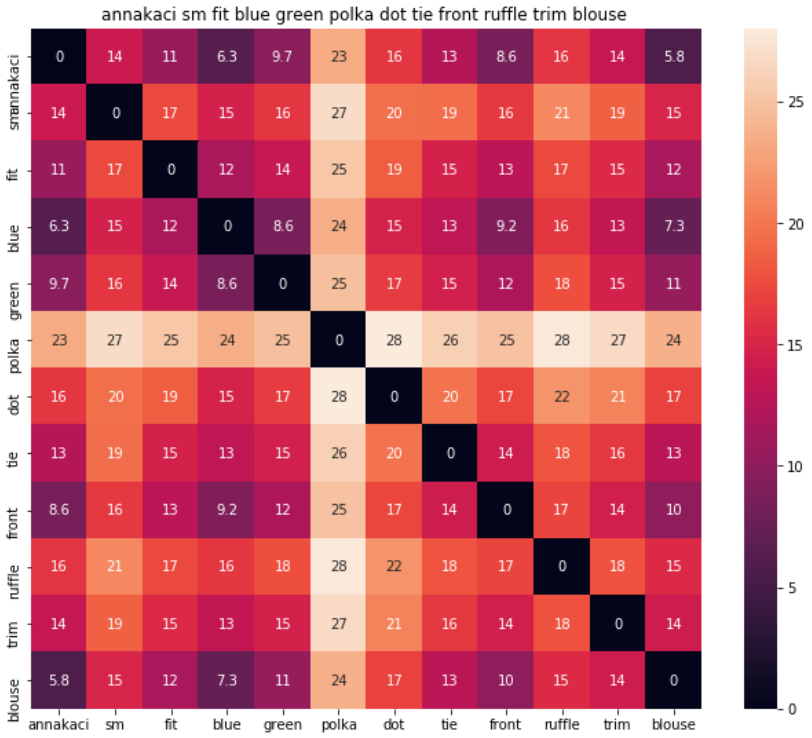


```
ASIN : B00OIBU11K
BRAND : Anna-Kaci
euclidean distance from given input image : 0.8568677
================================================================================
================================================
```

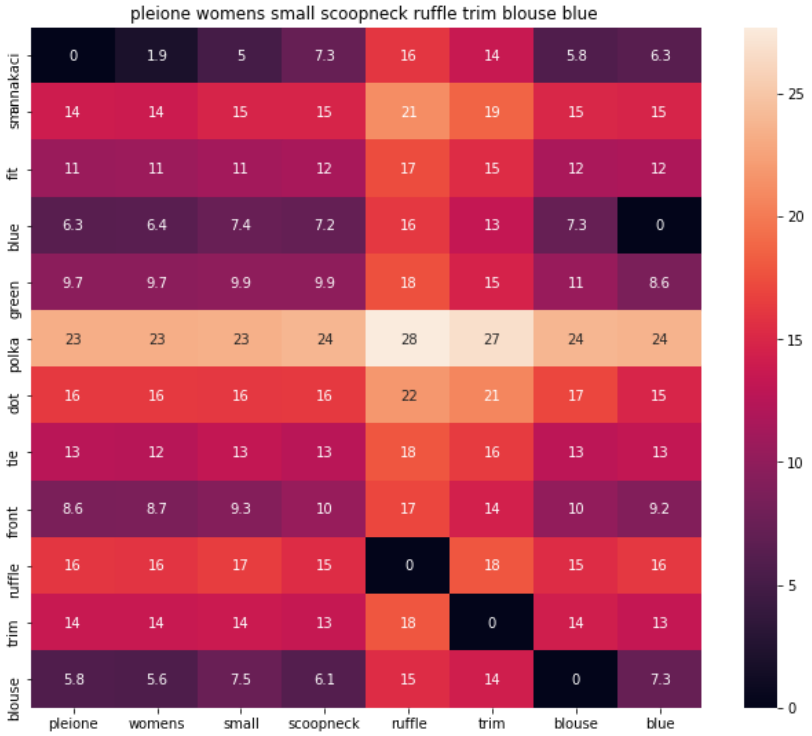
[9.4] IDF weighted Word2Vec for product similarity

In [24]:
```
doc_id = 0
w2v_title_weight = []
# for every title we build a weighted vector representation
for i in data['title']:
    w2v_title_weight.append(build_avg_vec(i, 300, doc_id,'weighted'))
    doc_id += 1
# w2v_title = np.array(# number of doc in courpus * 300), each row corresponds to a doc
w2v_title_weight = np.array(w2v_title_weight)
```

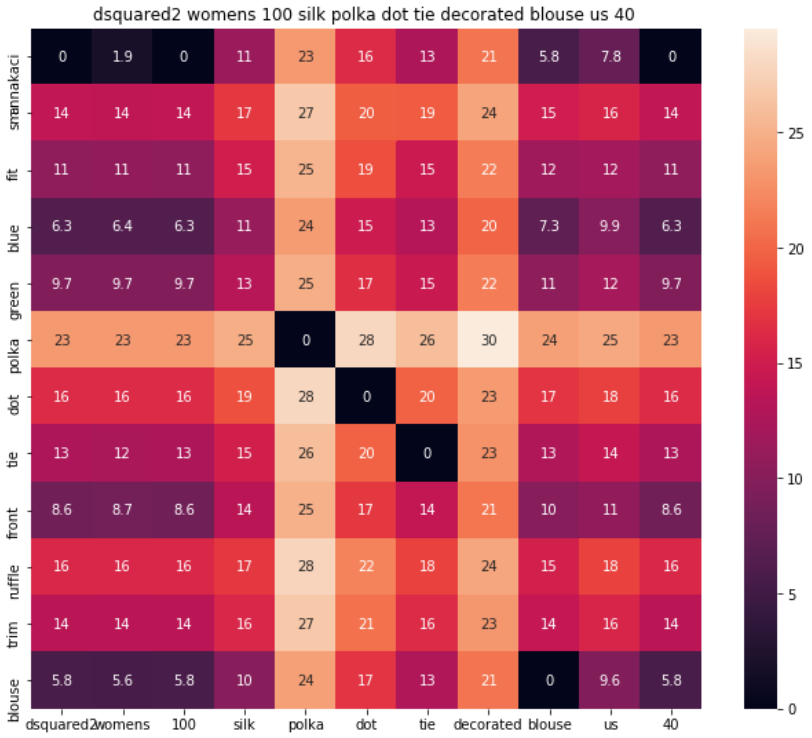
In [25]:
```
def weighted_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

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

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

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

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

weighted_w2v_model(931, 25)
#931
#12566
# in the give heat map, each cell contains the euclidean distance between word
s i, j
```



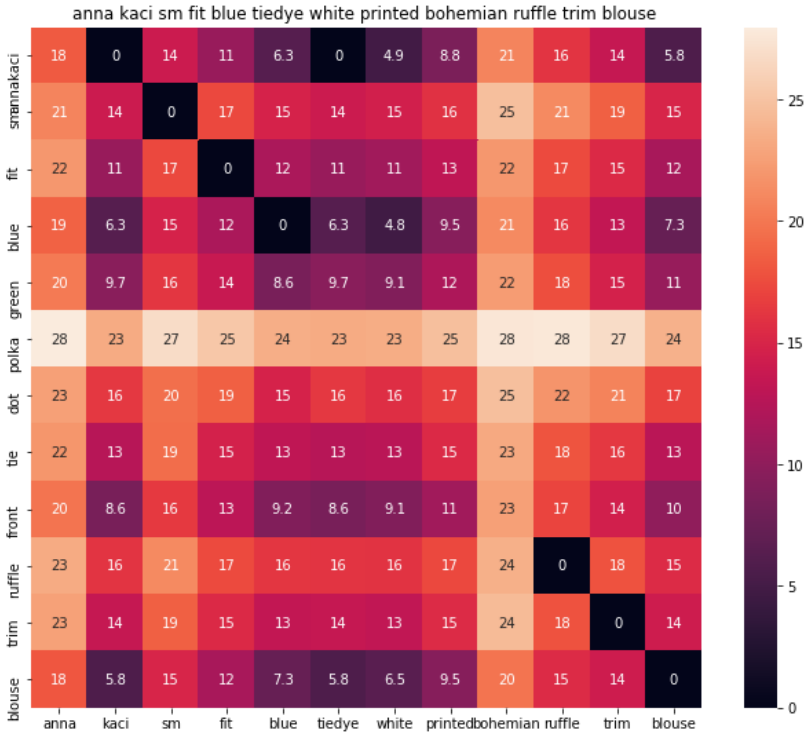
ASIN : B00KLHUIBS
Brand : Anna-Kaci
euclidean distance from input : 0.0
==========
==========



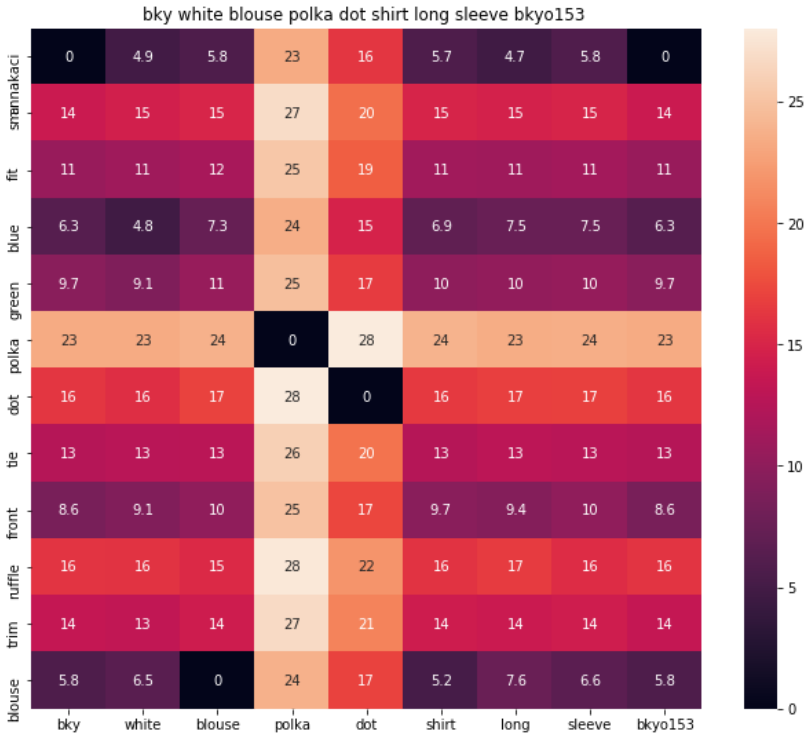
ASIN : B072VHTT1D
Brand : Pleione
euclidean distance from input : 3.304624
==========
==========



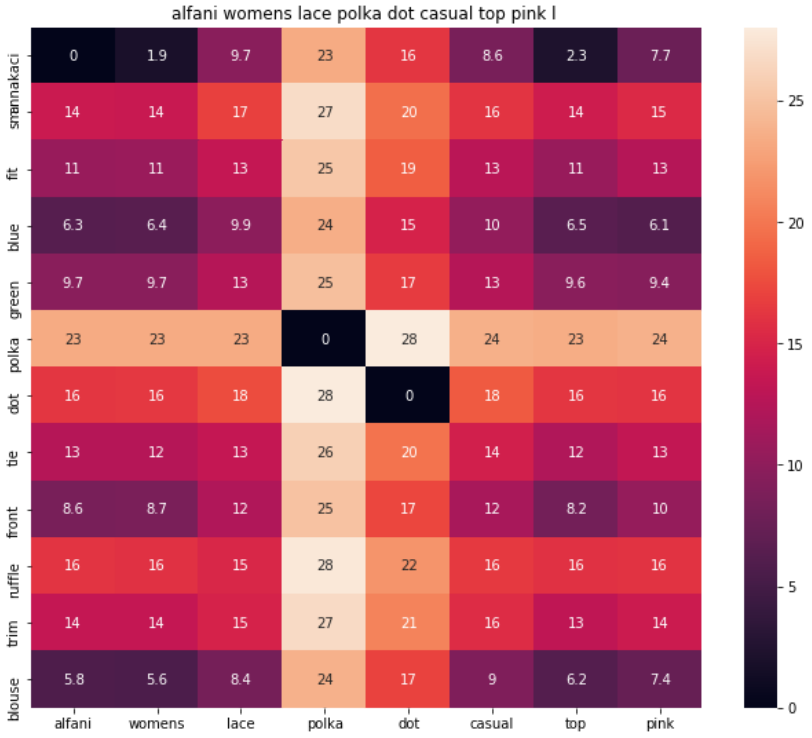
ASIN : B01AYBH28M
Brand : DSQUARED2
euclidean distance from input : 3.3703344
==========
==========



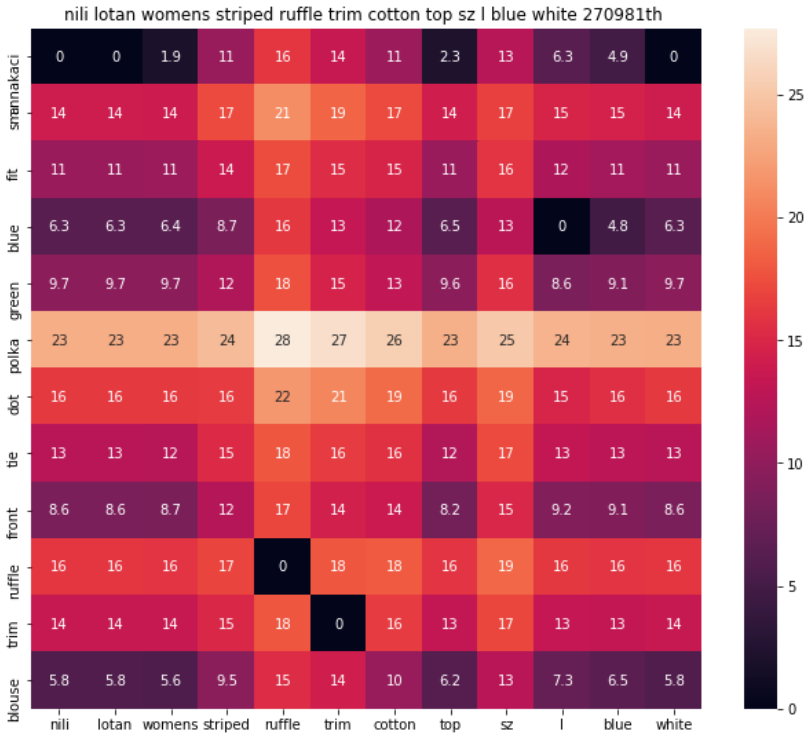
ASIN : B00YQ8S4K0
Brand : Anna-Kaci
euclidean distance from input : 3.419353
==========
==========



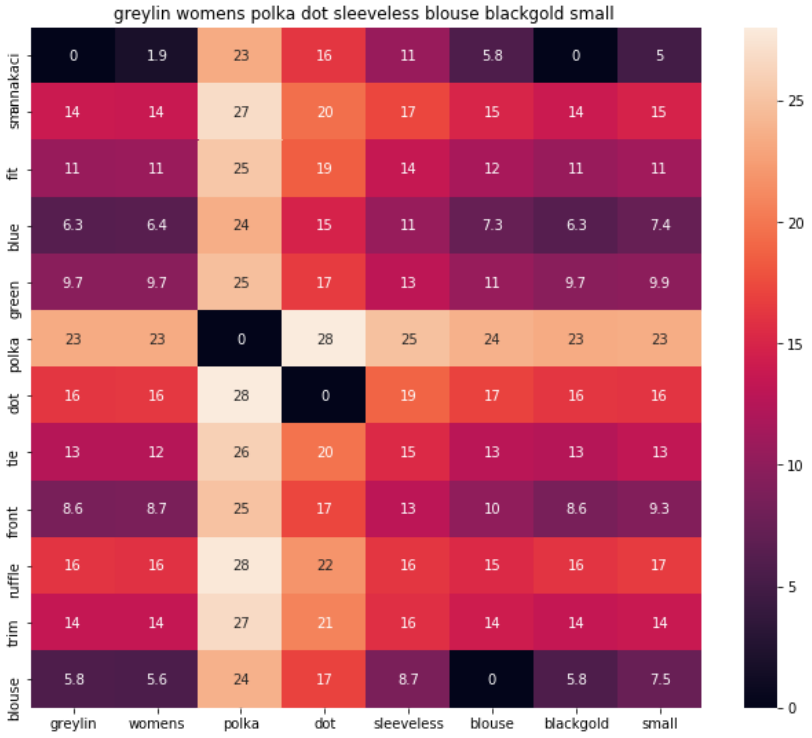
ASIN : B00ZD1LI16
Brand : bankhunyabangyai store
euclidean distance from input : 3.4255989
==========
==========



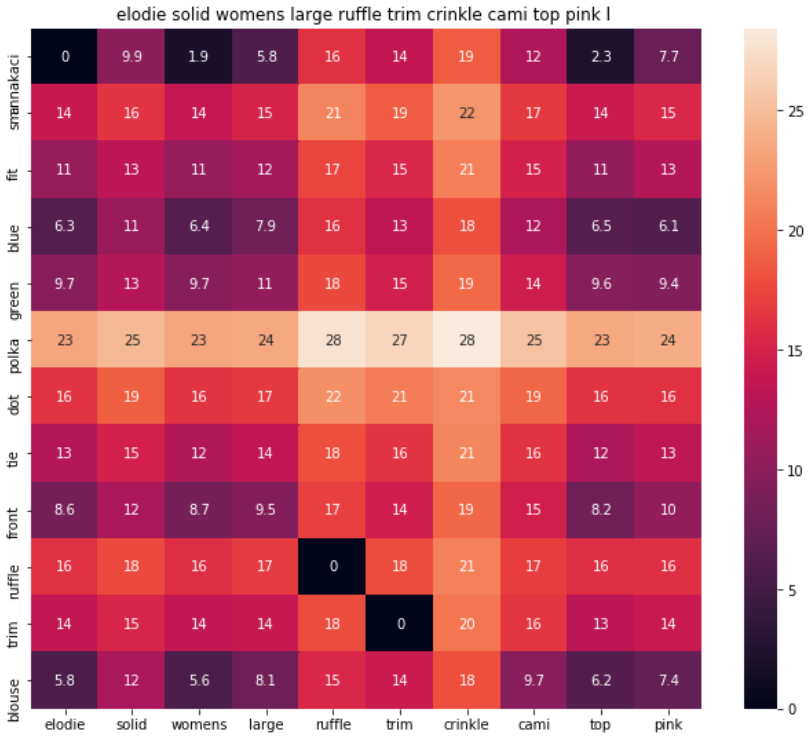
ASIN : B07125NJJ2
Brand : Alfani
euclidean distance from input : 3.44062
==========
==========



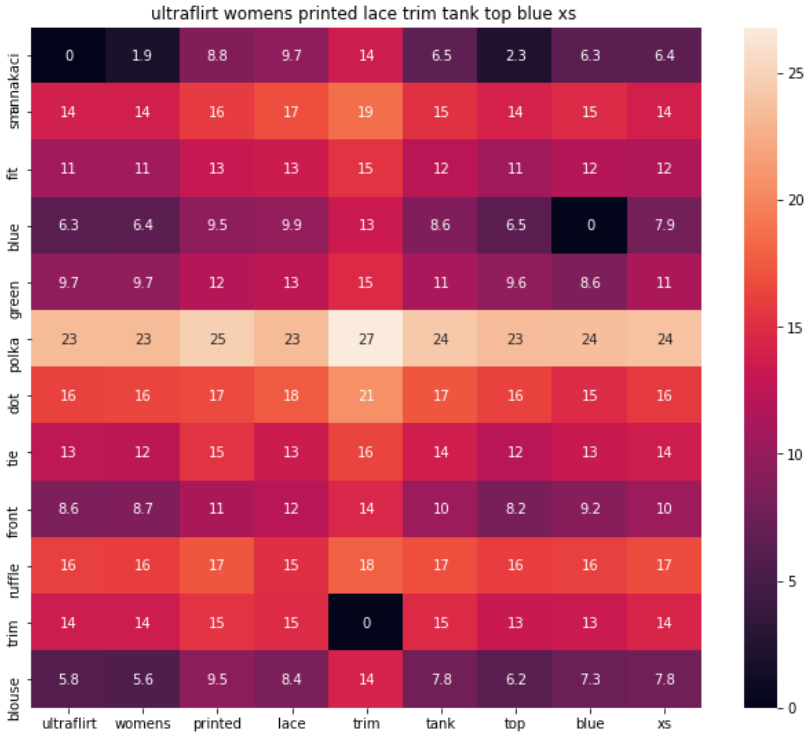
ASIN : B074PFR9CF
Brand : Nili Lotan
euclidean distance from input : 3.5470216
==========
==========



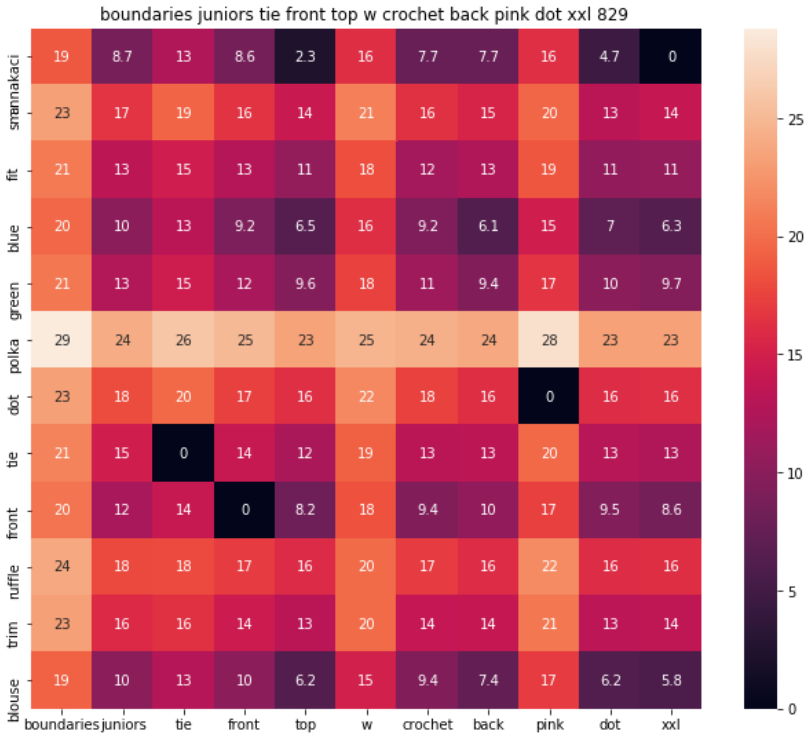
ASIN : B0725BZ69R
Brand : Greylin
euclidean distance from input : 3.5701468
==========
==========



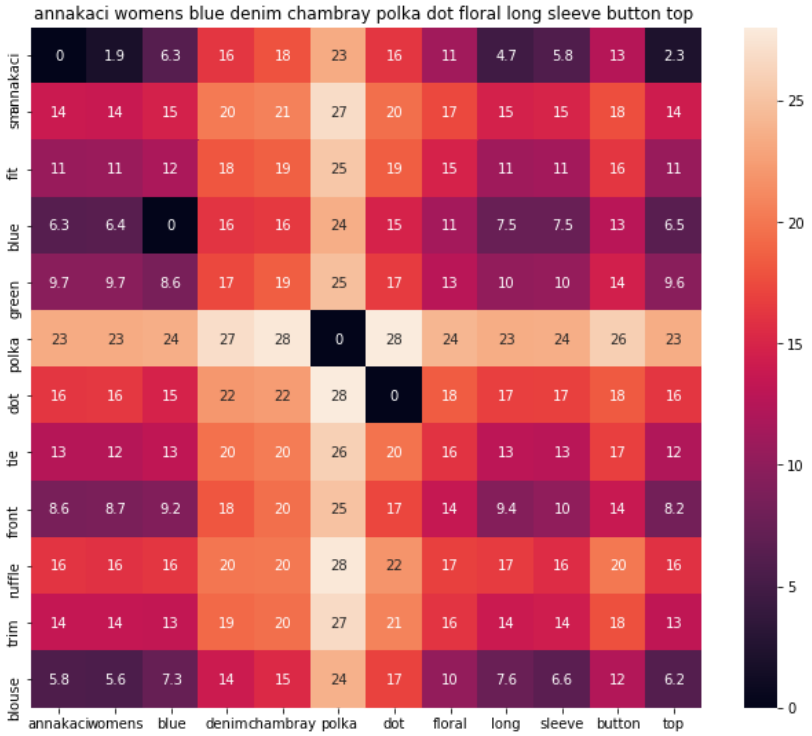
ASIN : B071XHZ2N2
Brand : Elodie
euclidean distance from input : 3.6468353
==========
==========



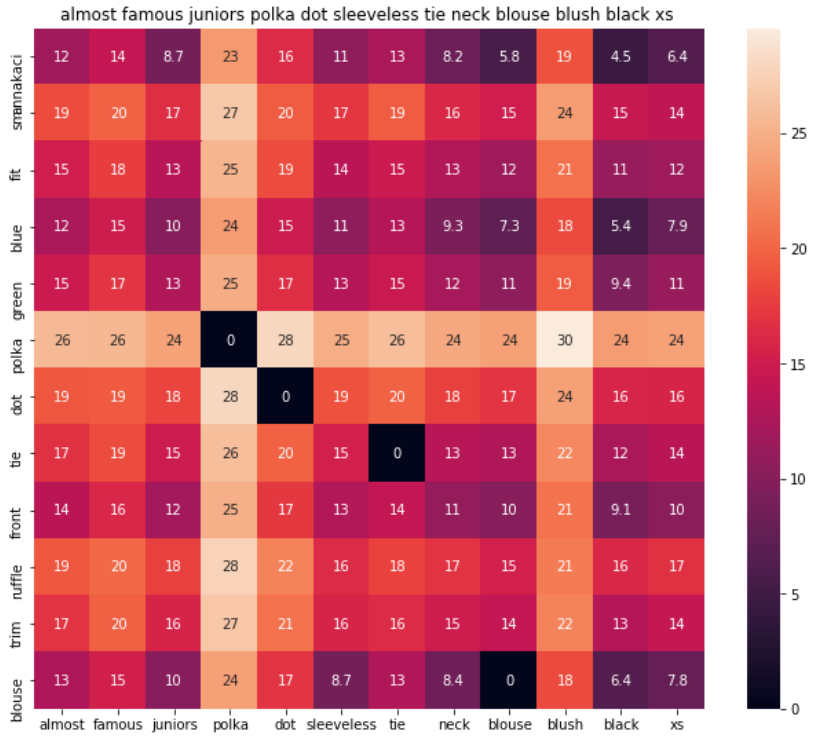
ASIN : B014JS4NIS
Brand : Ultra Flirt
euclidean distance from input : 3.6527064
==========
==========



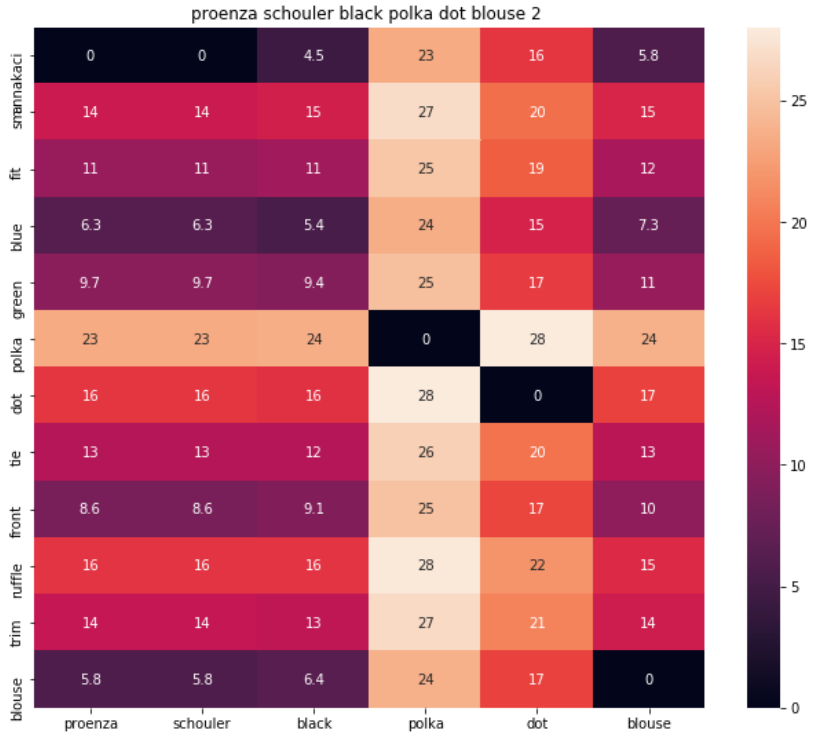
ASIN : B01JCZWOUW
Brand : No Boundaries
euclidean distance from input : 3.657713
==========
==========



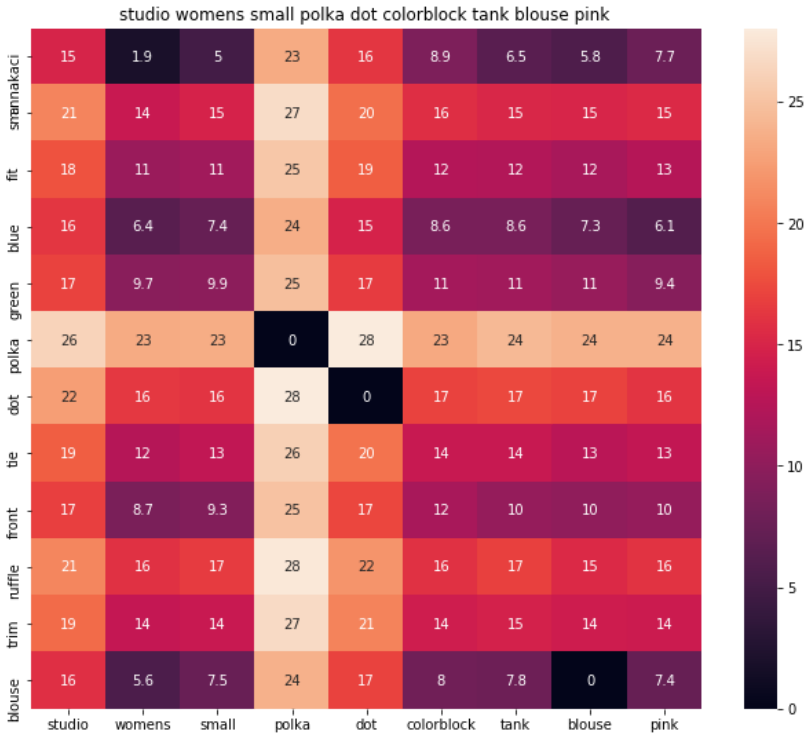
ASIN : B008SMIFN6
Brand : Anna-Kaci
euclidean distance from input : 3.6822639
==========
==========



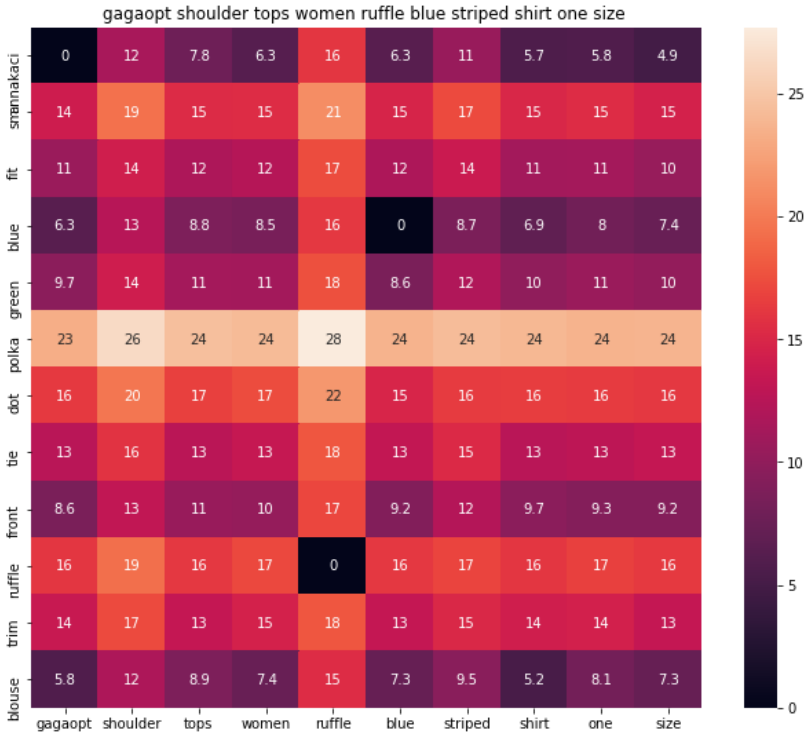
ASIN : B0745J9HNS
Brand : Almost Famous
euclidean distance from input : 3.7160878
==========
==========



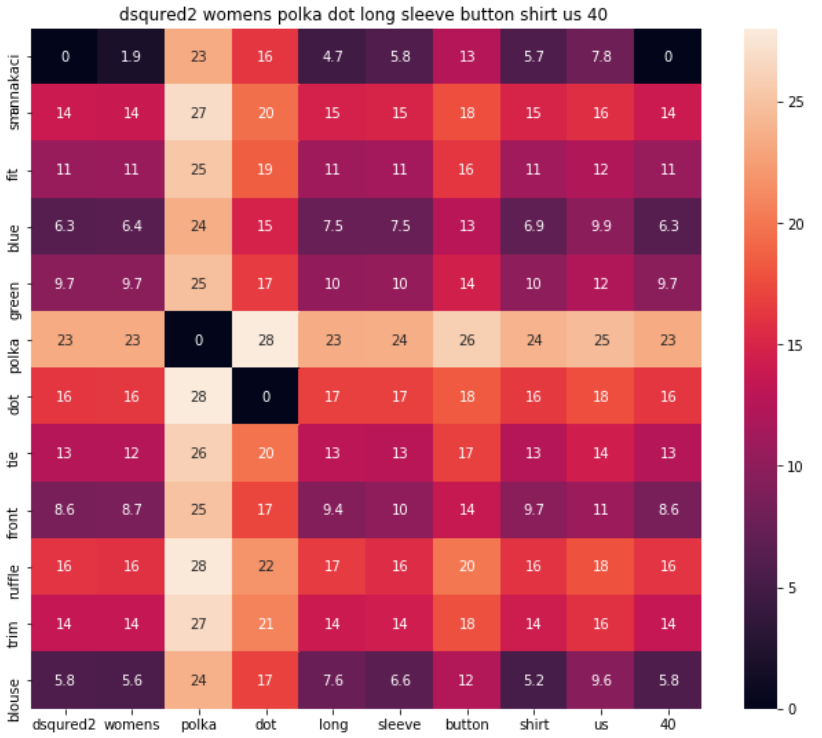
ASIN : B074TLHLMN
Brand : Proenza Schouler
euclidean distance from input : 3.7218063
==========
==========



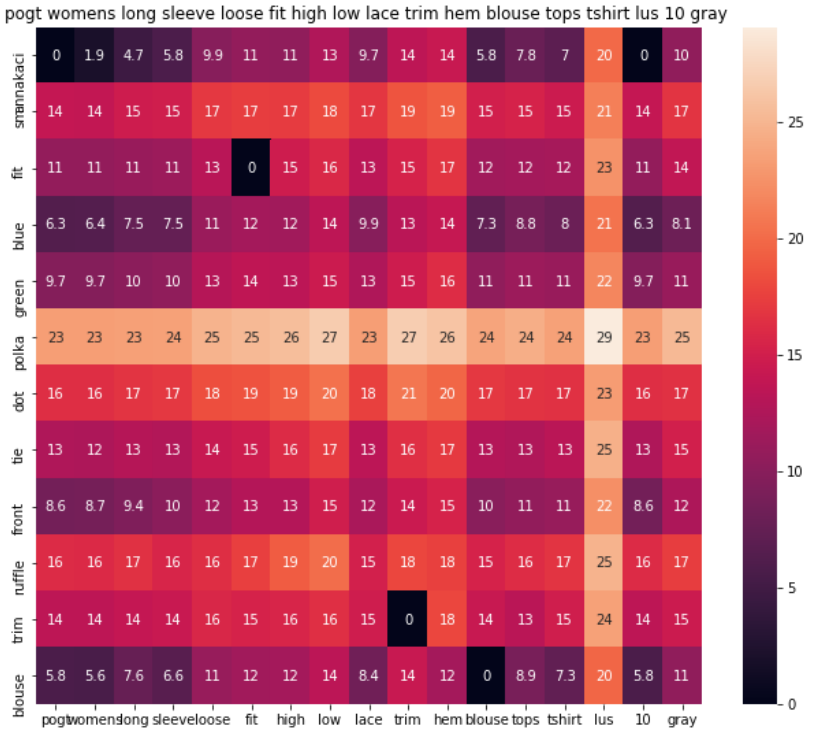
ASIN : B0721KC2HT
Brand : Studio M
euclidean distance from input : 3.7407608
==========
==========



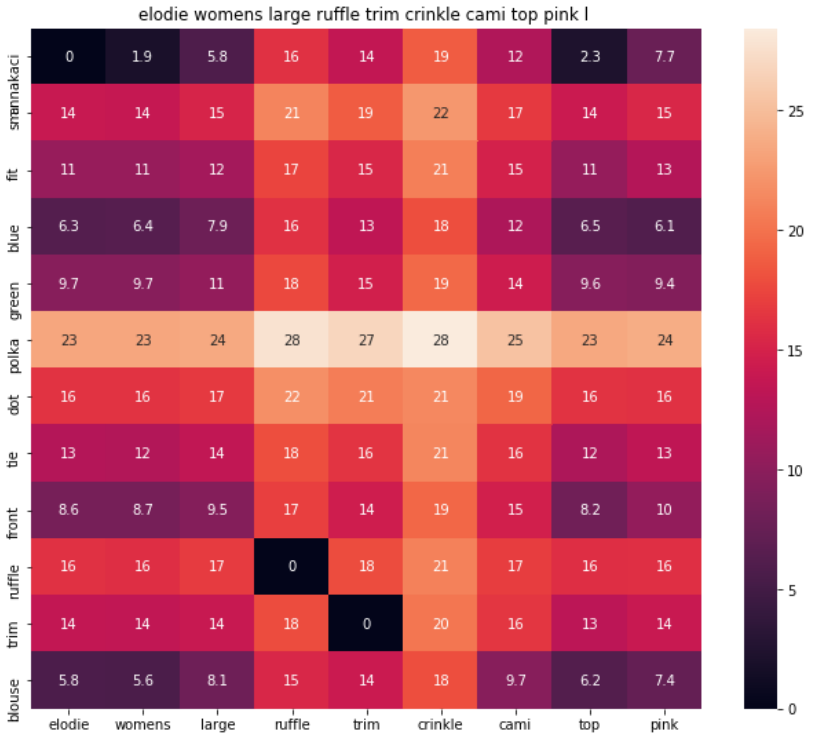
ASIN : B0731KJL5J
Brand : gagaopt
euclidean distance from input : 3.7530386
==========
==========



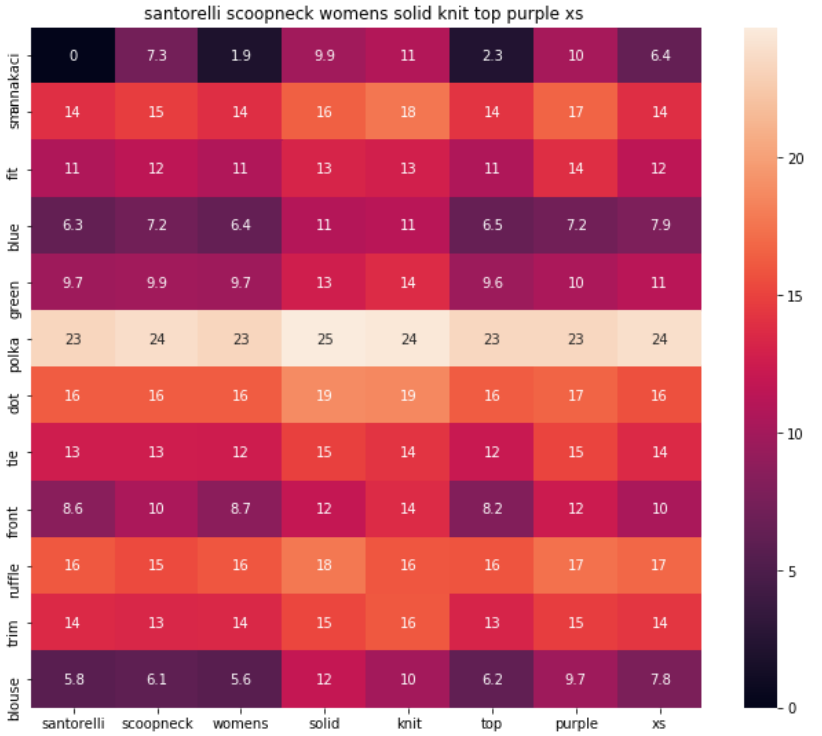
ASIN : B01APYN5AM
Brand : DSQUARED2
euclidean distance from input : 3.7708433
==========
==========



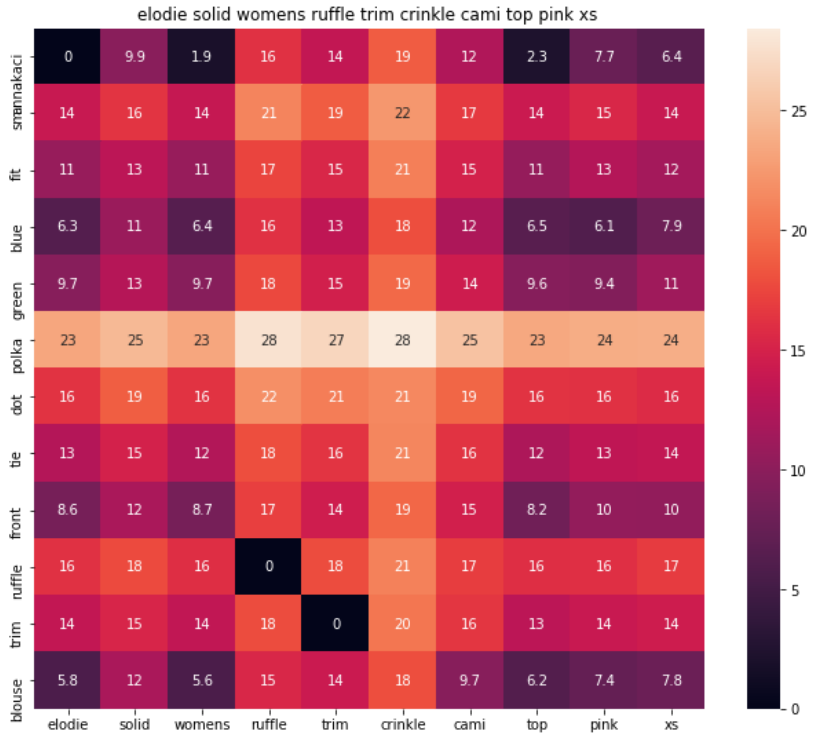
ASIN : B01M19NC0Y
Brand : POGT
euclidean distance from input : 3.7775102
==========
==========



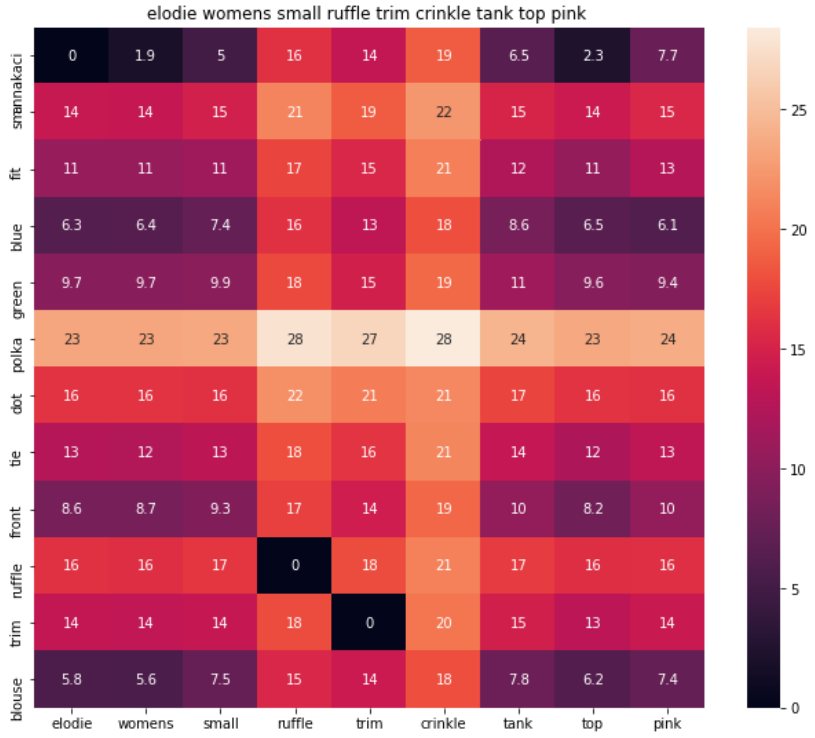
ASIN : B06XD3ZTCB
Brand : Elodie
euclidean distance from input : 3.7784634
==========
==========



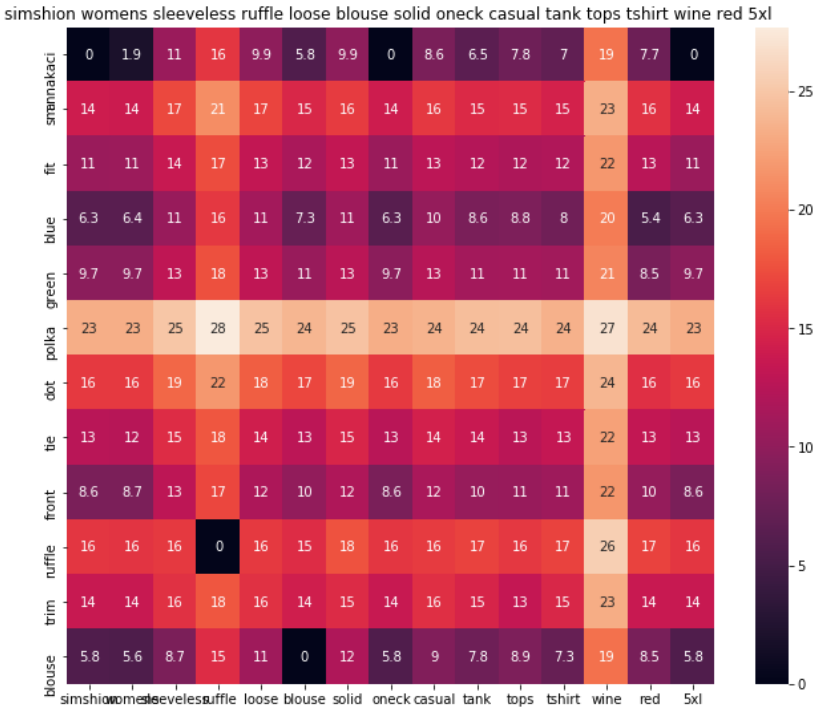
ASIN : B074QW18HZ
Brand : Santorelli
euclidean distance from input : 3.7946742
==========
==========



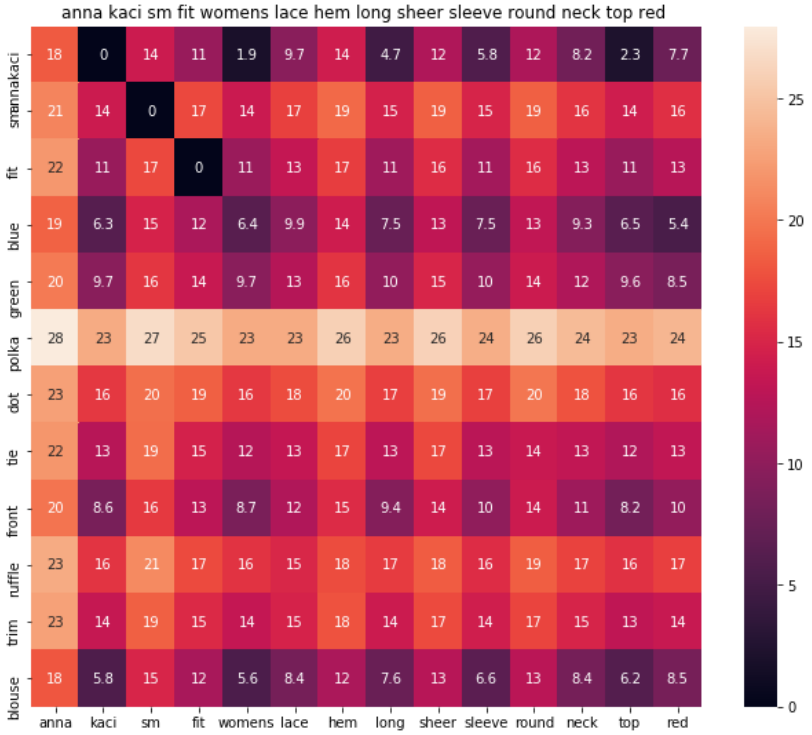
ASIN : B072KRN8ZP
Brand : Elodie
euclidean distance from input : 3.794913
==========
==========



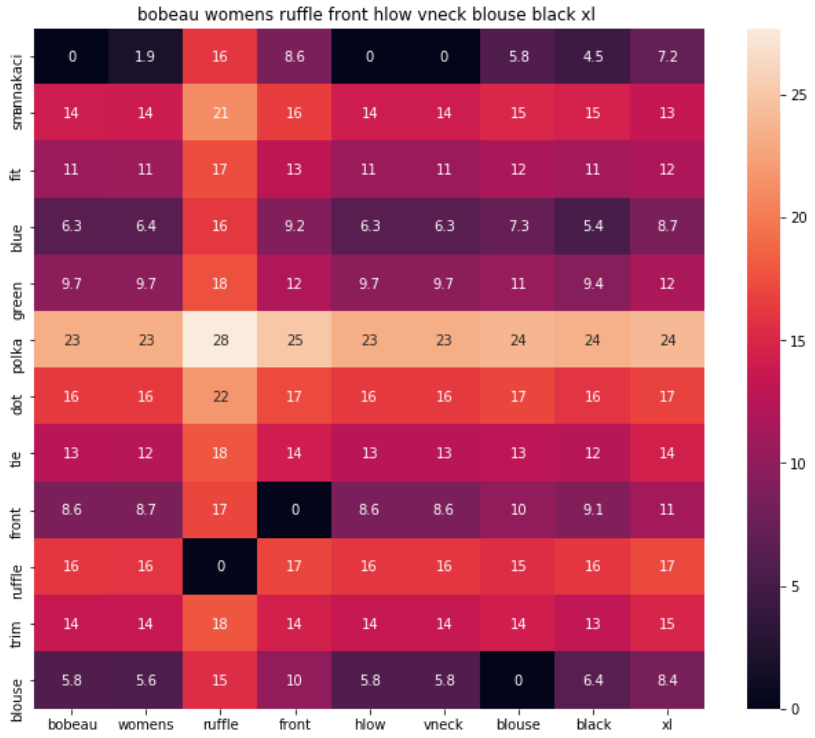
ASIN : B06XCTT55W
Brand : Elodie
euclidean distance from input : 3.7962322
==========
==========



ASIN : B072QS773R
Brand : SIMSHION
euclidean distance from input : 3.8021717
==========
==========



ASIN : B019XS6OLS
Brand : Anna-Kaci
euclidean distance from input : 3.8030992
==========
==========



```
ASIN : B0758J19PY
Brand : Bobeau
euclidean distance from input : 3.8109663
================================================================================
================================================
```

Observation

- As we see that all model are performing well on Text data.
- As we comapire all model Tfidf work well in all the above Text data beacuse most of the time the result related to qurey text.

[9.6] Weighted similarity using brand and color.

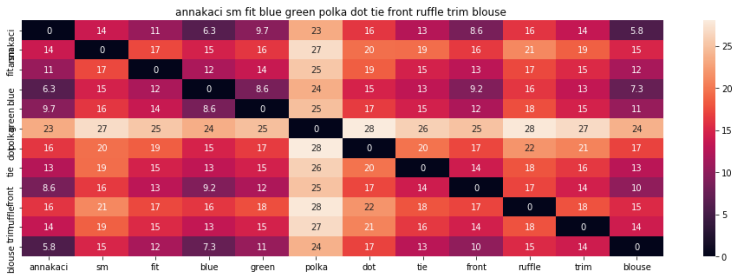
In [35]:
```
# some of the brand values are empty.
# Need to replace Null with string "NULL"
data['brand'].fillna(value="Not given", inplace=True )

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

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

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

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

#extra_features = hstack((brand_features, type_features, color_features)).tocs
r()
extra_features = ((brand_features)).tocsr()
extra_features1 = ((color_features)).tocsr()
extra_features2 = ((type_features)).tocsr()
```

In [27]:
```
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(we
ighted/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(we
ighted/avg) of length 300 corresponds to each word in give title
    s2_vec = get_word_vec(sentance2, doc_id2, model)

    # s1_s2_dist = np.array(#number of words in title1 * #number of words in t
itle2)
    # s1_s2_dist[i,j] = euclidean distance between words i, j
    s1_s2_dist = get_distance(s1_vec, s2_vec)

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

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

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

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

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

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

```
    ax2.grid(False)
    ax2.set_xticks([])
    ax2.set_yticks([])

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

    plt.show()
```

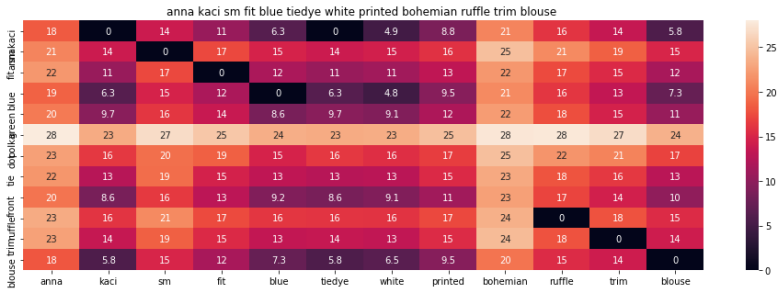
In [28]:
```
def idf_w2v_brand(doc_id, w1, w2, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand and color features

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

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

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


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

idf_w2v_brand(931, 4, 8, 25)
# in the give heat map, each cell contains the euclidean distance between word
s i, j
```

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00KLHUIBS	Anna-Kaci	Blue/Green



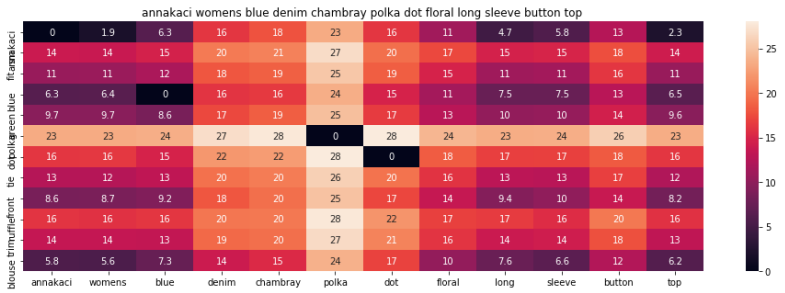
ASIN : B00KLHUIBS
Brand : Anna-Kaci
euclidean distance from input : 0.0
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00YQ8S4K0	Anna-Kaci	Blue



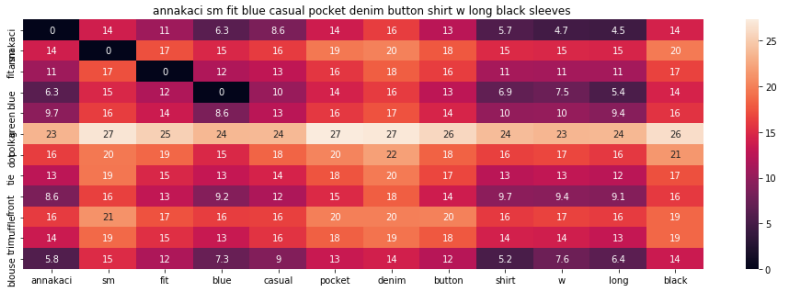
ASIN : B00YQ8S4K0
Brand : Anna-Kaci
euclidean distance from input : 1.8064510027567546
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B008SMIFN6	Anna-Kaci	Blue



ASIN : B008SMIFN6
Brand : Anna-Kaci
euclidean distance from input : 1.8940879503885906
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B0097LQOOY	Anna-Kaci	Blue

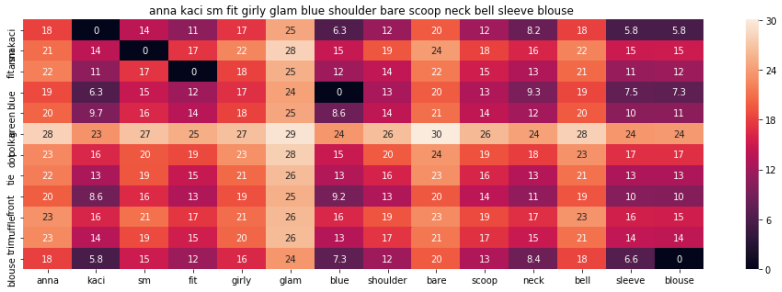


ASIN : B0097LQOOY
Brand : Anna-Kaci
euclidean distance from input : 2.015669345855713
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00726I25U	Anna-Kaci	Blue

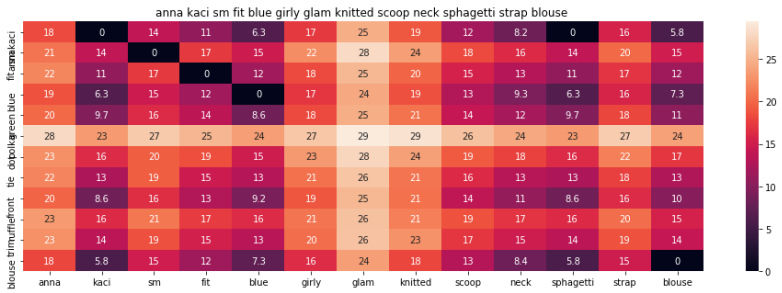
ASIN : B00726I25U
Brand : Anna-Kaci
euclidean distance from input : 2.1188316345214844
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B0128Z19IQ	Anna-Kaci	Blue



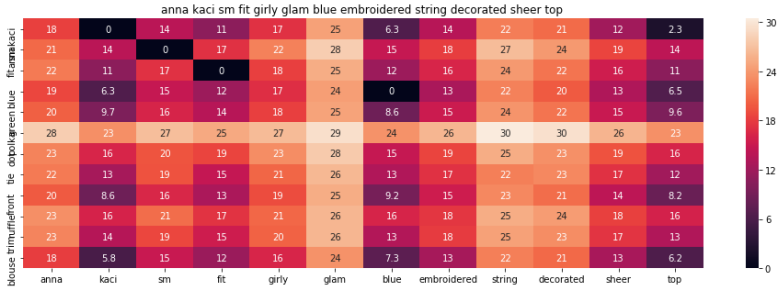
ASIN : B0128Z19IQ
Brand : Anna-Kaci
euclidean distance from input : 2.2515010833740234
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B010EH1PCK	Anna-Kaci	Blue



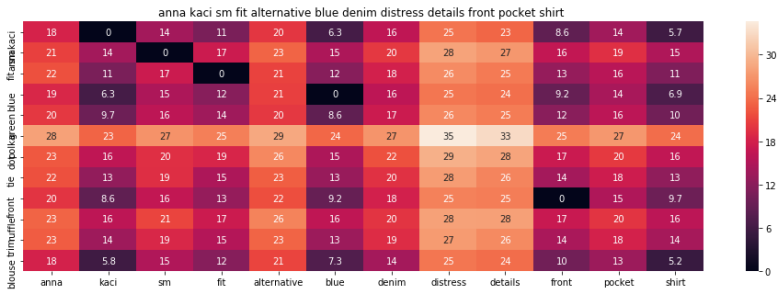
ASIN : B010EH1PCK
Brand : Anna-Kaci
euclidean distance from input : 2.2853997548421225
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B0128Z1BNO	Anna-Kaci	Blue



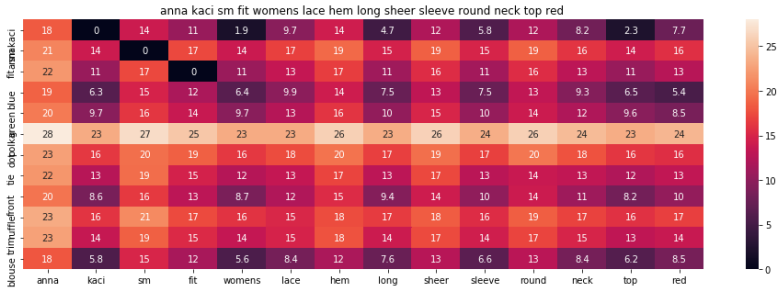
ASIN : B0128Z1BNO
Brand : Anna-Kaci
euclidean distance from input : 2.3356444040934243
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B010EH2WCW	Anna-Kaci	Blue



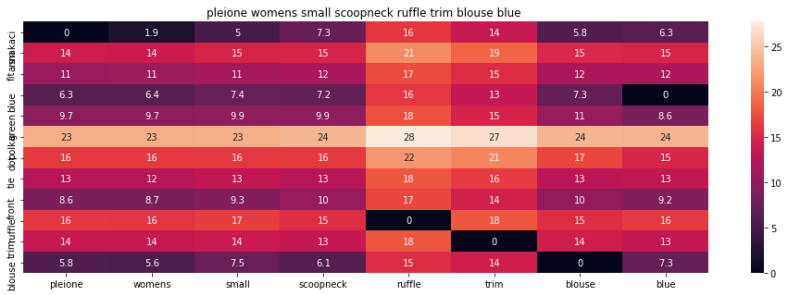
ASIN : B010EH2WCW
Brand : Anna-Kaci
euclidean distance from input : 2.340008099873861
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B019XS6OLS	Anna-Kaci	Red



ASIN : B019XS6OLS
Brand : Anna-Kaci
euclidean distance from input : 2.4224002568545933
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B072VHTT1D	Pleione	Blue

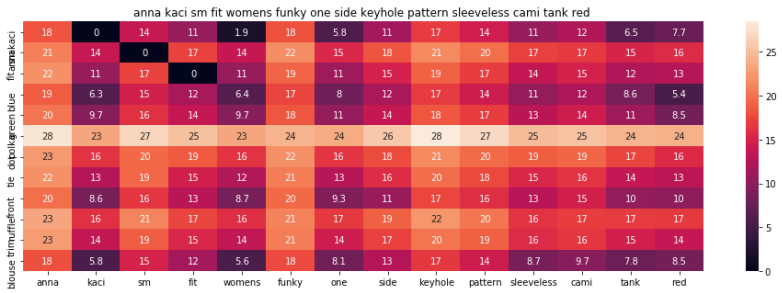


ASIN : B072VHTT1D
Brand : Pleione
euclidean distance from input : 2.434874693552653
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B010EH3S02	Anna-Kaci	White

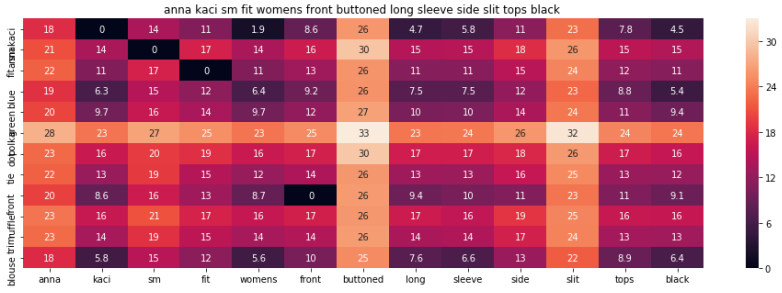
ASIN : B010EH3S02
Brand : Anna-Kaci
euclidean distance from input : 2.4536291965467725
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B019NQLRF8	Anna-Kaci	Red



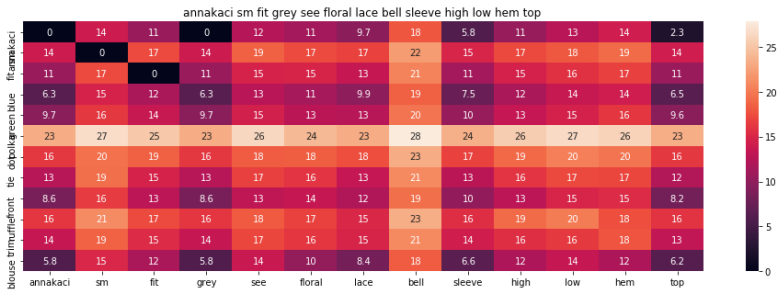
ASIN : B019NQLRF8
Brand : Anna-Kaci
euclidean distance from input : 2.4759787130974726
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B019820A4Q	Anna-Kaci	Black



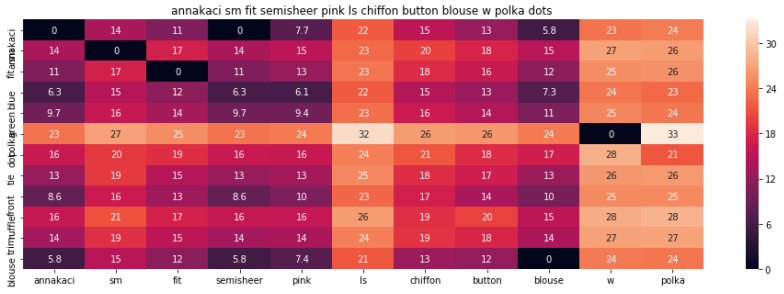
ASIN : B019820A4Q
Brand : Anna-Kaci
euclidean distance from input : 2.4928237009667353
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00NAEMMW2	Anna-Kaci	Grey



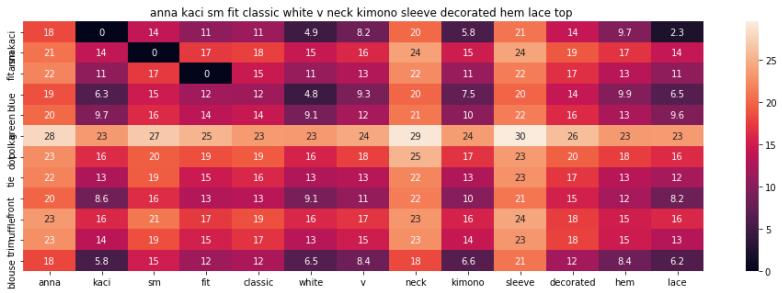
ASIN : B00NAEMMW2
Brand : Anna-Kaci
euclidean distance from input : 2.497477313739359
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B008Z5ST3C	Anna-Kaci	Pink



ASIN : B008Z5ST3C
Brand : Anna-Kaci
euclidean distance from input : 2.5052251228315625
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B011WFBMBC	Anna-Kaci	White

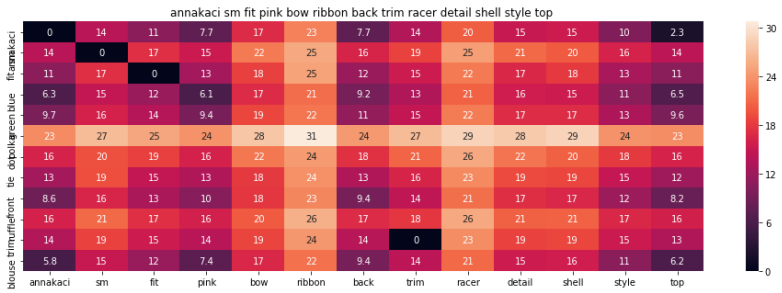


ASIN : B011WFBMBC
Brand : Anna-Kaci
euclidean distance from input : 2.5233897575361524
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B007HCB8XM	Anna-Kaci	White

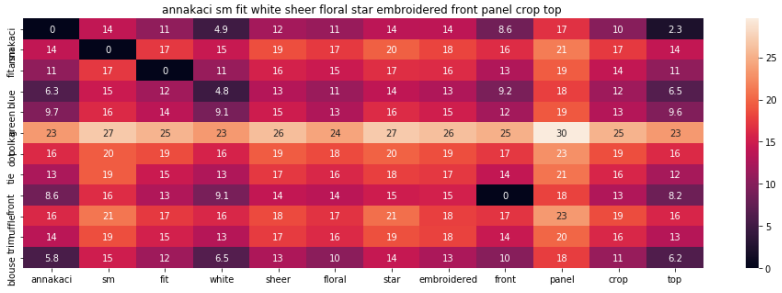
ASIN : B007HCB8XM
Brand : Anna-Kaci
euclidean distance from input : 2.5261344322187695
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00KOBQEBO	Anna-Kaci	Pink



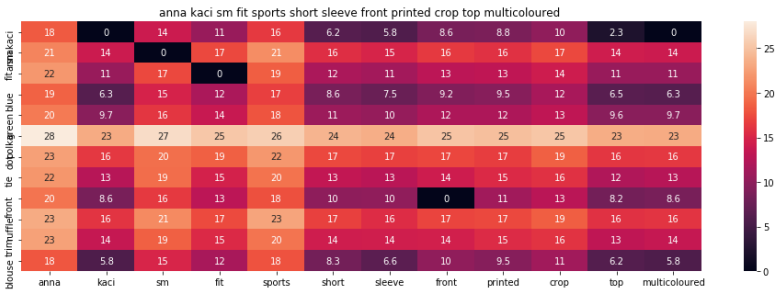
ASIN : B00KOBQEBO
Brand : Anna-Kaci
euclidean distance from input : 2.537382543943623
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00NAB1R8A	Anna-Kaci	White



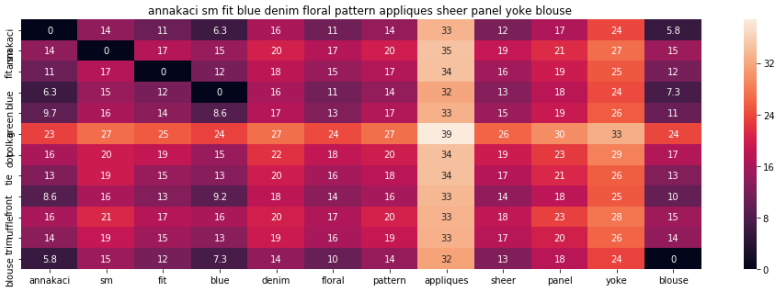
ASIN : B00NAB1R8A
Brand : Anna-Kaci
euclidean distance from input : 2.542425732674499
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B018J051YC	Anna-Kaci	Multicoloured



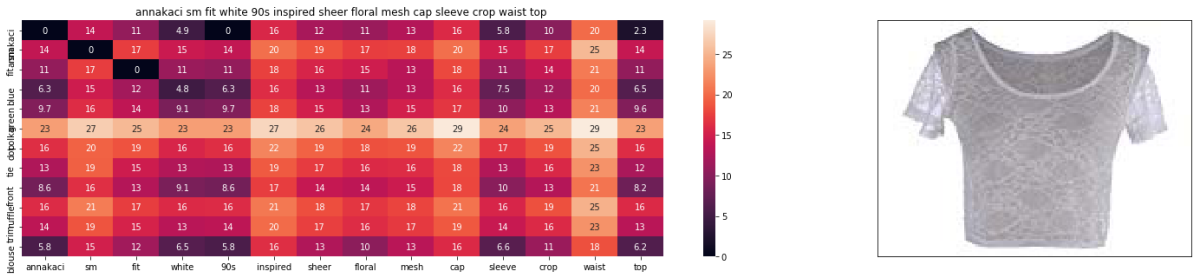
ASIN : B018J051YC
Brand : Anna-Kaci
euclidean distance from input : 2.5553300587955112
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00E7Z8G8M	Anna-Kaci	Blue



ASIN : B00E7Z8G8M
Brand : Anna-Kaci
euclidean distance from input : 2.5572856267293296
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00JJ1TG42	Anna-Kaci	White



ASIN : B00JJ1TG42
Brand : Anna-Kaci
euclidean distance from input : 2.563126187386413
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B008LE9GA2	Anna-Kaci	Multicolor

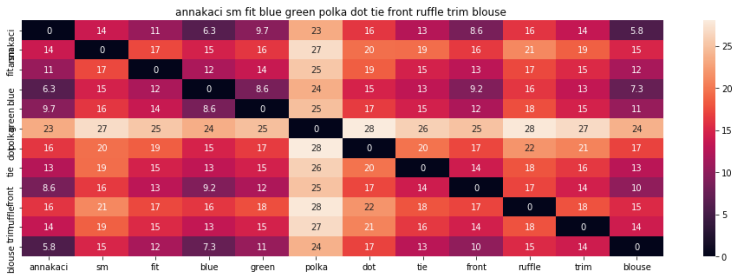
ASIN : B008LE9GA2
Brand : Anna-Kaci
euclidean distance from input : 2.577172697447041
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B0731KJL5J	gagaopt	Blue

ASIN : B0731KJL5J
Brand : gagaopt
euclidean distance from input : 2.5843462149302163
==========
==========

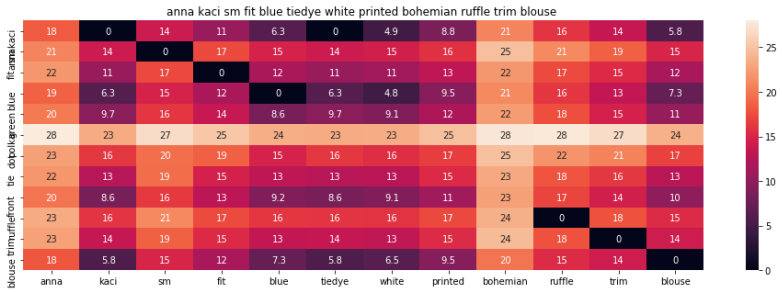
In [29]:
```
# brand and color weight =50
# title vector weight = 5

idf_w2v_brand(931, 5, 50, 25)
```

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00KLHUIBS	Anna-Kaci	Blue/Green



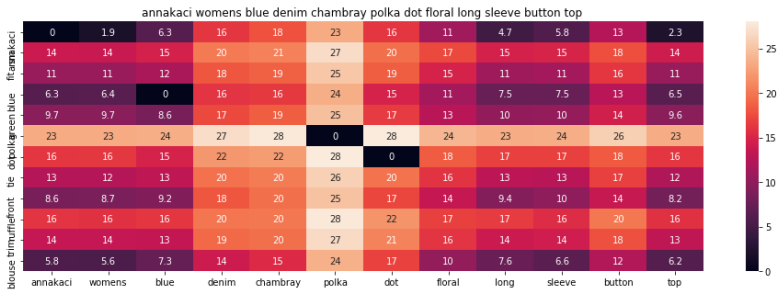
ASIN : B00KLHUIBS
Brand : Anna-Kaci
euclidean distance from input : 0.0
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00YQ8S4K0	Anna-Kaci	Blue



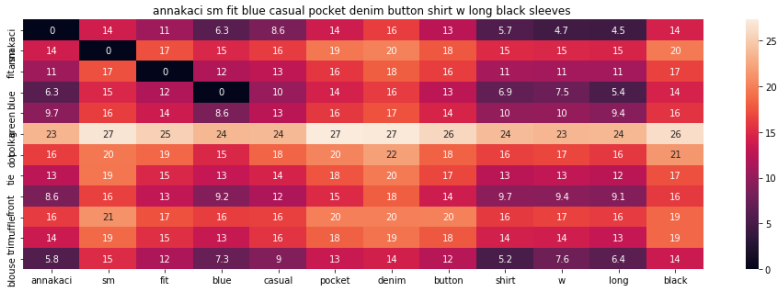
ASIN : B00YQ8S4K0
Brand : Anna-Kaci
euclidean distance from input : 1.2199411912397904
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B008SMIFN6	Anna-Kaci	Blue



ASIN : B008SMIFN6
Brand : Anna-Kaci
euclidean distance from input : 1.2438421769575638
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B0097LQOOY	Anna-Kaci	Blue

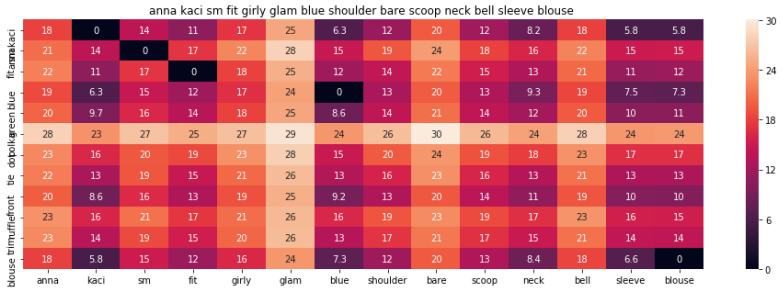


ASIN : B0097LQOOY
Brand : Anna-Kaci
euclidean distance from input : 1.2770007393576883
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00726I25U	Anna-Kaci	Blue

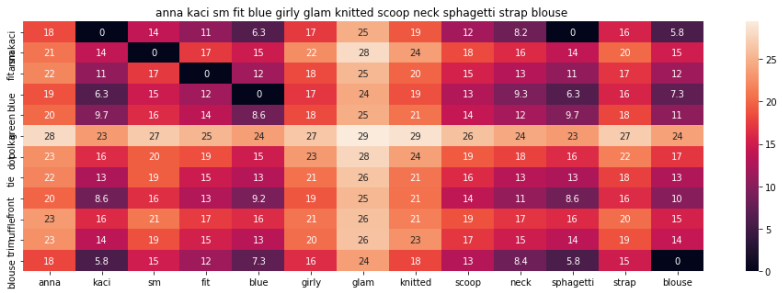
ASIN : B00726I25U
Brand : Anna-Kaci
euclidean distance from input : 1.3051359003240413
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B0128Z19IQ	Anna-Kaci	Blue



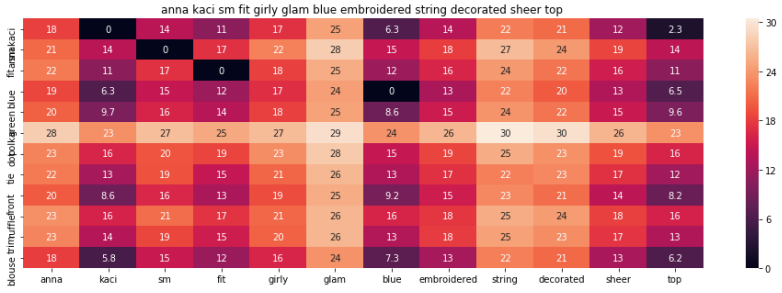
ASIN : B0128Z19IQ
Brand : Anna-Kaci
euclidean distance from input : 1.3413184772838245
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B010EH1PCK	Anna-Kaci	Blue



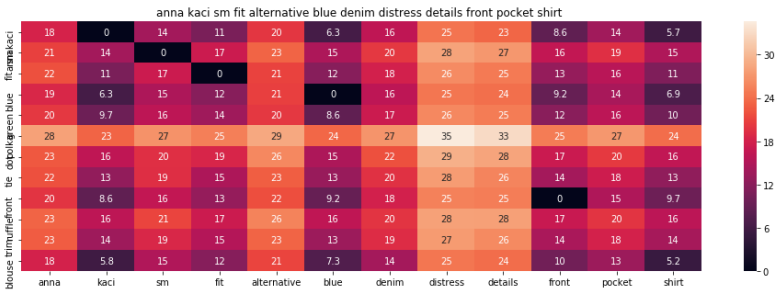
ASIN : B010EH1PCK
Brand : Anna-Kaci
euclidean distance from input : 1.350563569502397
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B0128Z1BNO	Anna-Kaci	Blue



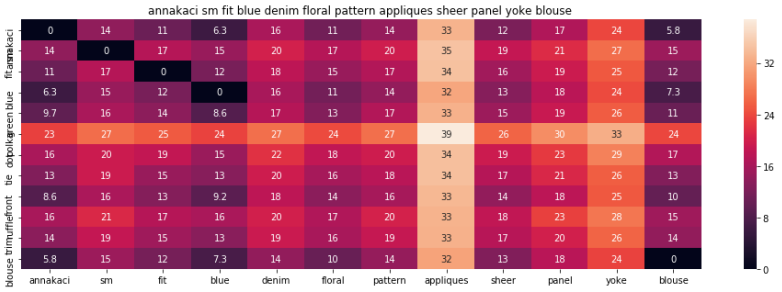
ASIN : B0128Z1BNO
Brand : Anna-Kaci
euclidean distance from input : 1.364266655661843
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B010EH2WCW	Anna-Kaci	Blue



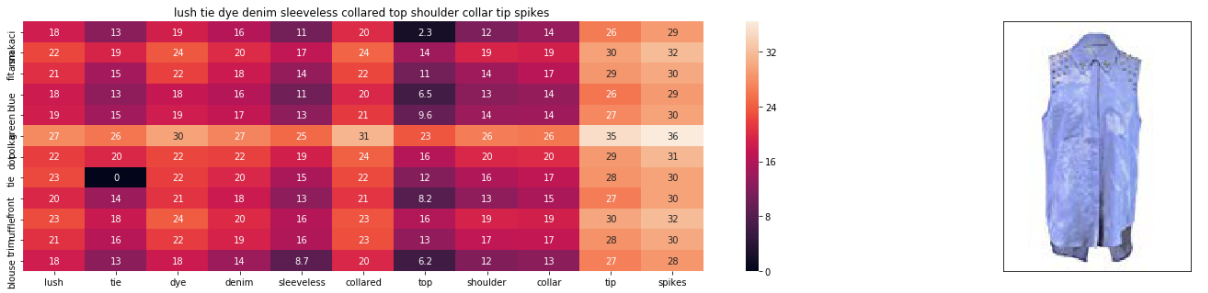
ASIN : B010EH2WCW
Brand : Anna-Kaci
euclidean distance from input : 1.3654567371715198
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00E7Z8G8M	Anna-Kaci	Blue



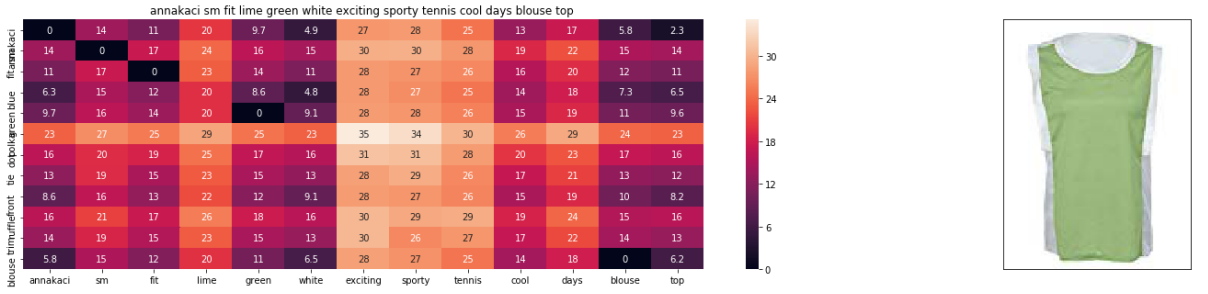
ASIN : B00E7Z8G8M
Brand : Anna-Kaci
euclidean distance from input : 1.4247142444957386
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00FH9CBE2	Anna-Kaci	Blue



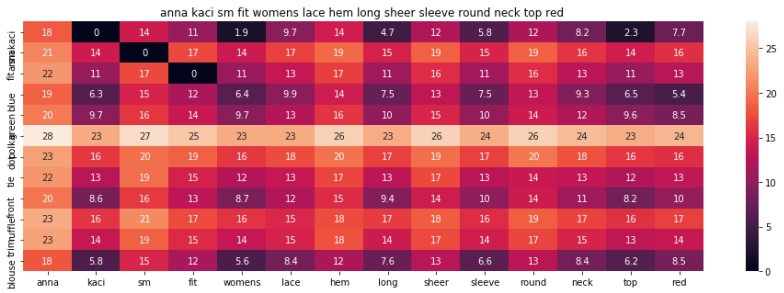
ASIN : B00FH9CBE2
Brand : Anna-Kaci
euclidean distance from input : 1.5177067843350498
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B007Y9SZSY	Anna-Kaci	Green/White



ASIN : B007Y9SZSY
Brand : Anna-Kaci
euclidean distance from input : 1.7635778603804593
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B019XS6OLS	Anna-Kaci	Red

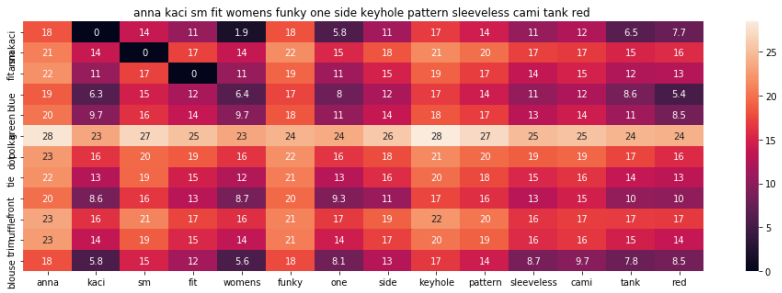


ASIN : B019XS6OLS
Brand : Anna-Kaci
euclidean distance from input : 1.9203279214315787
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B010EH3S02	Anna-Kaci	White

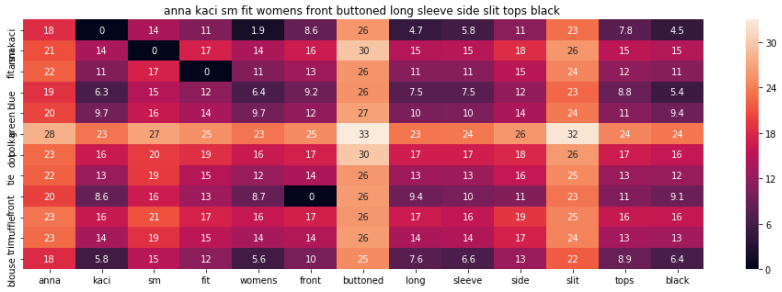
ASIN : B010EH3S02
Brand : Anna-Kaci
euclidean distance from input : 1.9288449266584076
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B019NQLRF8	Anna-Kaci	Red



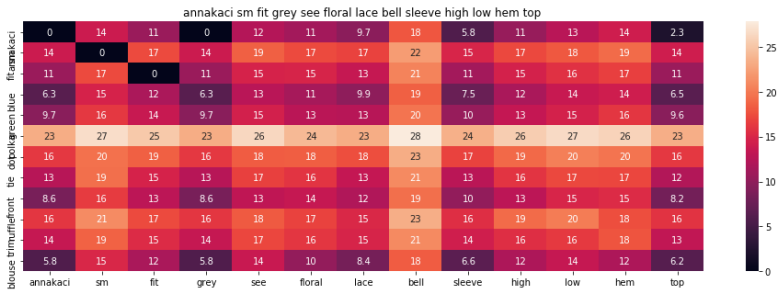
ASIN : B019NQLRF8
Brand : Anna-Kaci
euclidean distance from input : 1.9349402233447535
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B019820A4Q	Anna-Kaci	Black



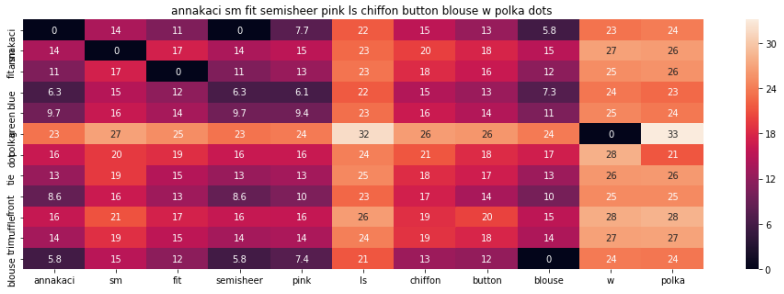
ASIN : B019820A4Q
Brand : Anna-Kaci
euclidean distance from input : 1.9395343326198777
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00NAEMMW2	Anna-Kaci	Grey



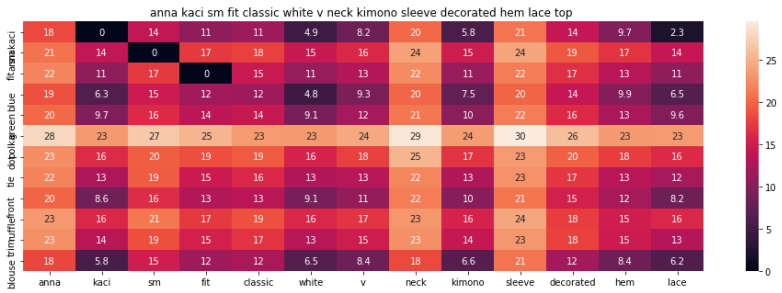
ASIN : B00NAEMMW2
Brand : Anna-Kaci
euclidean distance from input : 1.9408034824001512
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B008Z5ST3C	Anna-Kaci	Pink



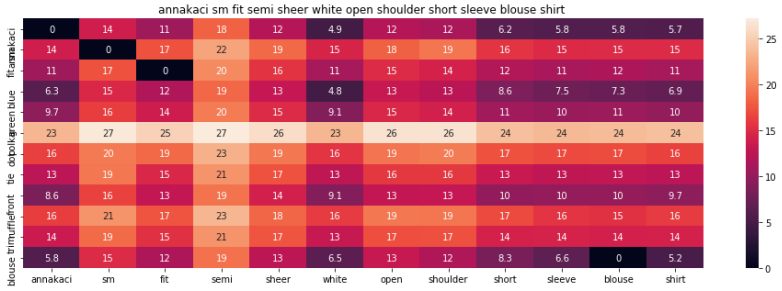
ASIN : B008Z5ST3C
Brand : Anna-Kaci
euclidean distance from input : 1.942916547252779
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B011WFBMBC	Anna-Kaci	White



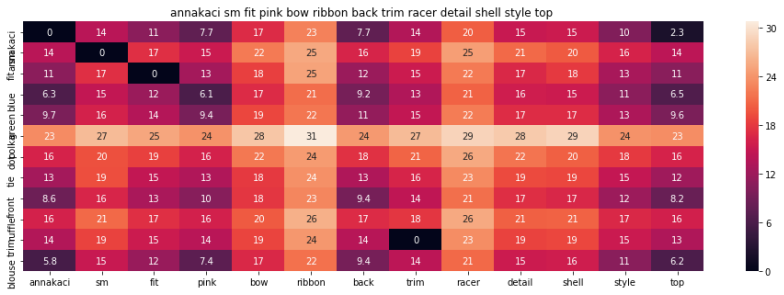
ASIN : B011WFBMBC
Brand : Anna-Kaci
euclidean distance from input : 1.9478705211963159
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B007HCB8XM	Anna-Kaci	White



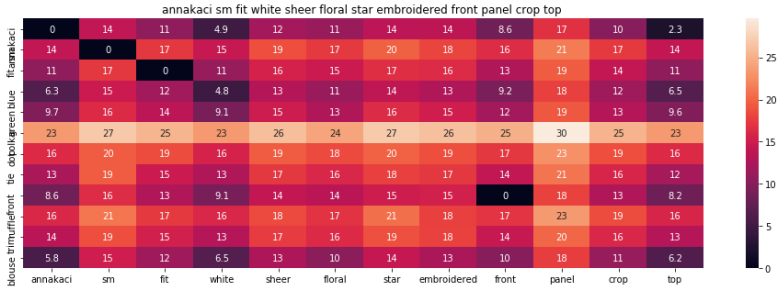
ASIN : B007HCB8XM
Brand : Anna-Kaci
euclidean distance from input : 1.9486190688370297
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00KOBQEBO	Anna-Kaci	Pink



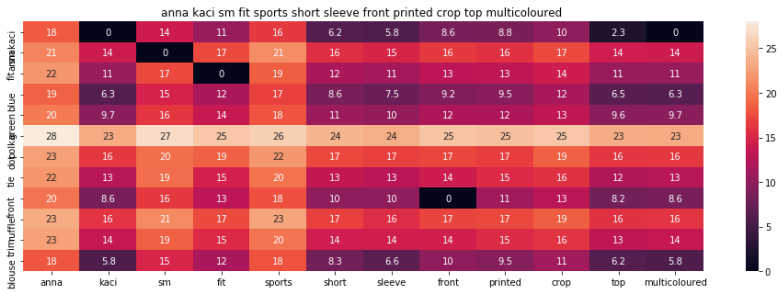
ASIN : B00KOBQEBO
Brand : Anna-Kaci
euclidean distance from input : 1.9516867443408472
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00NAB1R8A	Anna-Kaci	White



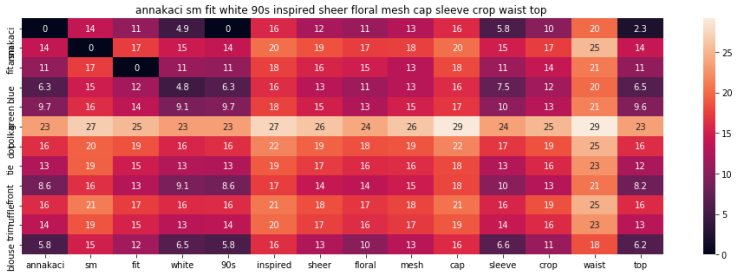
ASIN : B00NAB1R8A
Brand : Anna-Kaci
euclidean distance from input : 1.9530621507795014
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B018J051YC	Anna-Kaci	Multicoloured



ASIN : B018J051YC
Brand : Anna-Kaci
euclidean distance from input : 1.9565815211186348
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B00JJ1TG42	Anna-Kaci	White



ASIN : B00JJ1TG42
Brand : Anna-Kaci
euclidean distance from input : 1.958707729337296
==========
==========

Asin	Brand	Color
B00KLHUIBS	Anna-Kaci	Blue/Green
B008LE9GA2	Anna-Kaci	Multicolor


```
ASIN : B008LE9GA2
Brand : Anna-Kaci
euclidean distance from input : 1.962538587047701
================================================================================
================================================
```

observation

- As we see that Brand and colour has good result.

[10.2] Keras and Tensorflow to extract features

In [30]:
```
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
from keras import applications
from sklearn.metrics import pairwise_distances
import matplotlib.pyplot as plt
import requests
from PIL import Image
import pandas as pd
import pickle
```

```
Using TensorFlow backend.
```

In [31]:
```
# https://gist.github.com/fchollet/f35fbc80e066a49d65f1688a7e99f069
# Code reference: https://blog.keras.io/building-powerful-image-classification
-models-using-very-little-data.html



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

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

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

# each image is converted into 25088 length dense-vector


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

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


def save_bottlebeck_features():

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

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

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

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

    bottleneck_features_train = model.predict_generator(generator, nb_train_sa
mples // batch_size)
    bottleneck_features_train = bottleneck_features_train.reshape((16042,2508
8))

    np.save(open('16k_data_cnn_features.npy', 'wb'), bottleneck_features_trai
n)
```

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


save_bottlebeck_features()

'''
```

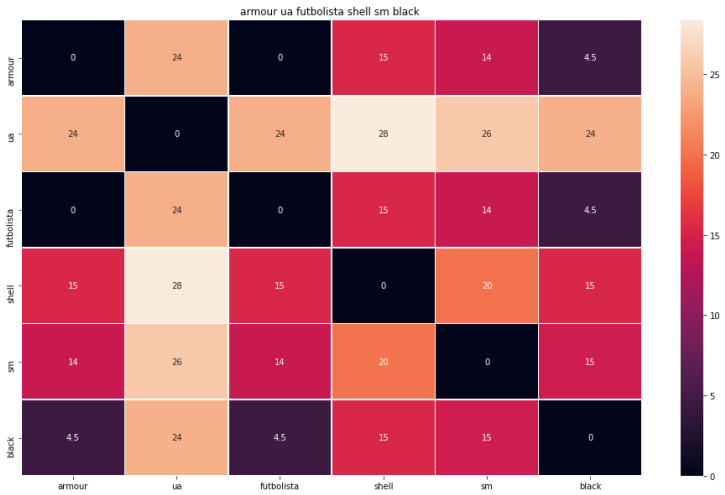
Out[31]:
```
"\n# dimensions of our images.\nimg_width, img_height = 224, 224\n\ntop_model
_weights_path = 'bottleneck_fc_model.h5'\ntrain_data_dir = 'images2/'\nnb_tra
in_samples = 16042\nepochs = 50\nbatch_size = 1\n\n\ndef save_bottlebeck_feat
ures():\n    \n    #Function to compute VGG-16 CNN for image feature extracti
on.\n    \n    asins = []\n    datagen = ImageDataGenerator(rescale=1. / 255)
\n    \n    # build the VGG16 network\n    model = applications.VGG16(include
_top=False, weights='imagenet')\n    generator = datagen.flow_from_directory
(\n        train_data_dir,\n        target_size=(img_width, img_height),\n
batch_size=batch_size,\n        class_mode=None,\n        shuffle=False)\n\n
for i in generator.filenames:\n        asins.append(i[2:-5])\n\n    bottlenec
k_features_train = model.predict_generator(generator, nb_train_samples // bat
ch_size)\n    bottleneck_features_train = bottleneck_features_train.reshape
((16042,25088))\n    \n    np.save(open('16k_data_cnn_features.npy', 'wb'), b
ottleneck_features_train)\n    np.save(open('16k_data_cnn_feature_asins.npy',
'wb'), np.array(asins))\n    \n\nsave_bottlebeck_features()\n\n"
```

[10.3] Visual features based product similarity.

In [41]:
```
#load the features and corresponding ASINS info.
bottleneck_features_train = np.load('16k_data_cnn_features.npy')
asins = np.load('16k_data_cnn_feature_asins.npy')
asins = list(asins)

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


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

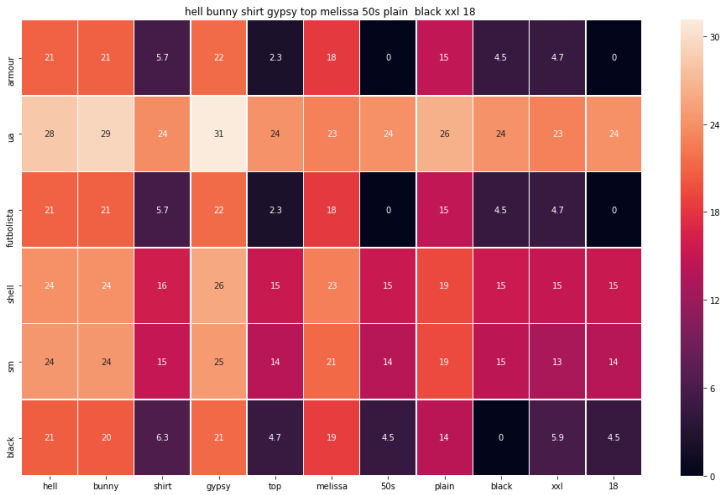
In [47]:
```
def get_similar_products_cnn(doc_id, w1, w2, w3, w4, w5, num_results):
    doc_id = asins.index(df_asins[doc_id])
    title = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].resh
ape(1,-1))
    brand  = pairwise_distances(extra_features, extra_features[doc_id])
    color = pairwise_distances(extra_features1, extra_features1[doc_id])
    image = pairwise_distances(bottleneck_features_train, bottleneck_features_
train[doc_id].reshape(1,-1))
    types = pairwise_distances(extra_features2, extra_features2[doc_id])
    pairwise_dist = (w1 * title[0:16042] +  w2 * brand[0:16042] + w3 * image +
w4*color[0:16042] + w5 * types[0:16042])/float(w1 + w2 + w3 + w4 + w5)


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

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

    for i in range(0, len(indices)):
        heat_map_w2v_brand(data['title'].loc[df_indices[0]],data['title'].loc[
df_indices[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indic
es[i],df_indices[0], df_indices[i], 'weighted')
        print('ASIN of the product :',data['asin'].loc[df_indices[i]])
        print('Brand of the product :',data['brand'].loc[df_indices[i]])
        print('Title of the product :',data['title'].loc[df_indices[i]])
        print('Type of the product :',data['product_type_name'].loc[df_indices
[i]])
        print('Euclidean distance from input :', pdists[i])
        print('='*60)
get_similar_products_cnn(931, 50, 50, 100, 100, 50,  25)
```

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black



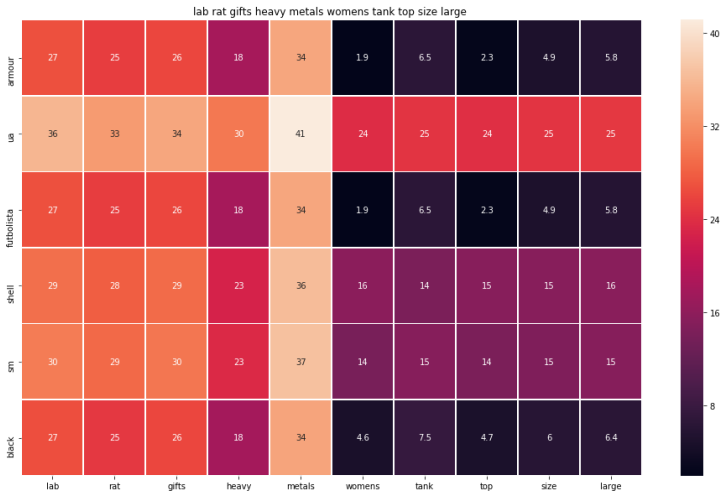
ASIN of the product : B0185VYVR8
Brand of the product : UNDER ARMOUR > UA Tops
Title of the product : armour ua futbolista shell sm black
Type of the product : SPORTING_GOODS
Euclidean distance from input : 0.019964892523629325
==========

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01ANZH0S6	Ripleys-Clothing	Black



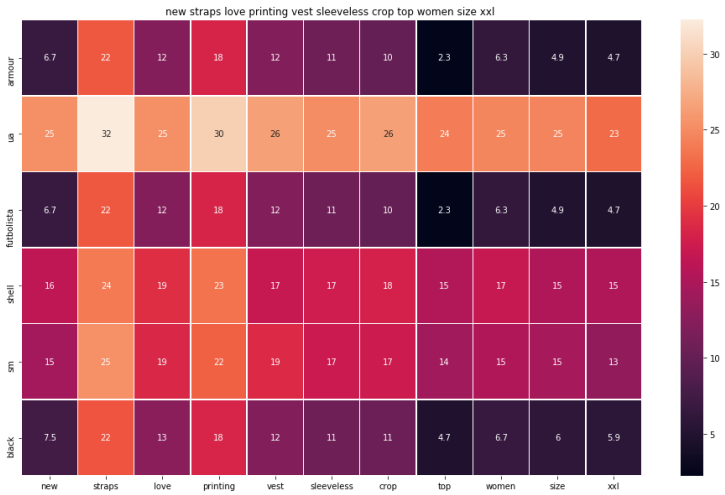
ASIN of the product : B01ANZH0S6
Brand of the product : Ripleys Clothing
Title of the product : hell bunny shirt gypsy top melissa 50s plain black xx
l 18
Type of the product : SHIRT
Euclidean distance from input : 15.289102128833708
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01N3PVWQN	Lab-Rat-Gifts	Black



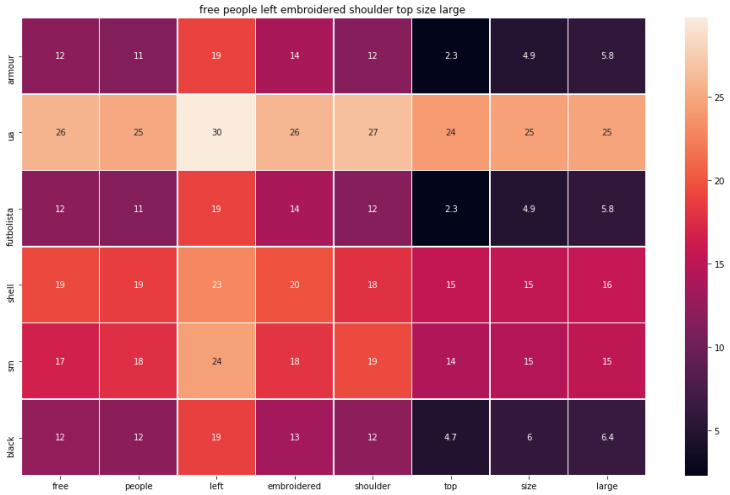
ASIN of the product : B01N3PVWQN
Brand of the product : Lab Rat Gifts
Title of the product : lab rat gifts heavy metals womens tank top size large
Type of the product : SHIRT
Euclidean distance from input : 15.31512673066409
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B00KDW8B1A	namnoishop-Crop-Tops	Black



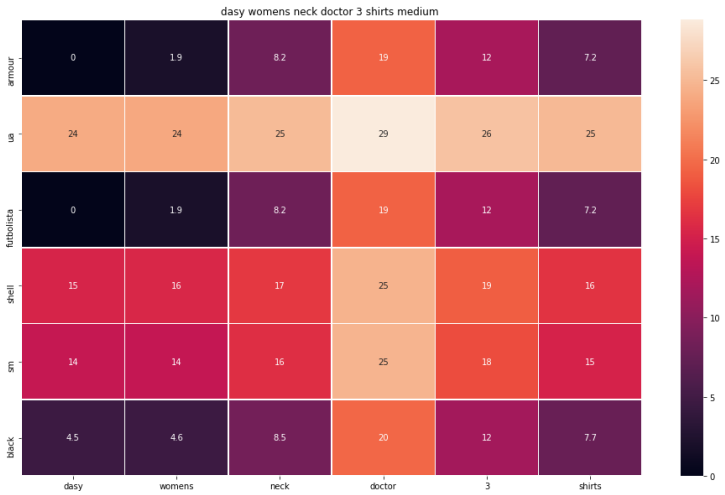
ASIN of the product : B00KDW8B1A
Brand of the product : namnoishop Crop Tops
Title of the product : new straps love printing vest sleeveless crop top wome
n size xxl
Type of the product : SPORTING_GOODS
Euclidean distance from input : 15.354112005993274
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01C7UUMLC	Free-People	Black



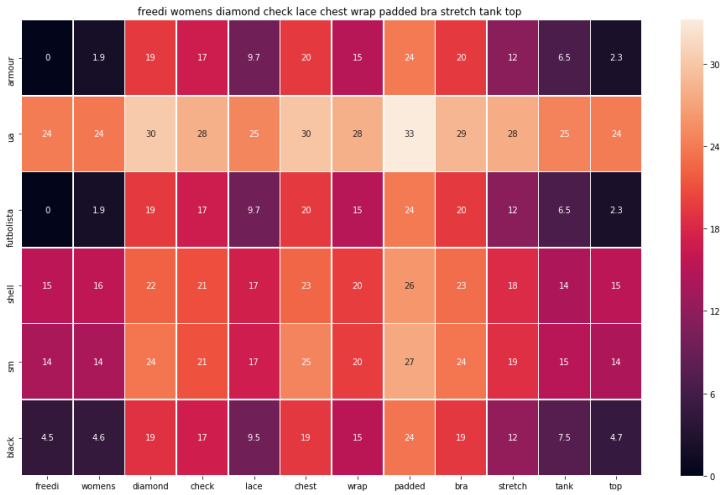
ASIN of the product : B01C7UUMLC
Brand of the product : Free People
Title of the product : free people left embroidered shoulder top size large
Type of the product : SHIRT
Euclidean distance from input : 15.502457405758932
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01588N7VC	Dasy	Black



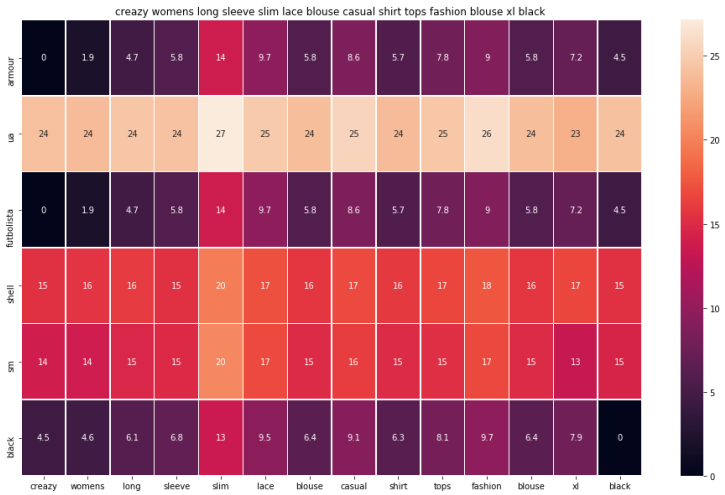
ASIN of the product : B01588N7VC
Brand of the product : Dasy
Title of the product : dasy womens neck doctor 3 shirts medium
Type of the product : SHIRT
Euclidean distance from input : 15.521170852654832
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B00W95T9UG	Freedi	Black



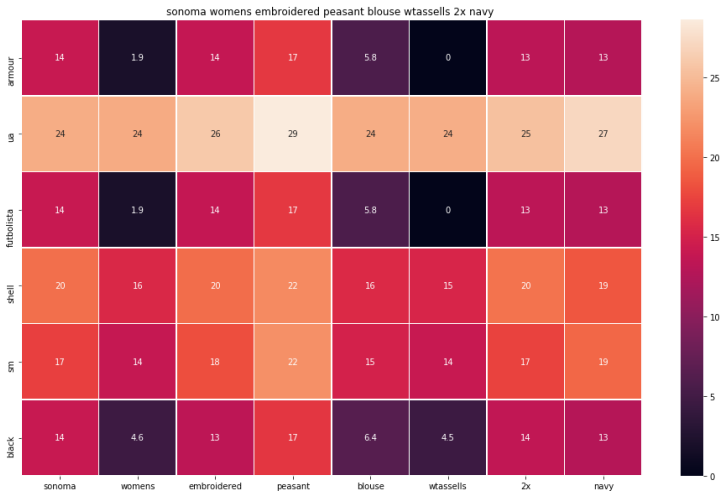
ASIN of the product : B00W95T9UG
Brand of the product : Freedi
Title of the product : freedi womens diamond check lace chest wrap padded bra stretch tank top
Type of the product : SHIRT
Euclidean distance from input : 15.596150449474028
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01EV1NKNW	Creazy	Black



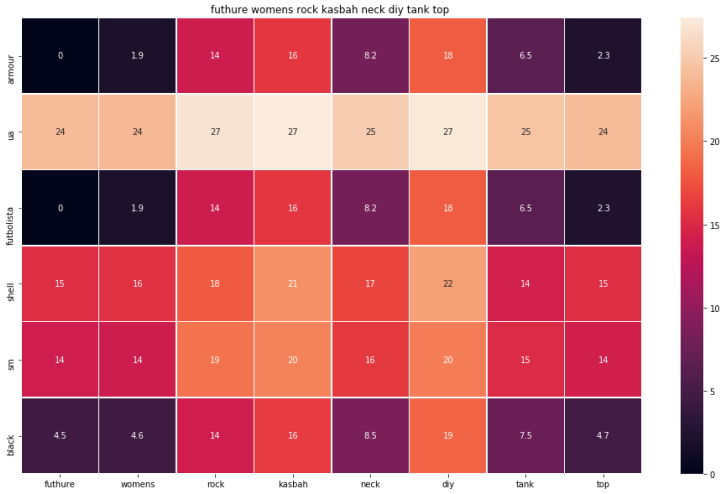
ASIN of the product : B01EV1NKNW
Brand of the product : Creazy
Title of the product : creazy womens long sleeve slim lace blouse casual shirt tops fashion blouse xl black
Type of the product : SHIRT
Euclidean distance from input : 15.59938557433452
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B0755R5GF1	Sonoma	Navy



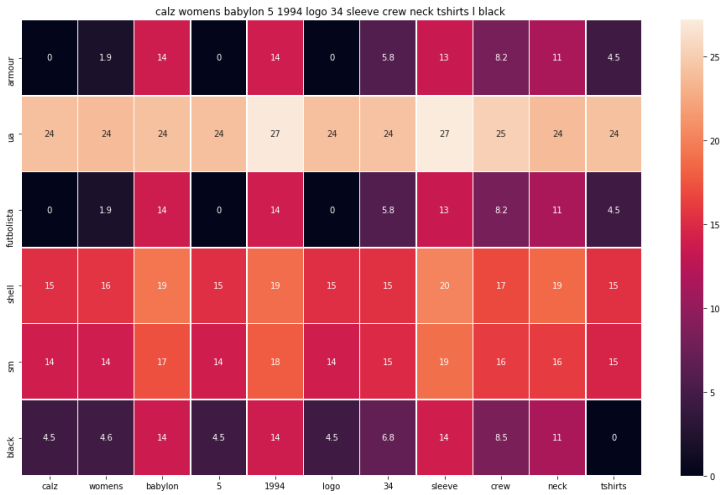
ASIN of the product : B0755R5GF1
Brand of the product : Sonoma
Title of the product : sonoma womens embroidered peasant blouse wtassells 2x navy
Type of the product : SHIRT
Euclidean distance from input : 15.613239421396475
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01KHDMUI4	Futhure	Black



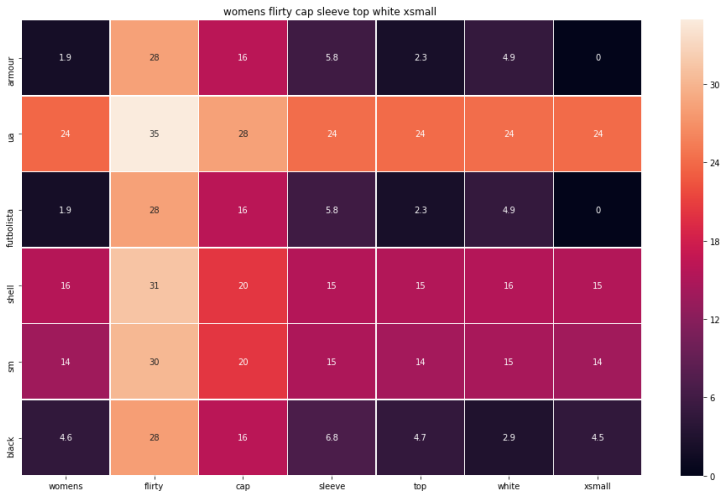
ASIN of the product : B01KHDMUI4
Brand of the product : Futhure
Title of the product : futhure womens rock kasbah neck diy tank top
Type of the product : BOOKS_1973_AND_LATER
Euclidean distance from input : 15.625129854468447
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01FLSHGN4	CALZ	Black



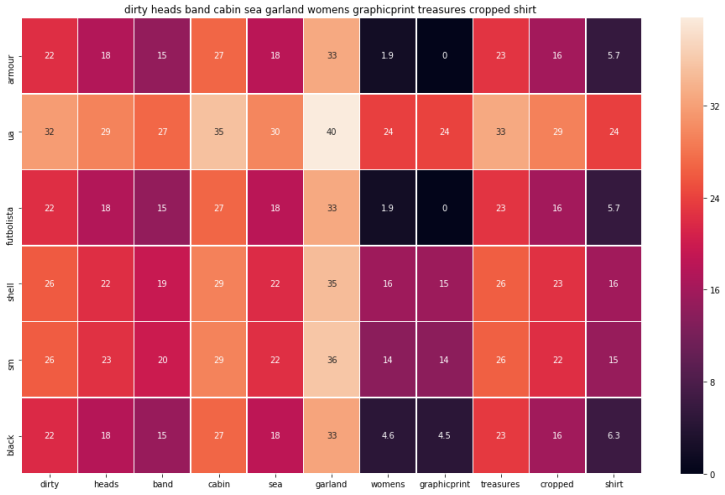
ASIN of the product : B01FLSHGN4
Brand of the product : CALZ
Title of the product : calz womens babylon 5 1994 logo 34 sleeve crew neck ts
hirts l black
Type of the product : BOOKS_1973_AND_LATER
Euclidean distance from input : 15.68121453638809
==========

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B00U7PRVEG	Wilson	White



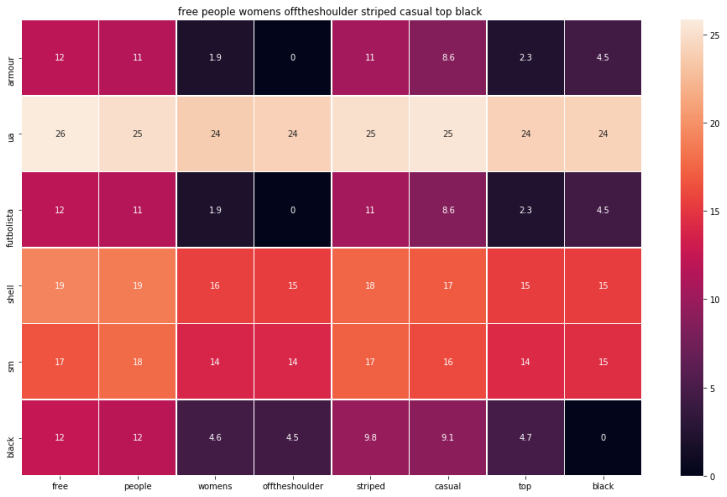
ASIN of the product : B00U7PRVEG
Brand of the product : Wilson
Title of the product : womens flirty cap sleeve top white xsmall
Type of the product : SPORTING_GOODS
Euclidean distance from input : 15.682543228194739
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01HO12FP0	JuligggfaArno	Black



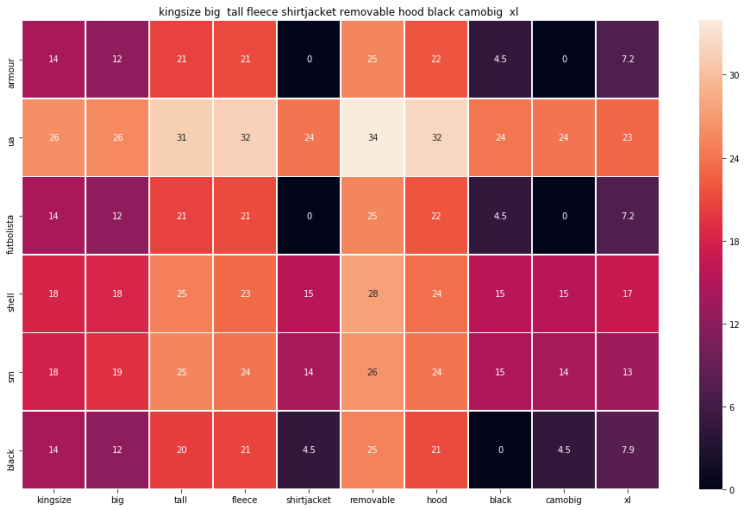
ASIN of the product : B01HO12FP0
Brand of the product : JuligggfaArno
Title of the product : dirty heads band cabin sea garland womens graphicprint treasures cropped shirt
Type of the product : BOOKS_1973_AND_LATER
Euclidean distance from input : 15.728478766298805
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01HCZLYW8	Free-People	Black



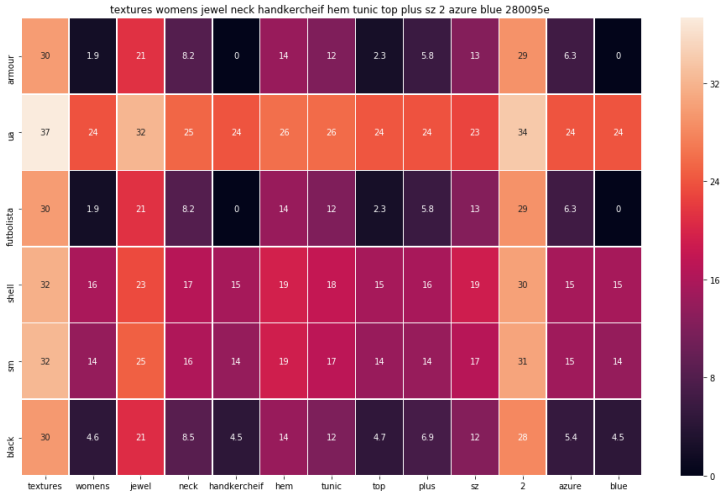
ASIN of the product : B01HCZLYW8
Brand of the product : Free People
Title of the product : free people womens offtheshoulder striped casual top b
lack
Type of the product : SHIRT
Euclidean distance from input : 15.753092432963172
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B0758NFBN7	Bargain-Catalog-Outlet	Black-Camo



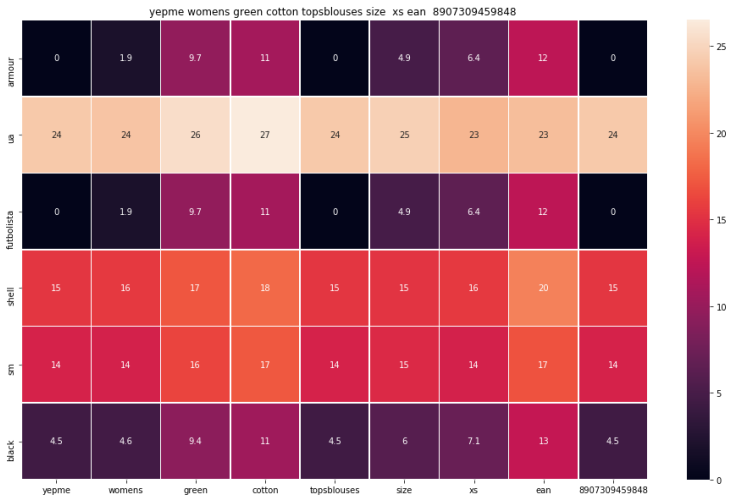
ASIN of the product : B0758NFBN7
Brand of the product : Bargain Catalog Outlet
Title of the product : kingsize big tall fleece shirtjacket removable hood b
lack camobig xl
Type of the product : SHIRT
Euclidean distance from input : 15.770887315904046
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B073T2SP79	Textures	Azure



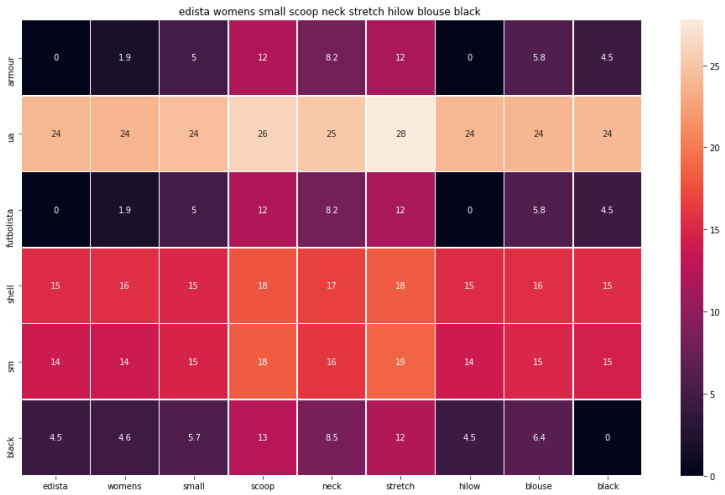
ASIN of the product : B073T2SP79
Brand of the product : Textures
Title of the product : textures womens jewel neck handkercheif hem tunic top
plus sz 2 azure blue 280095e
Type of the product : SHIRT
Euclidean distance from input : 15.78813295864536
==

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01JIUCV40	Yepme	Green



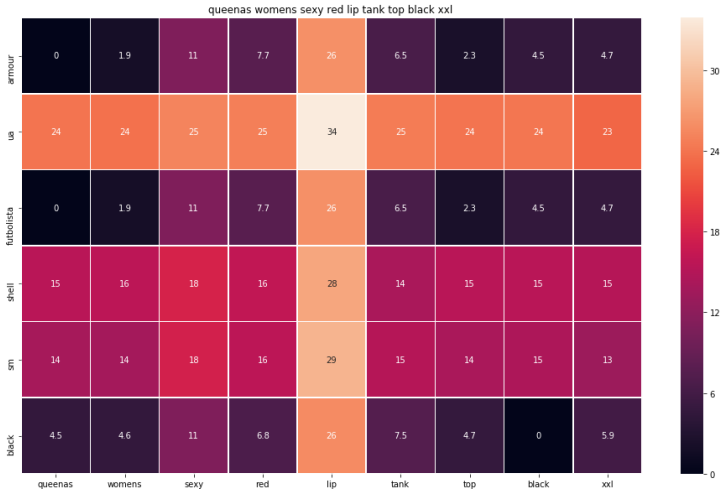
ASIN of the product : B01JIUCV40
Brand of the product : Yepme
Title of the product : yepme womens green cotton topsblouses size xs ean 8907309459848
Type of the product : SHIRT
Euclidean distance from input : 15.79297679583844
==========

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B071F4YG81	Edista	Black



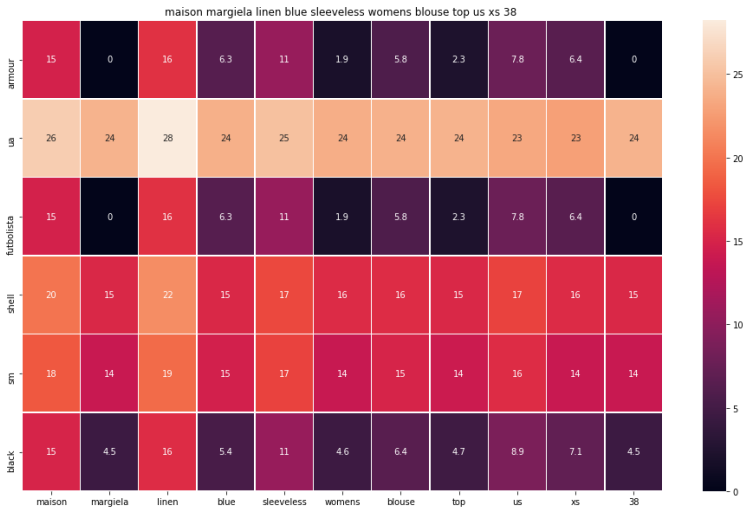
ASIN of the product : B071F4YG81
Brand of the product : Edista
Title of the product : edista womens small scoop neck stretch hilow blouse bl
ack
Type of the product : SHIRT
Euclidean distance from input : 15.811682763229609
==========

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01D77TDV4	QUEENAS	Black



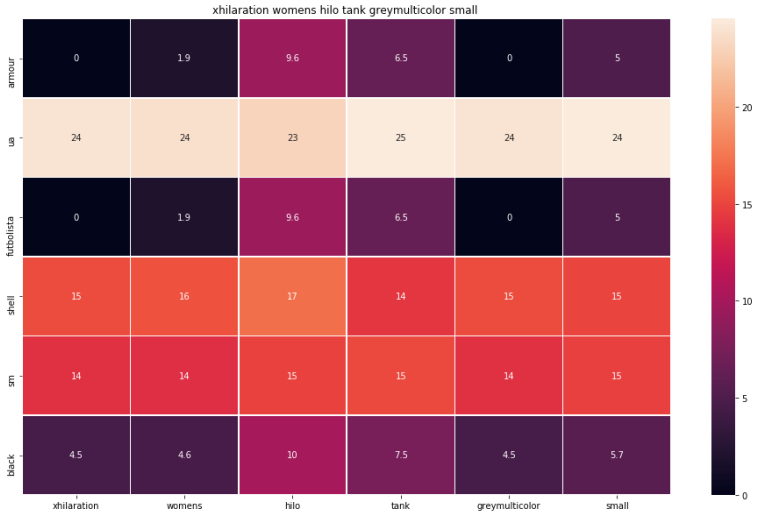
ASIN of the product : B01D77TDV4
Brand of the product : QUEENAS
Title of the product : queenas womens sexy red lip tank top black xxl
Type of the product : BOOKS_1973_AND_LATER
Euclidean distance from input : 15.812993711192778
==========

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B074G3XPWW	Maison-Margiela	Blue



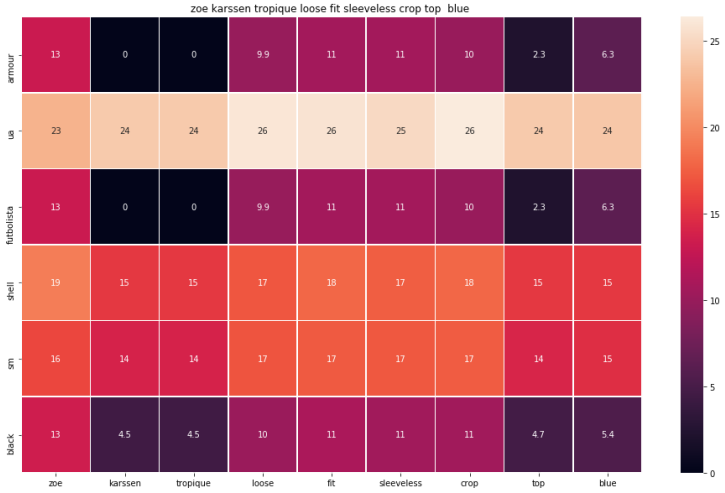
ASIN of the product : B074G3XPWW
Brand of the product : Maison Margiela
Title of the product : maison margiela linen blue sleeveless womens blouse to
p us xs 38
Type of the product : SHIRT
Euclidean distance from input : 15.82193956615794
==========

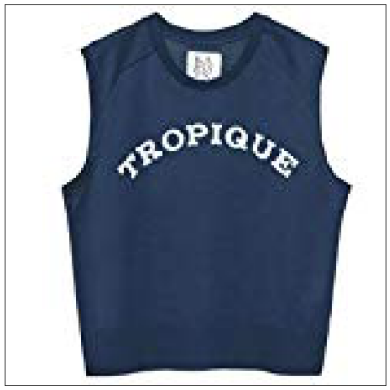
Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B071XX6JBT	Xhilaration	Grey/Multicolor,



ASIN of the product : B071XX6JBT
Brand of the product : Xhilaration
Title of the product : xhilaration womens hilo tank greymulticolor small
Type of the product : SHIRT
Euclidean distance from input : 15.826487283863127
==========

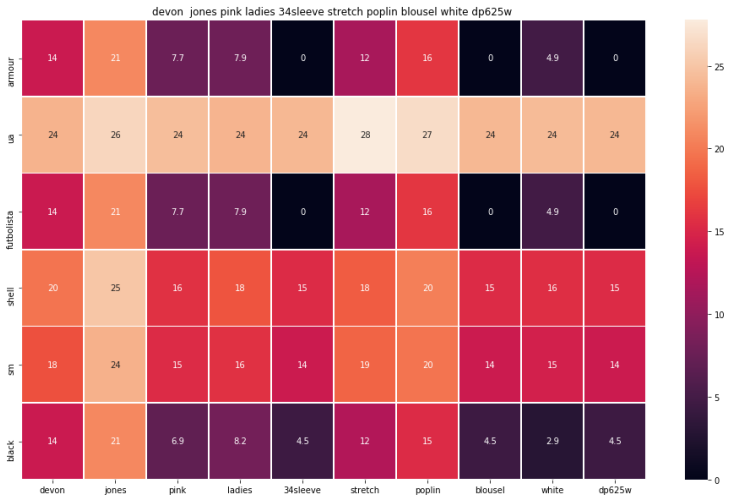
Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01ND46TFT	Zoe-Karssen	Blue





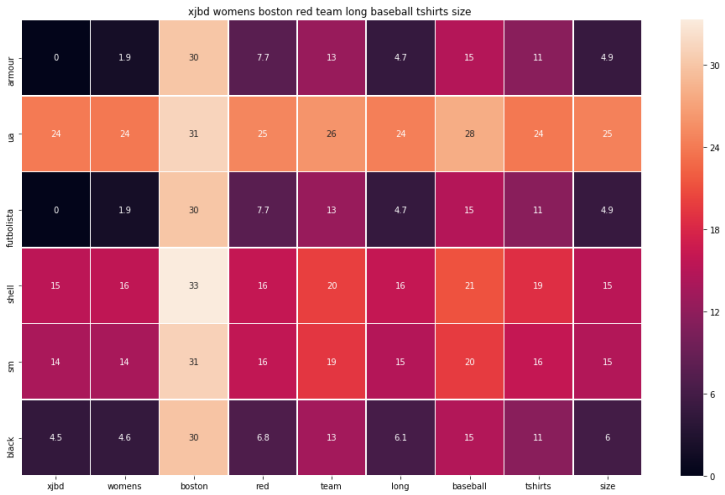
ASIN of the product : B01ND46TFT
Brand of the product : Zoe Karssen
Title of the product : zoe karssen tropique loose fit sleeveless crop top bl
ue
Type of the product : SHIRT
Euclidean distance from input : 15.836460091409057
==========

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B00KV9G0KE	Devon-&-Jones	White



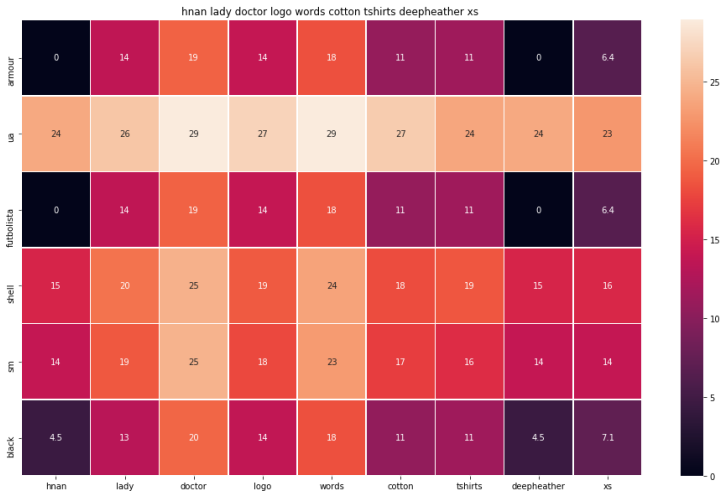
ASIN of the product : B00KV9G0KE
Brand of the product : Devon & Jones
Title of the product : devon jones pink ladies 34sleeve stretch poplin blous
el white dp625w
Type of the product : APPAREL
Euclidean distance from input : 15.837160600616647
==========

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B01M8G3JVK	XJBD	Black



ASIN of the product : B01M8G3JVK
Brand of the product : XJBD
Title of the product : xjbd womens boston red team long baseball tshirts size
Type of the product : BOOKS_1973_AND_LATER
Euclidean distance from input : 15.839540254859072
==========

Asin	Brand	Color
B0185VYVR8	UNDER-ARMOUR->-UA-Tops	Black
B0142LT93Q	H'nan	DeepHeather



ASIN of the product : B0142LT93Q
Brand of the product : H'nan
Title of the product : hnan lady doctor logo words cotton tshirts deepheather xs
Type of the product : SHIRT
Euclidean distance from input : 15.85689281146344
==========

Observation

- As we see that the VGG16 give good result as compaire to all the above performance.

step by step procedure

- First of all we get the data in json file then cleanning and perprocess data.
- Prepared data for perfomaing vaious(BOW,Tfidf,idf,avgw2v) representaion of text data.
- similarly reprent Brand and colour in onehot encoding and image in VGG16.
- Then we usePairwise_distance to compute ecludient distance.
- Then sort according to distance.
- At the end we perform AB test for testing which model perform better.