

**[4.2] Data and Code:**

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

**[4.2] Data and Code:**

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

```
In [3]: M 1 #import all the necessary packages.
2
3 from PIL import Image
4 import requests
5 from io import BytesIO
6 import matplotlib.pyplot as plt
7 import numpy as np
8 import pandas as pd
9 import warnings
10 from bs4 import BeautifulSoup
11 from nltk.corpus import stopwords
12 from nltk.tokenize import word_tokenize
13 import nltk
14 import math
15 import time
16 import re
17 import os
18 import seaborn as sns
19 from collections import Counter
20 from sklearn.feature_extraction.text import CountVectorizer
21 from sklearn.feature_extraction.text import TfidfVectorizer
22 from sklearn.metrics.pairwise import cosine_similarity
23 from sklearn.metrics import pairwise_distances
24 from matplotlib import gridspec
25 from scipy.sparse import hstack
26 import plotly
27 import plotly.figure_factory as ff
28 from plotly.graph_objs import Scatter, Layout
29
30 plotly.offline.init_notebook_mode(connected=True)
31 warnings.filterwarnings("ignore")
```

```
In [4]: M 1 # we have give a json file which consists of all information about
2 # the products
3 # Loading the data using pandas' read_json file.
4 data = pd.read_json('tops_fashion.json')
5
```

```
In [5]: M 1 print ('Number of data points : ', data.shape[0], \
2       '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 [6]: M 1 # each product/item has 19 features in the raw dataset.
2 data.columns # prints column-names or feature-names.
```

```
Out[6]: Index(['sku', 'asin', 'product_type_name', 'formatted_price', 'author',
   'color', 'brand', 'publisher', 'availability', 'reviews',
   'large_image_url', 'availability_type', 'small_image_url',
   'editorial_review', 'title', 'model', 'medium_image_url',
   'manufacturer', 'editorial_review'],
  dtype='object')
```

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

1. asin ( Amazon standard identification number)
2. brand ( brand to which the product belongs to )
3. color ( Color information of apparel, it can contain many colors as a value ex: red and black stripes )
4. product\_type\_name (type of the apparel, ex: SHIRT/TSHIRT )
5. medium\_image\_url ( url of the image )
6. title (title of the product.)
7. formatted\_price (price of the product)

```
In [7]: M 1 data = data[['asin', 'brand', 'color', 'medium_image_url', 'product_type_name', 'title', 'formatted_price']]
```

```
In [8]: M 1 print ('Number of data points : ', data.shape[0], \
2       'Number of features:', data.shape[1])
3 data.head() # prints the top rows in the table.
```

Number of data points : 183138 Number of features: 7

```
Out[8]:
```

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
0	B016I2TS4W	FNC7C	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Minions Como Superheroes Ironman Long Sleeve R...	None
1	B01N49AI08	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG Clothing Womens Izo Tunic	None
2	B01JDPCOHO	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG Clothing Womens Won Top	None
3	B01N19U5H5	Focal18	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Focal18 Sailor Collar Bubble Sleeve Blouse Shi...	None
4	B004GS12OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	Featherlite Ladies' Long Sleeve Stain Resistan...	\$26.26

**Basic stats for the feature: product\_type\_name**

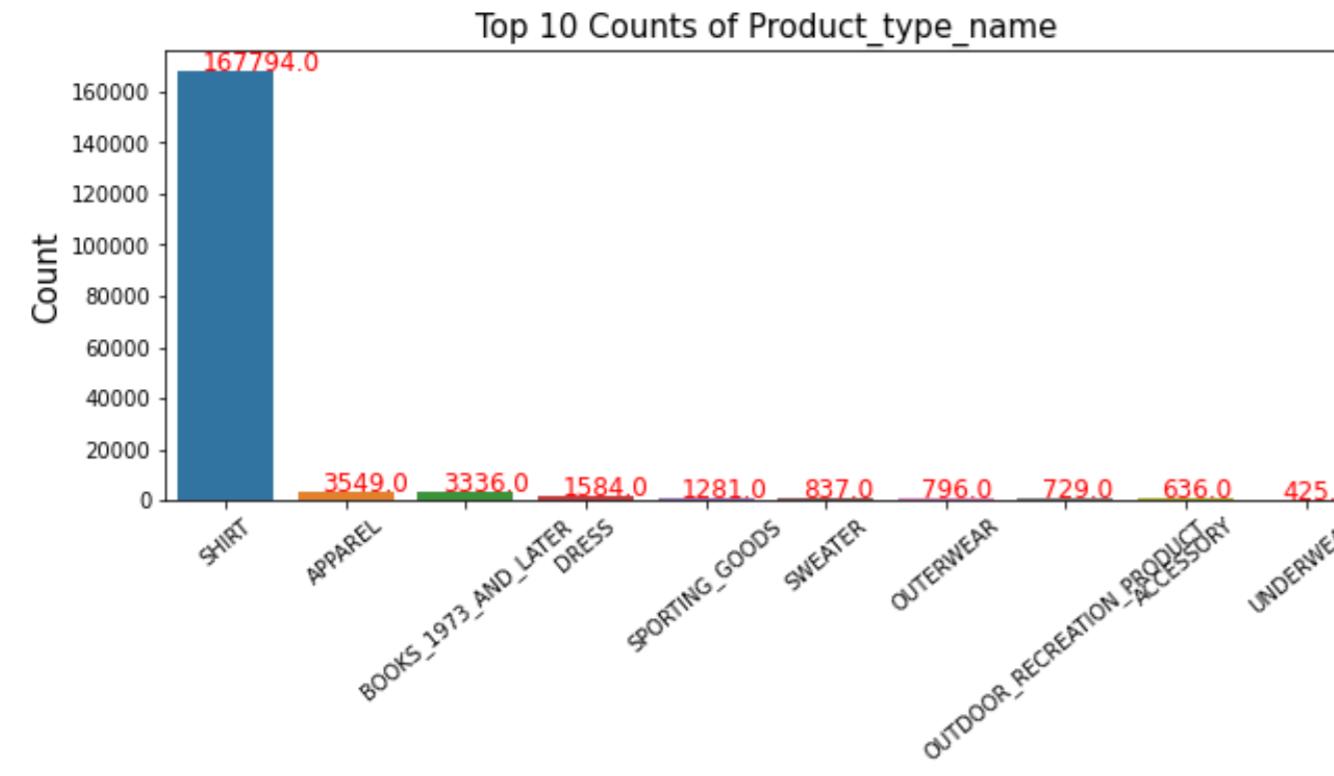
```
In [9]: M 1 # We have total 72 unique type of product_type_names
2 print(data['product_type_name'].value_counts())
3
4 # 91.62% (167794/183138) of the products are shirts,
5
```

```
SHIRT           167794
APPAREL          3549
BOOKS_1973_AND_LATER    3336
DRESS            1584
SPORTING_GOODS      1281
...
CONSUMER_ELECTRONICS    1
ABIS_SPORTS        1
VIDEO_DVD          1
COMPUTER_COMPONENT    1
GOLF_CLUB          1
Name: product_type_name, Length: 72, dtype: int64
```

```
In [10]: M 1 # names of different product types
2 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 [11]: 
1 ## top 10 most used frequent products,
2 fig = plt.figure(figsize = (8, 3))
3 ax = fig.add_axes([0,0,1,1])
4 ax.set_title('Top 10 Counts of Product_type_name', fontsize = 15)
5 top10_counts = pd.DataFrame(data['product_type_name'].value_counts()).reset_index()[:10]
6 sns.barplot(x=top10_counts['index'],y=top10_counts['product_type_name'])
7 for i in ax.patches:
8     ax.text(x = i.get_x() + 0.2, y = i.get_height() + 10, s = str(i.get_height()), fontsize = 12, color = 'red')
9 plt.xlabel('')
10 plt.xticks(rotation=40)
11 plt.ylabel('Count', fontsize = 15)
12 plt.show()
```



In [ ]: 1

## Basic stats for the feature: brand

```
In [12]: 
1 # We have total 72 unique type of product_type_names
2 print(data['brand'].value_counts())
3
4 # 91.62% (167794/183138) of the products are shirts,
5
```

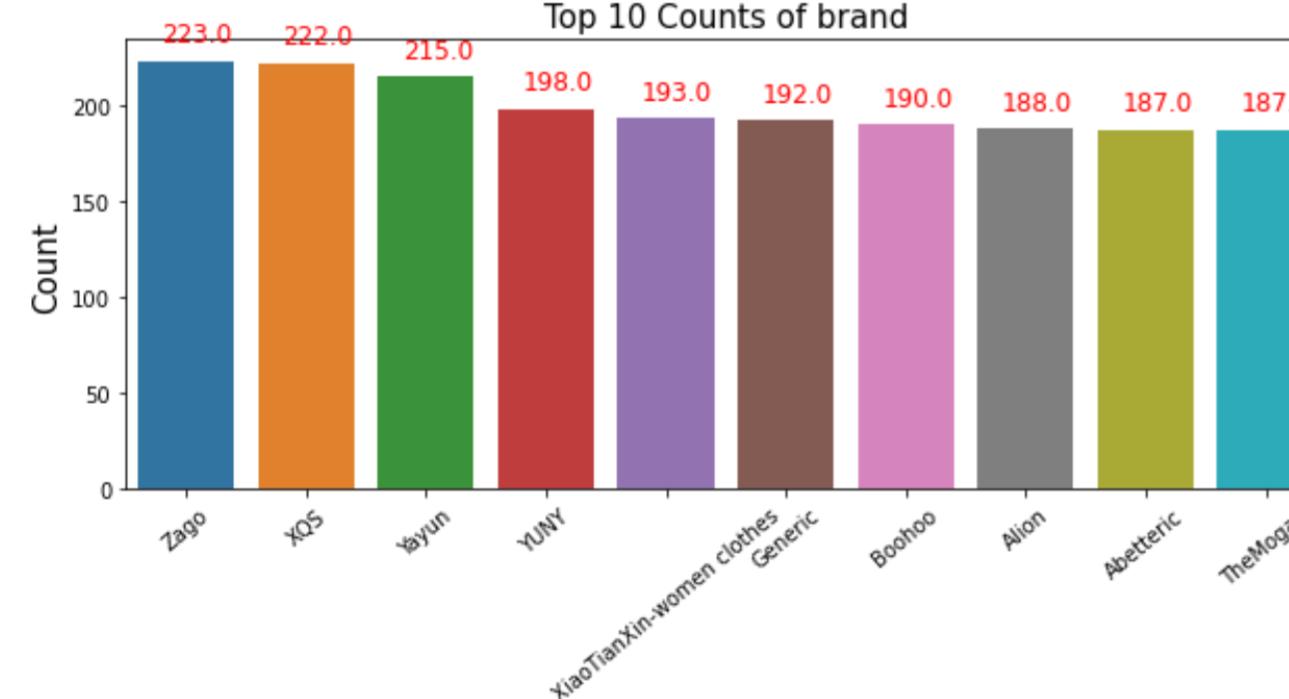
Zago	223
XQS	222
Yayun	215
YUNY	198
XiaoTianXin-women clothes	193
...	
Fornia Apparel	1
Holysong	1
FITS	1
Lady Hathaway	1
DAMEE	1

Name: brand, Length: 10577, dtype: int64

```
In [13]: 
1 # names of different product types
2 print(data['brand'].unique())
```

[FNC7C' 'FIG Clothing' 'Focal18' ... 'Z' "Rain's Pan Jacket" 'FFLMYUHULIU']

```
In [14]: 
1 ## top 10 most used frequent products,
2 fig = plt.figure(figsize = (8, 3))
3 ax = fig.add_axes([0,0,1,1])
4 ax.set_title('Top 10 Counts of brand', fontsize = 15)
5 top10_counts = pd.DataFrame(data['brand'].value_counts()).reset_index()[:10]
6 sns.barplot(x=top10_counts['index'],y=top10_counts['brand'])
7 for i in ax.patches:
8     ax.text(x = i.get_x() + 0.2, y = i.get_height() + 10, s = str(i.get_height()), fontsize = 12, color = 'red')
9 plt.xlabel('')
10 plt.xticks(rotation=40)
11 plt.ylabel('Count', fontsize = 15)
12 plt.show()
```



## Basic stats for the feature: color

```
In [15]: 
1 # We have total 72 unique type of product_type_names
2 print(data['color'].value_counts())
3
4 # 91.62% (167794/183138) of the products are shirts,
5
```

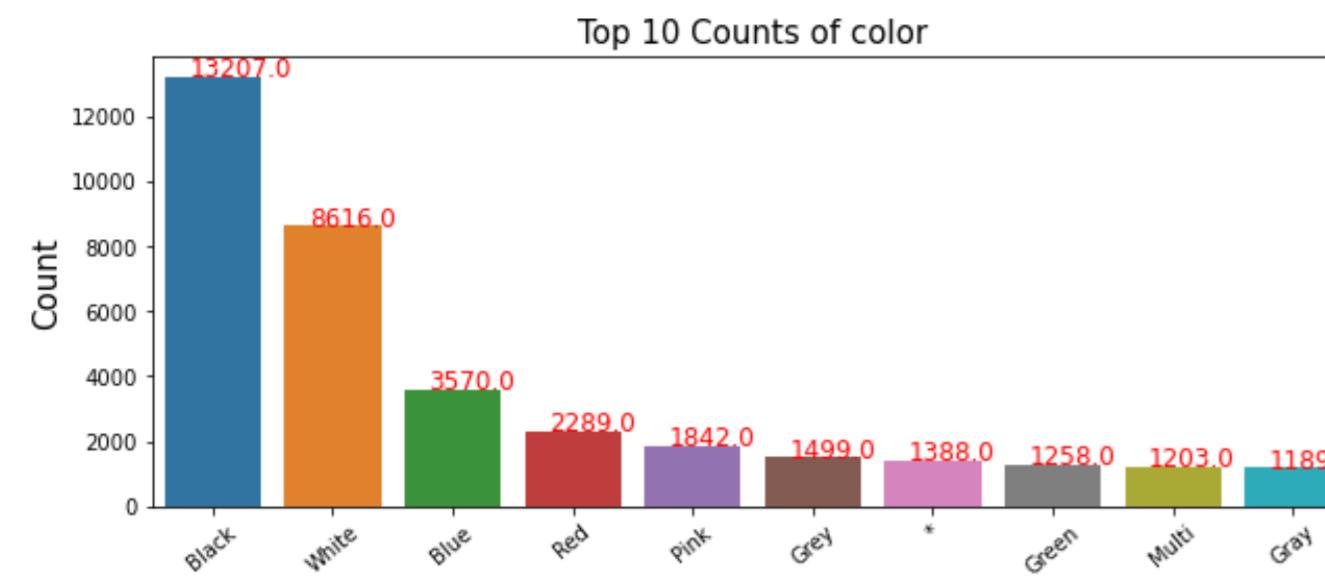
Black	13207
White	8616
Blue	3570
Red	2289
Pink	1842
...	
Football on Red	1
Orchid Gingham	1
Peony/White/Black	1
Ple Apr	1
Retro Heather Navy	1

Name: color, Length: 7380, dtype: int64

```
In [16]: 
1 # names of different product types
2 print(data['color'].unique())
```

[None 'Onyx Black/ Stone' 'Grape' ... 'Combo C' 'White, Soft Coral and Charcoal Varigated' 'Monochrome Plaid']

```
In [17]: 1 ## top 10 most used frequent products,
2 fig = plt.figure(figsize = (8, 3))
3 ax = fig.add_axes([0,0,1,1])
4 ax.set_title('Top 10 Counts of color', fontsize = 15)
5 top10_counts = pd.DataFrame(data['color'].value_counts()).reset_index()[:10]
6 sns.barplot(x=top10_counts['index'],y=top10_counts['color'])
7 for i in ax.patches:
8     ax.text(x = i.get_x() + 0.2, y = i.get_height() + 10, s = str(i.get_height()), fontsize = 12, color = 'red')
9 plt.xlabel('')
10 plt.xticks(rotation=40)
11 plt.ylabel('Count', fontsize = 15)
12 plt.show()
```



Basic stats for the feature: formatted\_price

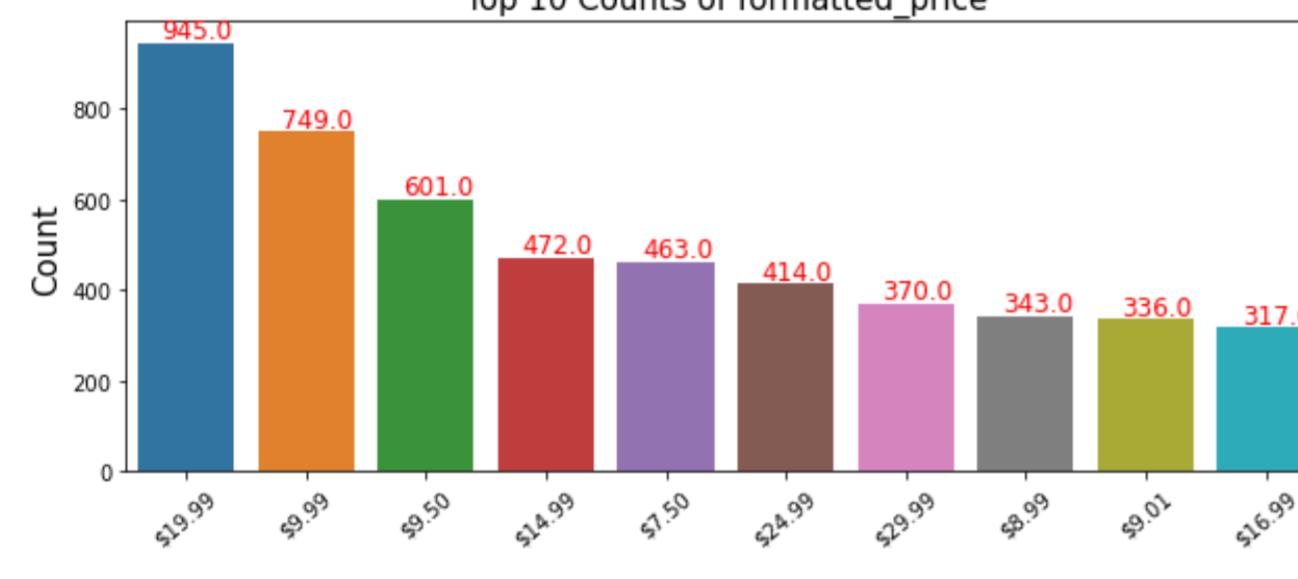
```
In [18]: 1 # We have total 72 unique type of product_type_names
2 print(data['formatted_price'].value_counts())
3
4 # 91.62% (167794/183138) of the products are shirts,
5
```

```
$19.99    945
$9.99    749
$9.50    601
$14.99   472
$7.50    463
...
$22.34    1
$129.97   1
$35.28    1
$100.77   1
$61.48    1
Name: formatted_price, Length: 3135, dtype: int64
```

```
In [19]: 1 # names of different product types
2 print(data['formatted_price'].unique())
```

```
[None '$26.26' '$9.99' ... '$16.05' '$88.29' '$28.05']
```

```
In [20]: 1 ## top 10 most used frequent products,
2 fig = plt.figure(figsize = (8, 3))
3 ax = fig.add_axes([0,0,1,1])
4 ax.set_title('Top 10 Counts of formatted_price', fontsize = 15)
5 top10_counts = pd.DataFrame(data['formatted_price'].value_counts()).reset_index()[:10]
6 sns.barplot(x=top10_counts['index'],y=top10_counts['formatted_price'])
7 for i in ax.patches:
8     ax.text(x = i.get_x() + 0.2, y = i.get_height() + 10, s = str(i.get_height()), fontsize = 12, color = 'red')
9 plt.xlabel('')
10 plt.xticks(rotation=40)
11 plt.ylabel('Count', fontsize = 15)
12 plt.show()
```



#### Imputing missing values

\* let's consider data points without null values in price and color

```
In [21]: 1 # consider products which have price information
2 # data['formatted_price'].isnull() => gives the information
3 # about the dataframe row's which have null values price == None/NULL
4 data = data.loc[~data['formatted_price'].isnull()]
5 print('Number of data points After eliminating price=NULL :', data.shape[0])
```

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

```
In [22]: 1 # consider products which have color information
2 # data['color'].isnull() => gives the information about the dataframe row's which have null values price == None/NULL
3 data = data.loc[~data['color'].isnull()]
4 print('Number of data points After eliminating color=NULL :', data.shape[0])
```

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

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

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

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

```
In [21]: 1 data.to_pickle('pickels/28k_apparel_data')
```

```
In [23]: 1 from PIL import Image
2 import requests
3 from io import BytesIO
4 i = 0
5 # for index, row in images.iterrows():
6 #     try:
7 #         url = row['large_image_url']
8 #         response = requests.get(url)
9 #         img = Image.open(BytesIO(response.content))
10 #         img.save('images/28k_images/' + row['asin'] + '.jpeg')
11 #         i+=1
12 #     except:
13 #         print('Unidentified image error: ', i)
14 #         i+=1
```

#### [5.2] Remove near duplicate items

##### [5.2.1] Understand about duplicates.

```
In [23]: 1 # read data from pickle file from previous stage
2 data = pd.read_pickle('pickels/28k_apparel_data')
3
4 # find number of products that have duplicate titles.
5 print(sum(data.duplicated(['title'])))
6
7 # we have 2325 products which have same title but different color
```

```
2325
```

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





These shirts exactly same except in color



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 [24]: 1 # read data from pickle file from previous stage
2 data = pd.read_pickle('pickels/28k_apparel_data')
```

```
In [25]: 1 data.head()
```

Out[25]:

asin	brand	color	medium_image_url	product_type_name	title	formatted_price
4 B004GS12OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	Featherlite Ladies' Long Sleeve Stain Resistan...	\$26.26
6 B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	Women's Unique 100% Cotton T - Special Olympi...	\$9.99
11 B001LOUGE4	Fitness Etc.	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	Ladies Cotton Tank 2x1 Ribbed Tank Top	\$11.99
15 B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	FeatherLite Ladies' Moisture Free Mesh Sport S...	\$20.54
21 B014ICEDNA	FNCTC	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	Supernatural Chibis Sam Dean And Castiel Short...	\$7.50

```
In [26]: 1 # Remove All products with very few words in title
2 data_sorted = data[data['title'].apply(lambda x: len(x.split())>4)]
3 print("After removal of products with short description:", data_sorted.shape[0])
```

After removal of products with short description: 27949

```
In [27]: 1 # Sort the whole data based on title (alphabetical order of title)
2 data_sorted.sort_values('title', inplace=True, ascending=False)
3 data_sorted.head()
```

Out[27]:

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

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

Title 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 [46]: 1 indices = []
2 for i, row in data_sorted.iterrows():
3     indices.append(i)
```

```
In [42]: 1 import itertools
2 stage1_deduplicate_asins = []
3 i = 0
4 j = 0
5 num_data_points = data_sorted.shape[0]
6 while i < num_data_points and j < num_data_points:
7
8     previous_i = i
9
10    # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'X-Large']
11    a = data['title'].loc[indices[i]].split()
12
13    # search for the similar products sequentially
14    j = i+1
15    while j < num_data_points:
16
17        # store the list of words of jth string in b, ex: b = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'Small']
18        b = data['title'].loc[indices[j]].split()
19
20        # store the maximum length of two strings
21        length = max(len(a), len(b))
22
23        # count is used to store the number of words that are matched in both strings
24        count = 0
25
26        # itertools.zip_longest(a,b): will map the corresponding words in both strings, it will append None in case of unequal strings
27        # example: a =[a', 'b', 'c', 'd']
28        # b =[a', 'b', 'd']
29        # itertools.zip_longest(a,b): will give [(a', 'a'), ('b', 'b'), ('c', 'd'), ('d', None)]
30        for k in itertools.zip_longest(a,b):
31            if (k[0] == k[1]):
32                count += 1
33
34        # if the number of words in which both strings differ are > 2 , we are considering it as those two apparel are different
35        # if the number of words in which both strings differ are < 2 , we are considering it as those two apparel are same, hence we are ignoring them
36        if (length - count) > 2: # number of words in which both sensences differ
37            # if both strings are differ by more than 2 words we include the 1st string index
38            stage1_deduplicate_asins.append(data_sorted['asin'].loc[indices[i]])
39
40
41        # start searching for similar apparel corresponds 2nd string
42        i = j
43        break
44    else:
45        j += 1
46    if previous_i == i:
47        break
```

```
In [43]: 1 data = data.loc[data['asin'].isin(stage1_deduplicate_asins)]
```

We removed the duplicates which differ only at the end.

```
In [44]: 1 print('Number of data points : ', data.shape[0])
```

Number of data points : 17592

```
In [45]: 1 data.to_pickle('pickels/17k_apparel_data')
```

Let's remove duplicates part - 2

In the previous cell, we sorted whole data in alphabetical order of titles. Then, we removed titles which are adjacent and very similar title

But there are some products whose titles are not adjacent but very similar.

Examples:

**Titles-1**  
86261. UltraClub Women's Classic Wrinkle-Free Long Sleeve Oxford Shirt, Pink, XX-Large  
115042. UltraClub Ladies Classic Wrinkle-Free Long-Sleeve Oxford Light Blue XXL

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

```
In [51]: 1 # This code snippet takes significant amount of time.
2 # O(n^2) time.
3 # Takes about an hour to run on a decent computer.
4
5 indices = []
6 for i, row in data.iterrows():
7     indices.append(i)
8
9 stage2_deduplicate_asins = []
10
11 while len(indices) != 0:
12     i = indices.pop()
13     stage2_deduplicate_asins.append(data['asin'].loc[i])
14     # consider the first apparel's title
15     a = data['title'].loc[i].split()
16     # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'X-Large']
17     for j in indices:
18
19         b = data['title'].loc[j].split()
20         # store the list of words of jth string in b, ex: b = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'X-Large']
21
22         length = max(len(a), len(b))
23
24         # count is used to store the number of words that are matched in both strings
25         count = 0
26
27         # itertools.zip_longest(a,b): will map the corresponding words in both strings, it will append None in case of unequal strings
28         # example: a = ['a', 'b', 'c', 'd']
29         # b = ['a', 'b', 'd']
30         # itertools.zip_longest(a,b): will give [('a', 'a'), ('b', 'b'), ('c', 'd'), ('d', None)]
31         for k in itertools.zip_longest(a, b):
32             if (k[0] == k[1]):
33                 count += 1
34
35         # if the number of words in which both strings differ are < 3 , we are considering it as those two apparels are same, hence we are ignoring them
36         if (length - count) < 3:
37             indices.remove(j)
```

```
In [52]: 1 # from whole previous products we will consider only
2 # the products that are found in previous cell
3 data = data.loc[data['asin'].isin(stage2_deduplicate_asins)]
```

```
In [53]: 1 print('Number of data points after stage two of dedupe: ', data.shape[0])
2 # from 17k apparels we reduced to 16k apparels
```

Number of data points after stage two of dedupe: 16434

```
In [54]: 1 data.to_pickle('pickles/16k_apparel_data')
2 # Storing these products in a pickle file
3 # candidates who wants to download these files instead
4 # of 180K they can download and use them from the Google Drive folder.
```

## 6. Text pre-processing

```
In [24]: 1 data = pd.read_pickle('pickles/16k_apparel_data')
2
3 # NLTK download stop words. [RUN ONLY ONCE]
4 # goto Terminal (Linux/Mac) or Command-Prompt (Window)
5 # In the terminal, type these commands
6 # $python3
7 # $import nltk
8 # $nltk.download()
```

```
In [25]: 1 # we use the list of stop words that are downloaded from nltk lib.
2 stop_words = set(stopwords.words('english'))
3 print('list of stop words:', stop_words)
4
5 def nlp_preprocessing(total_text, index, column):
6     if type(total_text) is not int:
7         string = ""
8         for words in total_text.split():
9             # remove the special chars in review like "#$@!%^&(*_+-~?>< etc.
10            word = ("").join([for e in words if e.isalnum()])
11            # Convert all letters to lower-case
12            word = word.lower()
13            # stop-word removal
14            if not word in stop_words:
15                string += word + " "
16        data[column][index] = string
```

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

```
In [26]: 1 start_time = time.clock()
2 # we take each title and we text-preprocess it.
3 for index, row in data.iterrows():
4     nlp_preprocessing(row['title'], index, 'title')
5 # we print the time it took to preprocess whole titles
6 print(time.clock() - start_time, "seconds")
```

5.6766508 seconds

```
In [27]: 1 data.head()
```

```
Out[27]:
```

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

```
In [51]: 1 data.to_pickle('pickles/16k_apparel_data_preprocessed')
```

## [8] Text based product similarity

```
In [28]: 1 data = pd.read_pickle('pickles/16k_apparel_data_preprocessed')
2 data.head()
```

```
Out[28]:
```

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

In [41]:

```

1 # Utility Functions which we will use through the rest of the workshop.
2
3
4 #Display an image
5 def display_img(url,ax,fig):
6     # we get the url of the apparel and download it
7     response = requests.get(url)
8     img = Image.open(BytesIO(response.content))
9     # we will display it in notebook
10    plt.imshow(img)
11
12 #plotting code to understand the algorithm's decision.
13 def plot_heatmap(keys, values, labels, url, text):
14     # keys: list of words of recommended title
15     # values: len(values) == len(keys), values(i) represents the occurence of the word keys(i)
16     # labels: len(labels) == len(keys), the values of labels depends on the model we are using
17     # if model == 'bag of words': labels(i) = values(i)
18     # if model == 'tfidf weighted bag of words':labels(i) = tfidf(keys(i))
19     # if model == 'idf weighted bag of words':labels(i) = idf(keys(i))
20     # url : apparel's url
21
22     # we will devide the whole figure into two parts
23     gs = gridspec.GridSpec(2, 2, width_ratios=[4,1], height_ratios=[4,1])
24     fig = plt.figure(figsize=(25,3))
25
26     # 1st, plotting heat map that represents the count of commonly occurred words in title2
27     ax = plt.subplot(gs[0])
28     # 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
29     ax = sns.heatmap(np.array([values]), annot=np.array([labels]))
30     ax.set_xticklabels(keys) # set that axis labels as the words of title
31     ax.set_title(text) # apparel title
32
33     # 2nd, plotting image of the the apparel
34     ax = plt.subplot(gs[1])
35     # we don't want any grid Lines for image and no labels on x-axis and y-axis
36     ax.grid(False)
37     ax.set_xticks([])
38     ax.set_yticks([])
39
40     # we call display_img based with paramete url
41     display_img(url, ax, fig)
42
43     # displays combine figure ( heat map and image together)
44     plt.show()
45
46 def plot_heatmap_image(doc_id, vec1, vec2, url, text, model):
47
48     # doc_id : index of the title1
49     # vec1 : input apparel's vector, it is of a dict type {word:count}
50     # vec2 : recommended apparel's vector, it is of a dict type {word:count}
51     # url : apparel's image url
52     # text: title of recomended apparel (used to keep title of image)
53     # model, it can be any of the models,
54     # 1. bag_of_words
55     # 2. tfidf
56     # 3. idf
57
58     # we find the common words in both titles, because these only words contribute to the distance between two title vec's
59     intersection = set(vec1.keys()) & set(vec2.keys())
60
61     # we set the values of non intersecting words to zero, this is just to show the difference in heatmap
62     for i in vec2:
63         if i not in intersection:
64             vec2[i]=0
65
66     # for Labeling heatmap, keys contains List of all words in title2
67     keys = list(vec2.keys())
68     # 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
69     values = [vec2[x] for x in vec2.keys()]
70
71     # labels: len(labels) == len(keys), the values of labels depends on the model we are using
72     # if model == 'bag of words': labels(i) = values(i)
73     # if model == 'tfidf weighted bag of words':labels(i) = tfidf(keys(i))
74     # if model == 'idf weighted bag of words':labels(i) = idf(keys(i))
75
76     if model == 'bag_of_words':
77         labels = values
78     elif model == 'tfidf':
79         labels = []
80         for x in vec2.keys():
81             # tfidf_title_vectorizer.vocabulary_ it contains all the words in the corpus
82             # tfidf_title_features[doc_id, index_of_word_in_corpus] will give the tfidf value of word in given document (doc_id)
83             if x in tfidf_title_vectorizer.vocabulary_:
84                 labels.append(tfidf_title_features[doc_id, tfidf_title_vectorizer.vocabulary_[x]])
85             else:
86                 labels.append(0)
87     elif model == 'idf':
88         labels = []
89         for x in vec2.keys():
90             # idf_title_vectorizer.vocabulary_ it contains all the words in the corpus
91             # idf_title_features[doc_id, index_of_word_in_corpus] will give the idf value of word in given document (doc_id)
92             if x in idf_title_vectorizer.vocabulary_:
93                 labels.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[x]])
94             else:
95                 labels.append(0)
96
97     plot_heatmap(keys, values, labels, url, text)
98
99
100 # this function gets a list of words along with the frequency of each
101 # word given "text"
102 def text_to_vector(text):
103     word = re.compile(r'\w+')
104     words = word.findall(text)
105     # words stores list of all words in given string, you can try 'words = text.split()' this will also gives same result
106     return Counter(words) # Counter counts the occurence of each word in list, it returns dict type object {word1:count}
107
108
109
110 def get_result(doc_id, content_a, content_b, url, model):
111     text1 = content_a
112     text2 = content_b
113
114     # vector1 = dict(word1:#count, word2:#count, etc.)
115     vector1 = text_to_vector(text1)
116
117     # vector1 = dict(word1:#count, word2:#count, etc.)
118     vector2 = text_to_vector(text2)
119
120     plot_heatmap_image(doc_id, vector1, vector2, url, text2, model)

```

In [30]:

```

1 ### Let's get the given image
2
3 url = data.iloc[12500]['medium_image_url']
4 response = requests.get(url)
5 img = Image.open(BytesIO(response.content))
6
7 fig = plt.figure(figsize=(10,5))
8 plt.imshow(img)
9 plt.title('Query Image for our models:\n')
10 print('*'*70)
11 print('\n')
12 print(data.iloc[12500]['title'])
13 print('\n')
14 print('*'*70)
15
*****
```

ralph lauren active womens striped hooded tee yellowwhite xs

\*\*\*\*\*

Query Image for our models:



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

In [30]:

```
1 from sklearn.feature_extraction.text import CountVectorizer
2 title_vectorizer = CountVectorizer()
3 title_features = title_vectorizer.fit_transform(data['title'])
4 title_features.get_shape() # get number of rows and columns in feature matrix.
5 # title_features.shape = #data_points * #words_in_corpus
6 # CountVectorizer().fit_transform(corpus) returns
7 # the a sparse matrix of dimensions #data_points * #words_in_corpus
8 # What is a sparse vector?
9 # title_features[doc_id, index_of_word_in_corpus] = number of times the word occurred in that doc
```

Out[30]: (16434, 12684)

In [32]:

```

1 def bag_of_words_model(doc_id, num_results):
2     # doc_id: apparel's id in given corpus
3
4     # pairwise_dist will store the distance from given input apparel to all remaining apparels
5     # the metric we used here is cosine, the coside distance is mesured as  $K(X, Y) = \langle X, Y \rangle / (\|X\| * \|Y\|)$ 
6     # http://scikit-learn.org/stable/modules/_metrics.html#cosine-similarity
7     pairwise_dist = pairwise_distances(title_features,title_features[doc_id])
8
9     # np.argsort will return indices of the smallest distances
10    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
11    pdists = np.sort(pairwise_dist.flatten())[0:num_results]
12
13    #data frame indices of the 9 smallest distance's
14    df_indices = list(data.index[indices])
15
16    for i in range(0,len(indices)):
17        # we will pass 1. doc_id, 2. title1, 3. title2, url, model
18        get_result(indices[i],data['title'].loc[df_indices[0]],data['title'].loc[df_indices[i]],data['medium_image_url'].loc[df_indices[i]],'bag_of_words')
19        print('ASIN :',data['asin'].loc[df_indices[i]])
20        print('Brand:', data['brand'].loc[df_indices[i]])
21        print ('Title:', data['title'].loc[df_indices[i]])
22        print ('Euclidean similarity with the query image :', pdists[i])
23        print('= '*60)
24
25    #call the bag-of-words model for a product to get similar products.
26    bag_of_words_model(12500, 20) # change the index if you want to.
27    # In the output heat map each value represents the count value
28    # of the label word, the color represents the intersection
29    # with inputs title.
30
31    #try 12566
32    #try 931

```

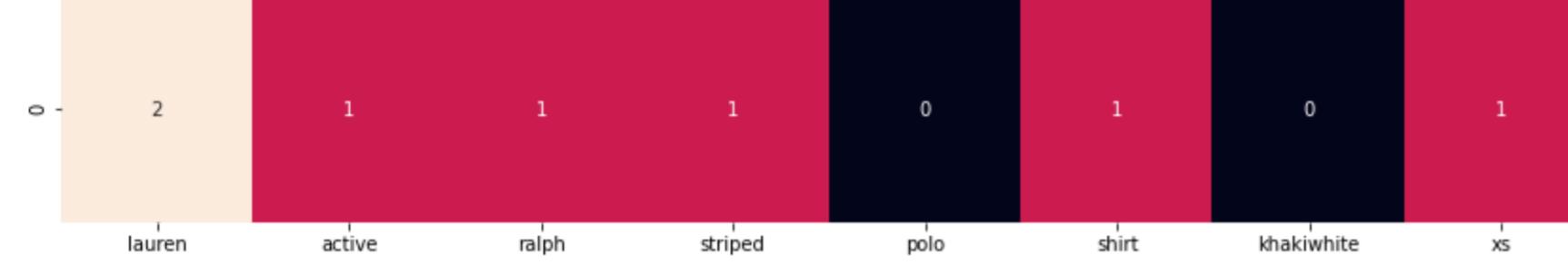
ralph lauren active womens striped hooded tee shirt yellowwhite xs



ASIN : B01JME4H6W  
Brand: Ralph Lauren Active  
Title: ralph lauren active womens striped hooded tee shirt yellowwhite xs  
Euclidean similarity with the query image : 0.0

=====

ralph lauren active ralph lauren striped polo shirt khakiwhite xs



ASIN : B00ILGK6H2  
Brand: Ralph Lauren Active  
Title: lauren active ralph lauren striped polo shirt khakiwhite xs  
Euclidean similarity with the query image : 2.645751310645907

=====

ralph lauren active womens fitness shirt yellowblack small



ASIN : B073WGPVWV  
Brand: Ralph Lauren Active  
Title: ralph lauren active womens fitness shirt yellowblack small  
Euclidean similarity with the query image : 2.8284271247461903

=====

lauren ralph lauren womens medium button shirt blue



ASIN : B074ZLGXPL  
Brand: Lauren by Ralph Lauren  
Title: lauren ralph lauren womens medium button shirt blue  
Euclidean similarity with the query image : 3.1622776601683795

=====

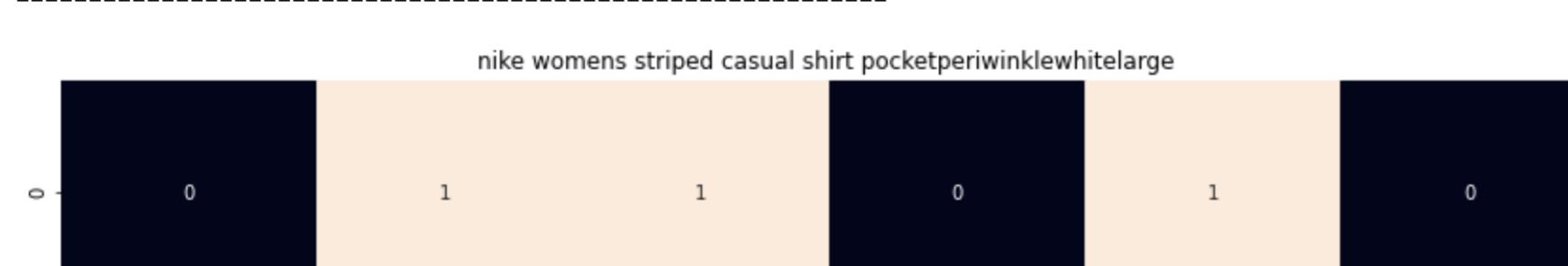
polo ralph lauren womens vneck tee shirt top xs small pinknavy



ASIN : B01ETUBK3W  
Brand: RALPH LAUREN  
Title: polo ralph lauren womens vneck tee shirt top xs small pinknavy  
Euclidean similarity with the query image : 3.1622776601683795

=====

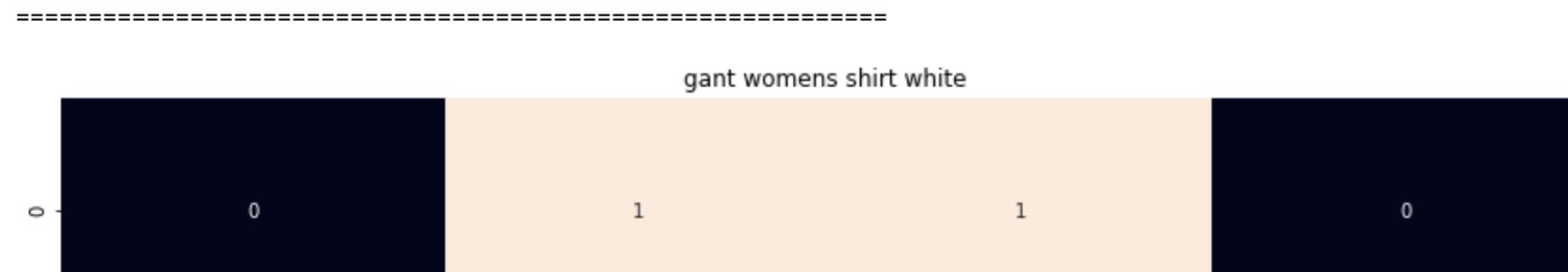
nike womens striped casual shirt pocketperiwinklewhitelarge



ASIN : B003VJMU7S  
Brand: Estwarkim  
Title: nike womens striped casual shirt pocketperiwinklewhitelarge  
Euclidean similarity with the query image : 3.1622776601683795

=====

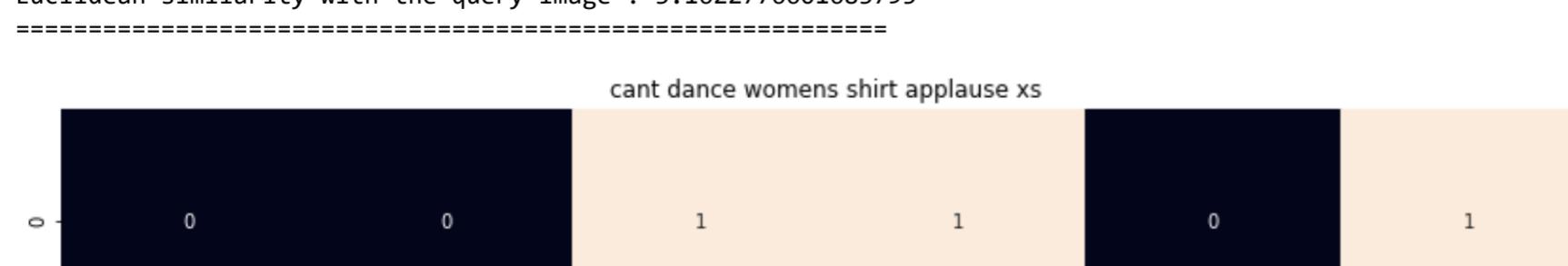
gant womens shirt white



ASIN : B01N80ZGXZ  
Brand: GANT  
Title: gant womens shirt white  
Euclidean similarity with the query image : 3.1622776601683795

=====

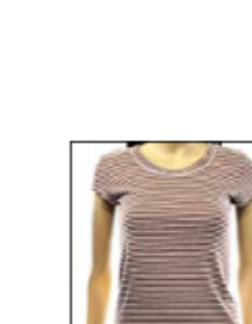
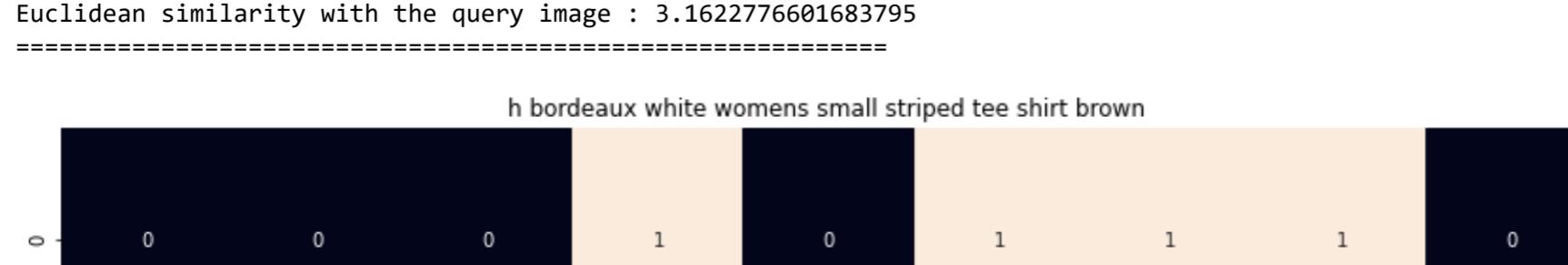
cant dance womens shirt applause xs



ASIN : B01IN5NRE  
Brand: I can't I have dance  
Title: cant dance womens shirt applause xs  
Euclidean similarity with the query image : 3.1622776601683795

=====

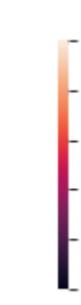
h bordeaux white womens small striped tee shirt brown



ASIN : B072BV847Z  
 Brand: H By Bordeaux  
 Title: h bordeaux white womens small striped tee shirt brown  
 Euclidean similarity with the query image : 3.1622776601683795  
 =====

ralph lauren womens short sleeve oxford button shirt xs white

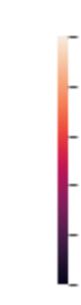
0	1	1	1	0	0	0	0	1	1	0
ralph	lauren	womens	short	sleeve	oxford	button	shirt	xs	white	



ASIN : B01FRIMR5U  
 Brand: RALPH LAUREN  
 Title: ralph lauren womens short sleeve oxford button shirt xs white  
 Euclidean similarity with the query image : 3.1622776601683795  
 =====

ralph lauren womens shirt size large winter cream

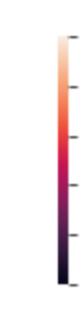
0	1	1	1	1	0	0	0	0	0
ralph	lauren	womens	shirt	size	large	winter	cream		



ASIN : B005YBMMTM  
 Brand: Lauren Jeans Co.  
 Title: ralph lauren womens shirt size large winter cream  
 Euclidean similarity with the query image : 3.1622776601683795  
 =====

state womens highlow shirt xs beige

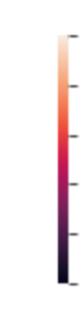
0	1	0	1	1	1	0
state	womens	highlow	shirt	xs	beige	



ASIN : B074MGSQ7J  
 Brand: State of Being  
 Title: state womens highlow shirt xs beige  
 Euclidean similarity with the query image : 3.1622776601683795  
 =====

j brand womens pinstripe shirt xs blue

0	0	1	0	1	1	0
j	brand	womens	pinstripe	shirt	xs	blue



ASIN : B06XYPIX1F  
 Brand: J Brand Jeans  
 Title: j brand womens pinstripe shirt xs blue  
 Euclidean similarity with the query image : 3.1622776601683795  
 =====

beachlunchlounge womens shirt blue

0	0	1	1	0
beachlunchlounge	womens	shirt	blue	



ASIN : B074SZ54KJ  
 Brand: Beach Lunch Lounge  
 Title: beachlunchlounge womens shirt blue  
 Euclidean similarity with the query image : 3.1622776601683795  
 =====

ralph lauren womens classic fit mesh shirt white

0	1	1	1	0	0	0	1	0
ralph	lauren	womens	classic	fit	mesh	shirt	white	



ASIN : B01KIXF87S  
 Brand: RALPH LAUREN  
 Title: ralph lauren womens classic fit mesh shirt white  
 Euclidean similarity with the query image : 3.1622776601683795  
 =====

merona womens favorite v neck tee sapphire xs

0	0	1	0	0	0	1	0	1
merona	womens	favorite	v	neck	tee	sapphire	xs	



ASIN : B06ZZXJ73F  
 Brand: Merona  
 Title: merona womens favorite v neck tee sapphire xs  
 Euclidean similarity with the query image : 3.3166247903554  
 =====

womens oxford shirt long sleeve

0	1	0	1	0	0	0
womens	oxford	shirt	long	sleeve		



ASIN : B01577JZOC  
 Brand: Boast  
 Title: womens oxford shirt long sleeve  
 Euclidean similarity with the query image : 3.3166247903554  
 =====

piper womens bei top xs

0	0	1	0	0	1
piper	womens	bei	top	xs	



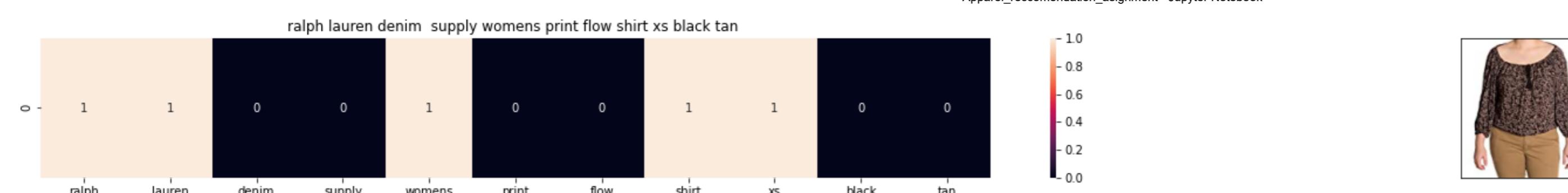
ASIN : B06X6FFZXQ  
 Brand: Piper  
 Title: piper womens bei top xs  
 Euclidean similarity with the query image : 3.3166247903554  
 =====

tyler boe womens polo shirt

0	0	1	0	1
tyler	boe	womens	polo	shirt



ASIN : B01MQ0TIKA  
 Brand: TYLER BOE  
 Title: tyler boe womens polo shirt  
 Euclidean similarity with the query image : 3.3166247903554  
 =====



ASIN : B00X07M40Y  
 Brand: RALPH LAUREN  
 Title: ralph lauren denim supply womens print flow shirt xs black tan  
 Euclidean similarity with the query image : 3.3166247903554  
 =====



- We can see that the BOW model is able to capture the attributes brand ralph lauren, size xs, and womens shirts of the query point well.
- But there are few irrelevant predictions as well

### [8.5] TF-IDF based product similarity

```
In [37]: 1 tfidf_title_vectorizer = TfidfVectorizer(min_df = 0)
2 tfidf_title_features = tfidf_title_vectorizer.fit_transform(data['title'])
3 # tfidf_title_features.shape = #data_points * # words_in_corpus
4 # CountVectorizer().fit_transform(corporus) returns the a sparse matrix of dimensions #data_points * #words_in_corpus
5 # tfidf_title_features[doc_id, index_of_word_in_corpus] = tfidf values of the word in given doc
```

In [38]:

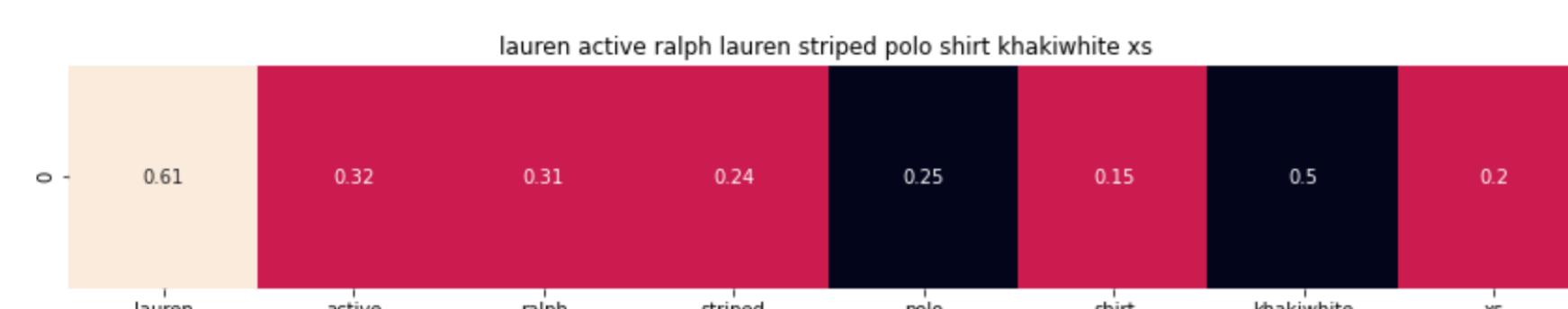
```

1 def tfidf_model(doc_id, num_results):
2     # doc_id: apparel's id in given corpus
3
4     # pairwise_dist will store the distance from given input apparel to all remaining apparels
5     # the metric we used here is cosine, the coside distance is mesured as  $K(X, Y) = \langle X, Y \rangle / (\|X\| * \|Y\|)$ 
6     # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
7     pairwise_dist = pairwise_distances(tfidf_title_features,tfidf_title_features[doc_id])
8
9     # np.argsort will return indices of 9 smallest distances
10    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
11    pdists = np.sort(pairwise_dist.flatten())[0:num_results]
12
13    #data frame indices of the 9 smallest distane's
14    df_indices = list(data.index[indices])
15
16    for i in range(0,len(indices)):
17        # we will pass 1. doc_id, 2. title1, 3. title2, url, model
18        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')
19        print('ASIN : ',data['asin'].loc[df_indices[i]])
20        print('BRAND : ',data['brand'].loc[df_indices[i]])
21        print('Eucliden distance from the given image : ', pdists[i])
22        print('=*'*125)
23
24 tfidf_model(12500, 20)
25 # in the output heat map each value represents the tfidf values of the label word, the color represents the intersection with inputs title

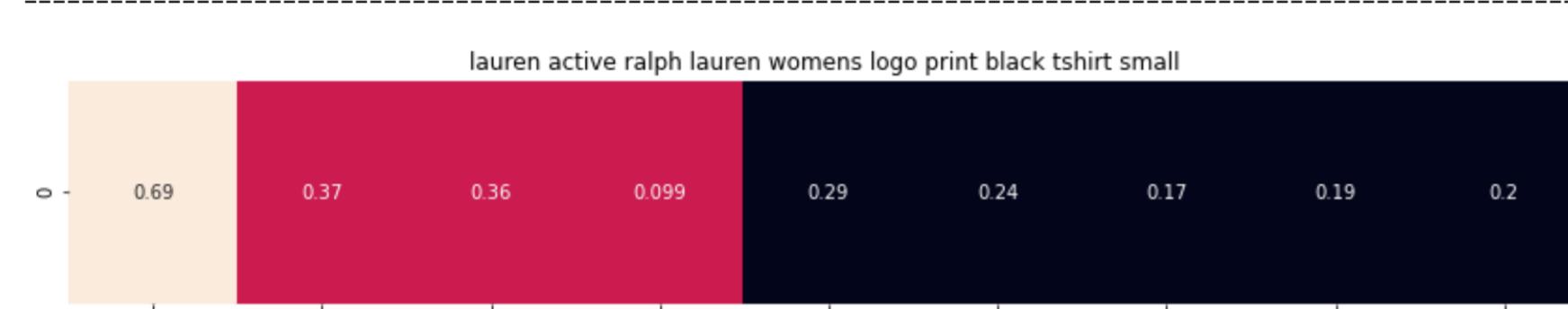
```



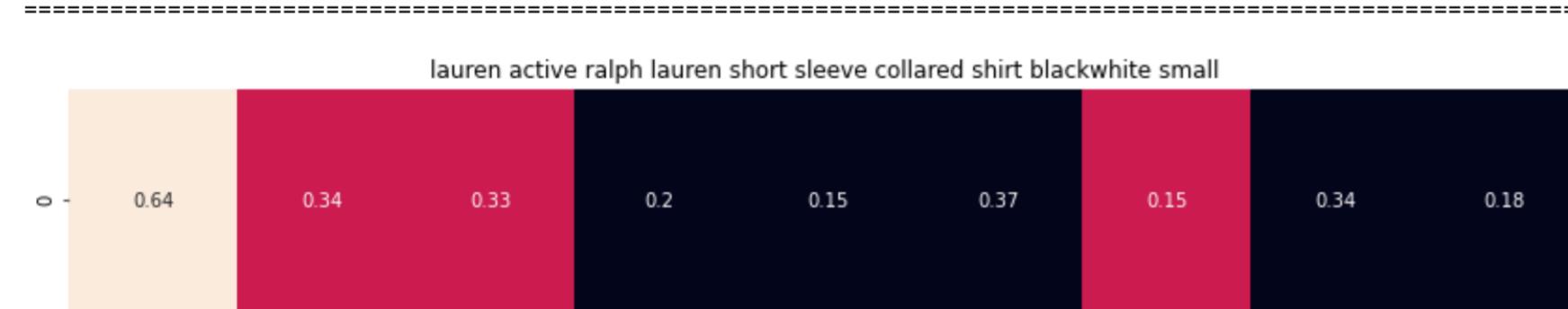
ASIN : B01JME4HGW  
BRAND : Ralph Lauren Active  
Eucliden distance from the given image : 0.0



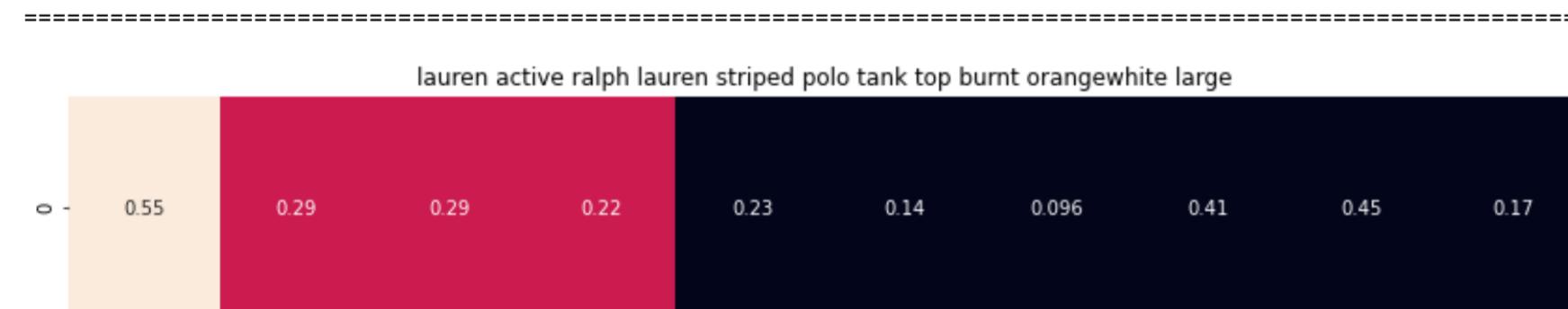
ASIN : B00ILGK6H2  
BRAND : Ralph Lauren Active  
Eucliden distance from the given image : 0.9320483484271197



ASIN : B01E1S2IGM  
BRAND : Ralph Lauren Active  
Eucliden distance from the given image : 0.9966336121056003



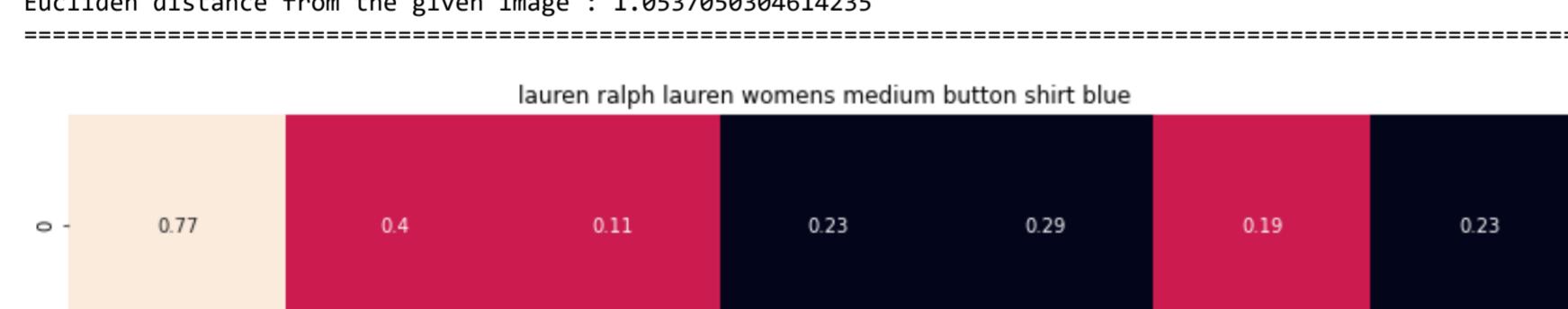
ASIN : B00HSX5ZAW  
BRAND : Ralph Lauren Active  
Eucliden distance from the given image : 1.0214919565354412



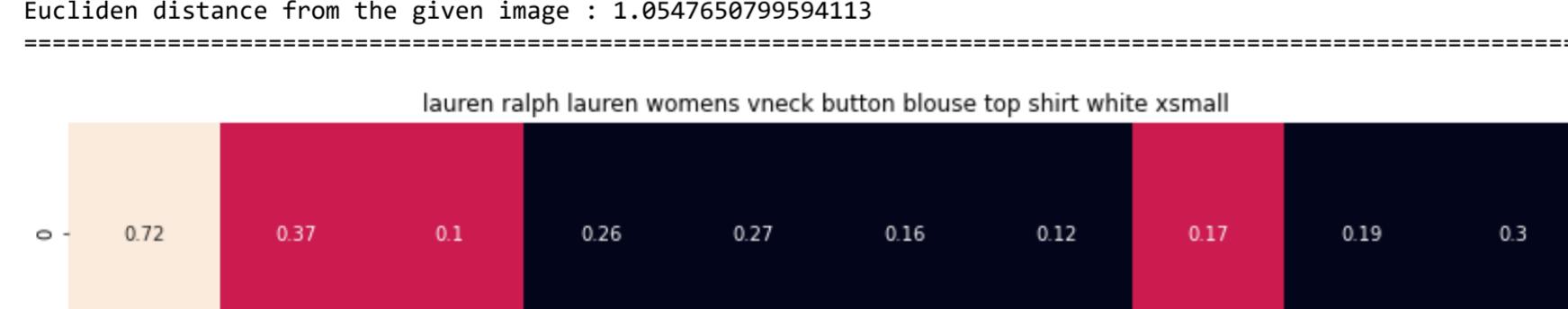
ASIN : B00ILGH5OY  
BRAND : Ralph Lauren Active  
Eucliden distance from the given image : 1.045589333445831



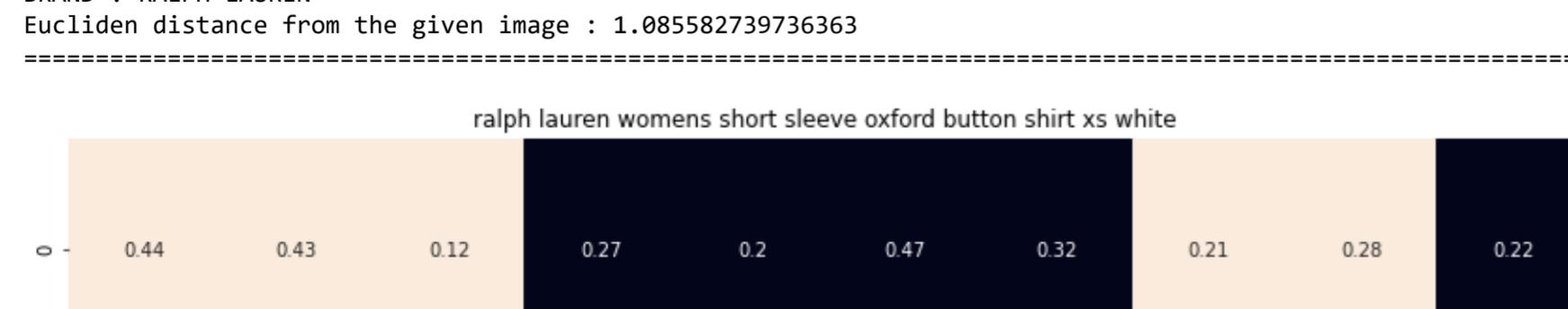
ASIN : B071GNR41  
BRAND : Lauren by Ralph Lauren  
Eucliden distance from the given image : 1.0537050304614235



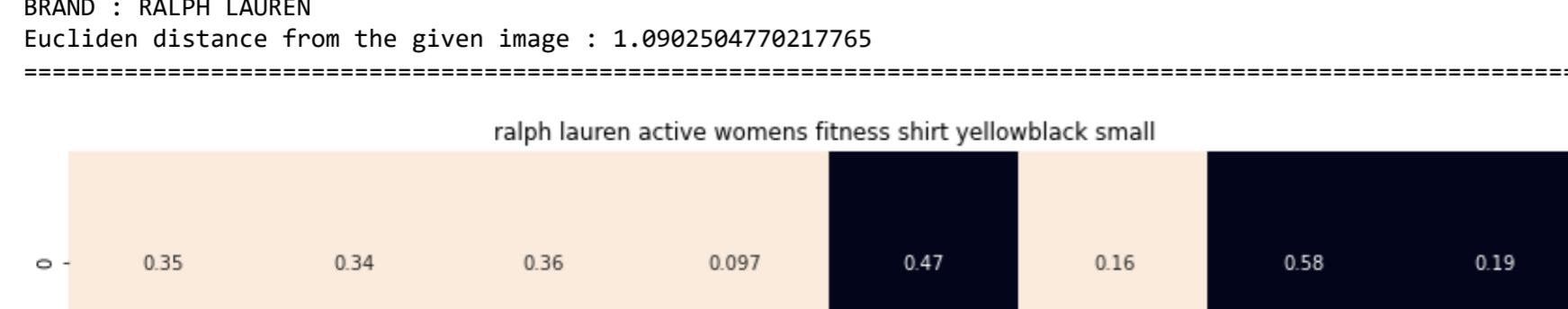
ASIN : B074ZLGXPL  
BRAND : Lauren by Ralph Lauren  
Eucliden distance from the given image : 1.0547650799594113



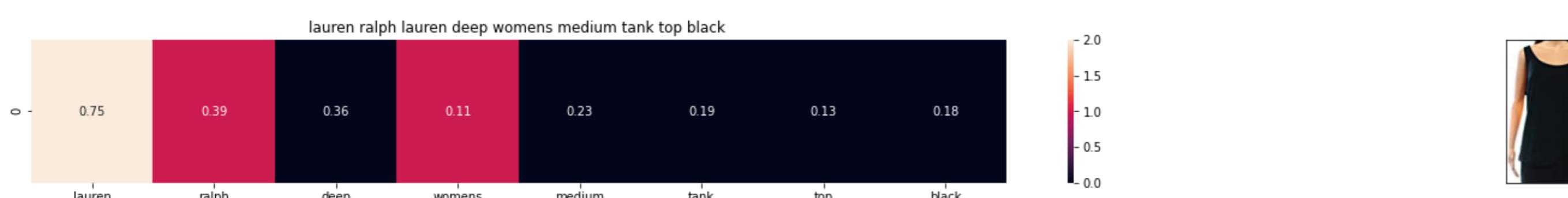
ASIN : B0713MCCFX  
BRAND : RALPH LAUREN  
Eucliden distance from the given image : 1.085582739736363



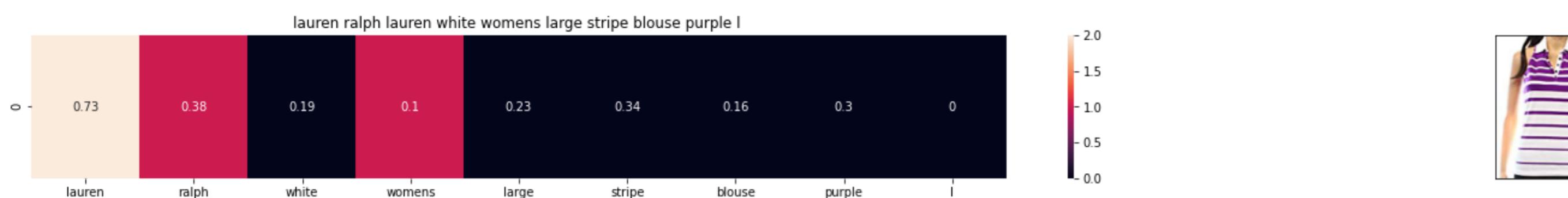
ASIN : B01FRIMR5U  
BRAND : RALPH LAUREN  
Eucliden distance from the given image : 1.0902504770217765



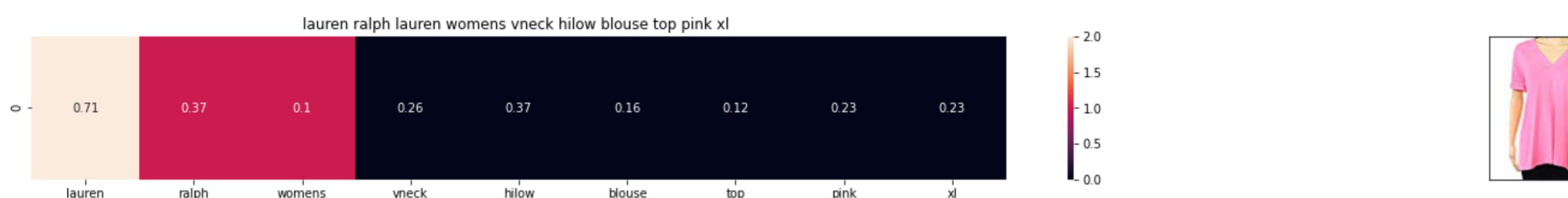
ASIN : B073WGPVWV  
 BRAND : Ralph Lauren Active  
 Euclidean distance from the given image : 1.0910155531611962



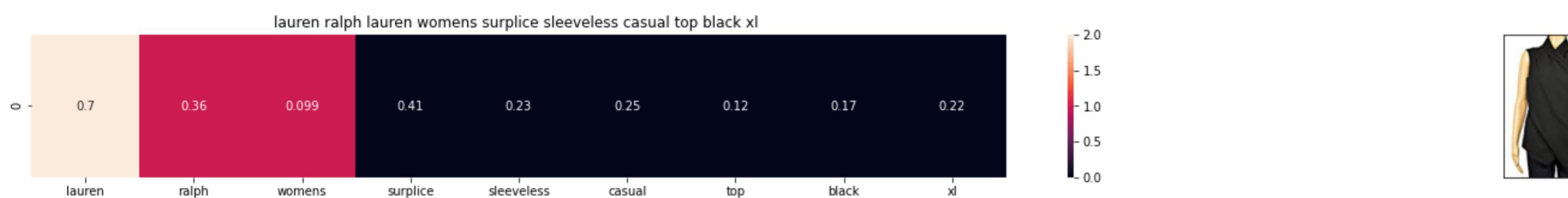
ASIN : B071P2NNYV  
 BRAND : Lauren by Ralph Lauren  
 Euclidean distance from the given image : 1.0935678614530933



ASIN : B0721MX9G5  
 BRAND : Lauren by Ralph Lauren  
 Euclidean distance from the given image : 1.1051456378086082



ASIN : B073RHR45N  
 BRAND : Lauren by Ralph Lauren  
 Euclidean distance from the given image : 1.1145413472144106



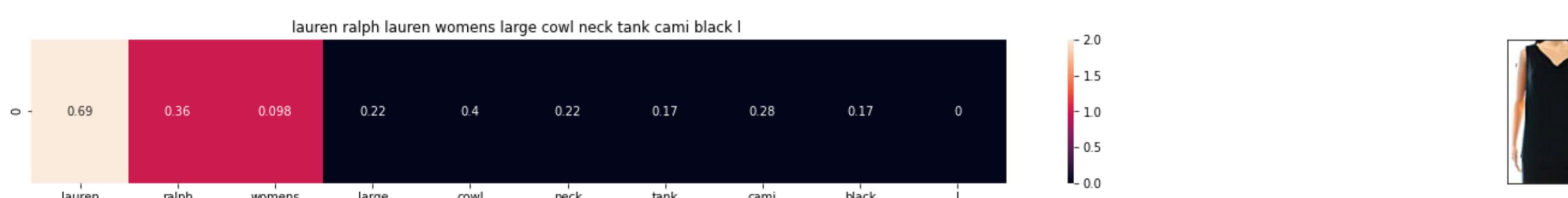
ASIN : B01N6RZPOW  
 BRAND : Lauren by Ralph Lauren  
 Euclidean distance from the given image : 1.1199457275970173



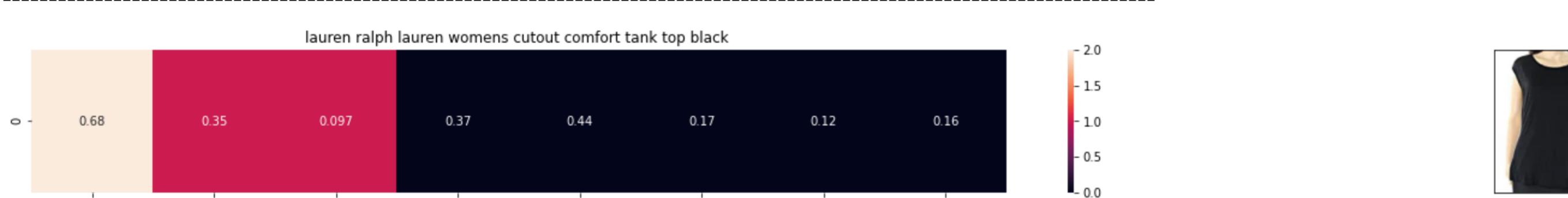
ASIN : B00CEMR4I8  
 BRAND : Polo Jeans Co.  
 Euclidean distance from the given image : 1.1206009276727187



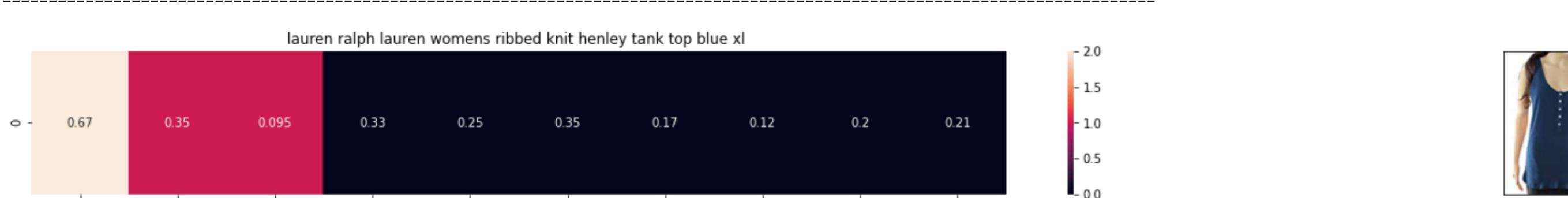
ASIN : B01MRL7J8F  
 BRAND : Polo Ralph Lauren  
 Euclidean distance from the given image : 1.1232830531545737



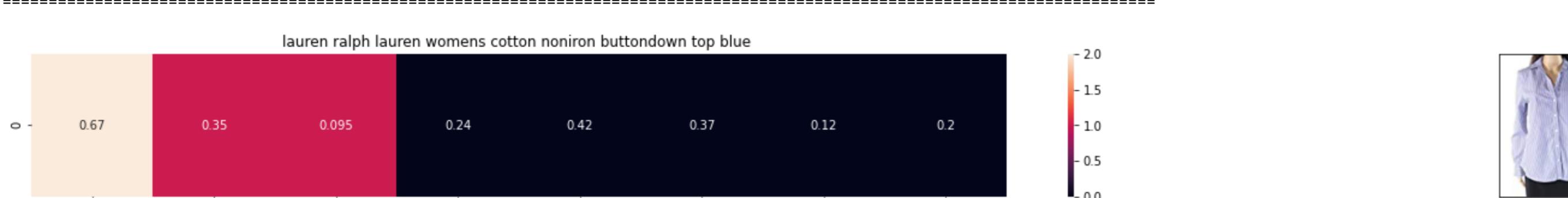
ASIN : B071YSRY3L  
 BRAND : Lauren by Ralph Lauren  
 Euclidean distance from the given image : 1.125300003432815



ASIN : B01N1K1YAO  
 BRAND : Lauren by Ralph Lauren  
 Euclidean distance from the given image : 1.127620677491475



ASIN : B0723511ZF  
 BRAND : Lauren by Ralph Lauren  
 Euclidean distance from the given image : 1.13298860836888



ASIN : B06XYMZRM4  
 BRAND : Lauren by Ralph Lauren  
 Euclidean distance from the given image : 1.1333503203455155

- Like BOW tfidf model also captures mostly brand and size from the title and still the hooded shirts are not captured well.

## [8.5] IDF based product similarity

In [42]:

```

1 idf_title_vectorizer = CountVectorizer()
2 idf_title_features = idf_title_vectorizer.fit_transform(data['title'])
3
4 # idf_title_features.shape = #data_points * #words_in_corpus
5 # CountVectorizer().fit_transform(corpus) returns a sparse matrix of dimensions #data_points * #words_in_corpus
6 # idf_title_features[doc_id, index_of_word_in_corpus] = number of times the word occurred in that doc

```

```
In [43]: M 1 def nContaining(word):  
2     # return the number of documents which had the given word  
3     return sum(1 for blob in data['title'] if word in blob.split())  
4  
5 def idf(word):  
6     # idf = log(#number of docs / #number of docs which had the given word)  
7     return math.log(data.shape[0] / (nContaining(word)))
```

```
In [44]: M 1 # we need to convert the values into float  
2 idf_title_features = idf_title_features.astype(np.float)  
3  
4 for i in idf_title_vectorizer.vocabulary_.keys():  
5     # for every word in whole corpus we will find its idf value  
6     idf_val = idf(i)  
7  
8     # to calculate idf_title_features we need to replace the count values with the idf values of the word  
9     # idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0] will return all documents in which the word i present  
10    for j in idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0]:  
11        # we replace the count values of word i in document j with idf_value of word i  
12        # idf_title_features[doc_id, index_of_word_in_corpus] = idf value of word  
13        idf_title_features[j,idf_title_vectorizer.vocabulary_[i]] = idf_val  
14  
15
```

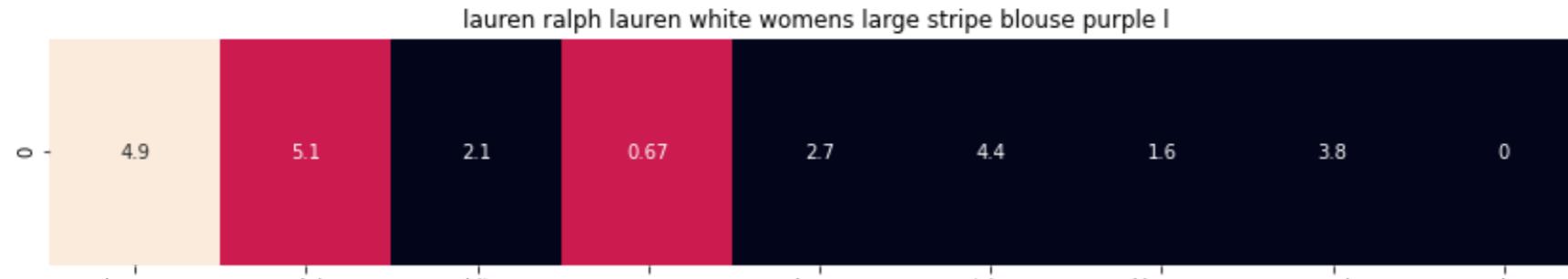
In [38]:

```

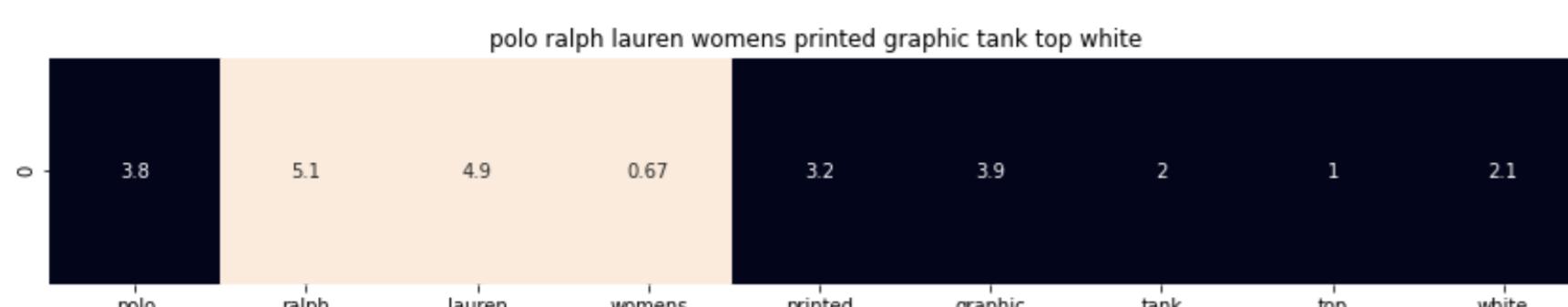
1 def idf_model(doc_id, num_results):
2     # doc_id: apparel's id in given corpus
3
4     # pairwise_dist will store the distance from given input apparel to all remaining apparels
5     # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y> / (||X|| * ||Y||)
6     # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
7     pairwise_dist = pairwise_distances(idf_title_features,idf_title_features[doc_id])
8
9     # np.argsort will return indices of 9 smallest distances
10    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
11    pdists = np.sort(pairwise_dist.flatten())[0:num_results]
12
13    #data frame indices of the 9 smallest distane's
14    df_indices = list(data.index[indices])
15
16    for i in range(0,len(indices)):
17        get_result(indices[i],data['title'].loc[df_indices[0]],data['title'].loc[df_indices[i]],'idf')
18        data['medium_image_url'].loc[df_indices[i]]['idt']
19        print('ASIN : ',data['asin'].loc[df_indices[i]])
20        print('Brand : ',data['brand'].loc[df_indices[i]])
21        print('euclidean distance from the given image : ', pdists[i])
22        print('*'*125)
23
24
25
26
27 idf_model(12500,20)
28 # in the output heat map each value represents the idf values of the label word, the color represents the intersection with inputs title

```

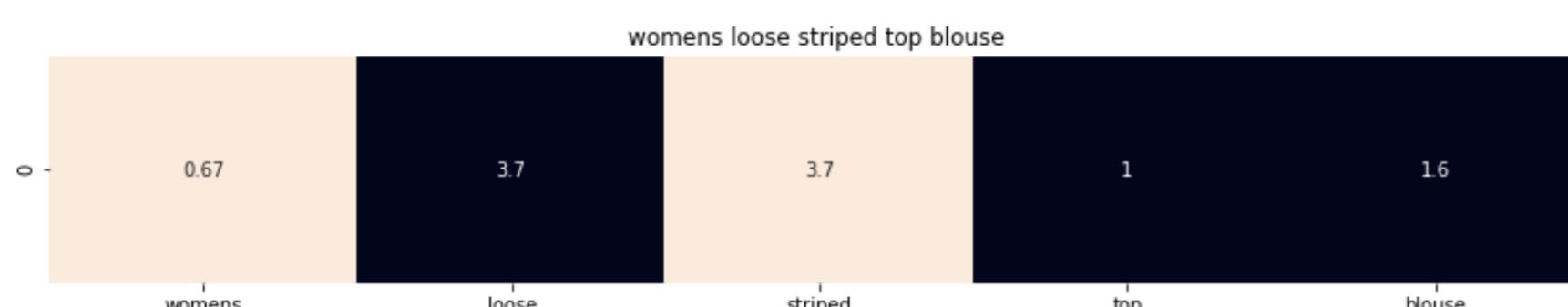




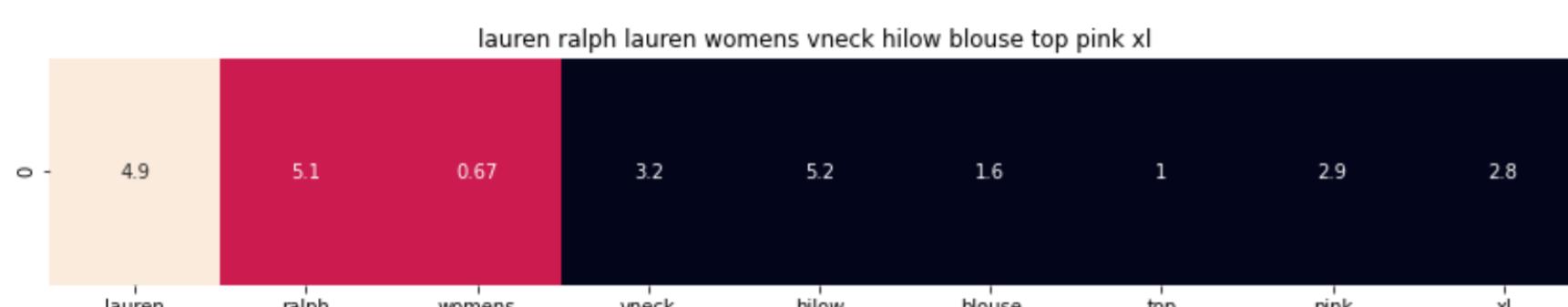
ASIN : B0721MX9GS  
Brand : Lauren by Ralph Lauren  
euclidean distance from the given image : 14.555043023530818



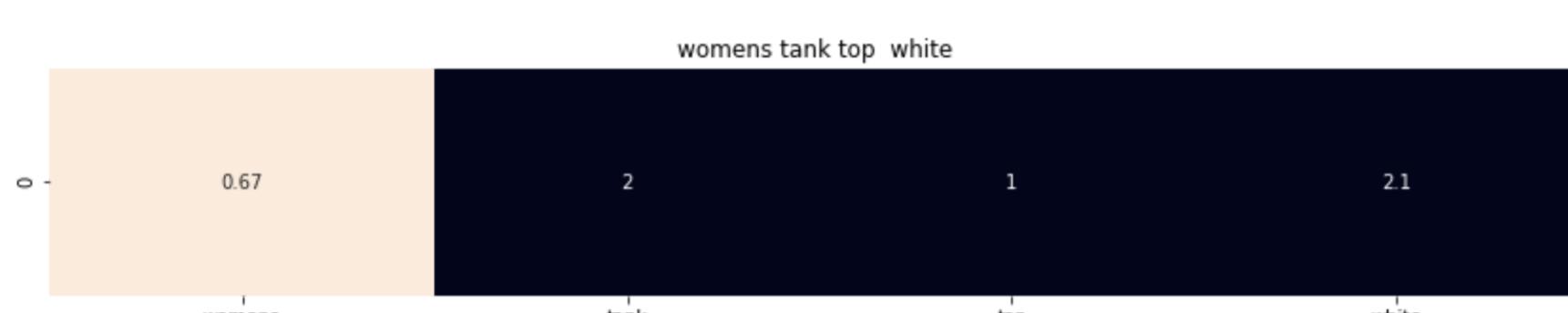
ASIN : B01NAP3SFK  
Brand : Polo Ralph Lauren  
euclidean distance from the given image : 14.596981500660664



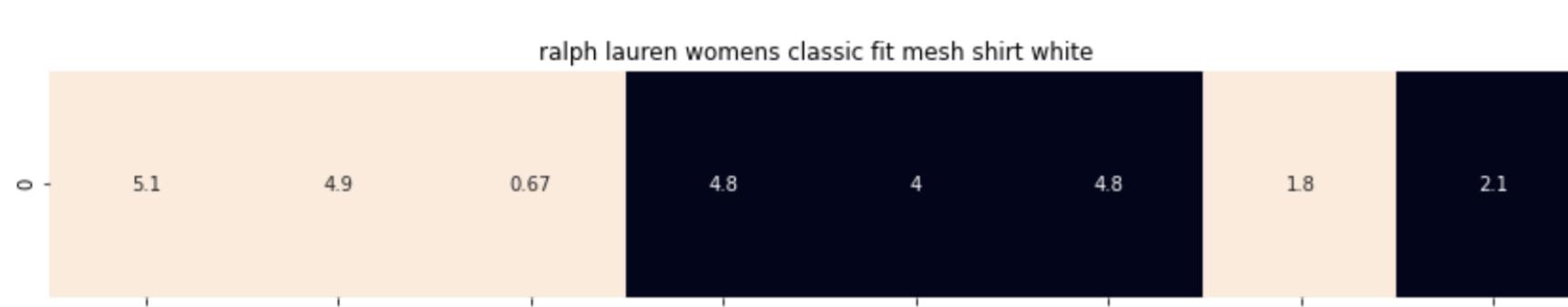
ASIN : B00ZZMYBRG  
Brand : HP-LEISURE  
euclidean distance from the given image : 14.745687999762726



ASIN : B073RHR45N  
Brand : Lauren by Ralph Lauren  
euclidean distance from the given image : 14.86907825286474



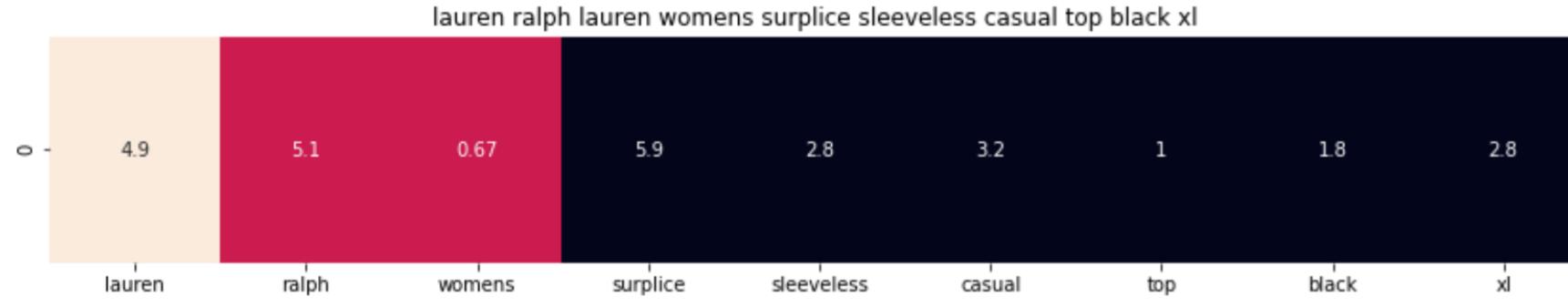
ASIN : B00JPOZ9GM  
Brand : Sofra  
euclidean distance from the given image : 14.926585124479391



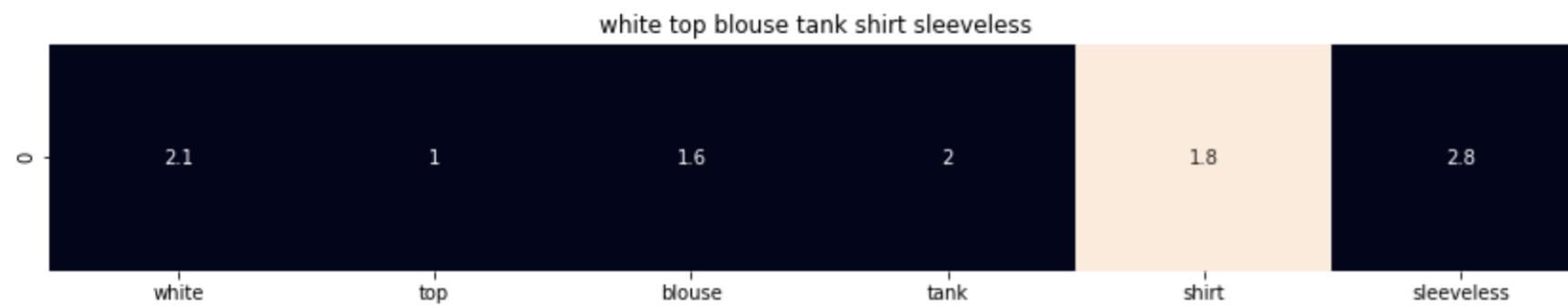
ASIN : B01KIXF87S  
Brand : RALPH LAUREN  
euclidean distance from the given image : 15.062503177033417



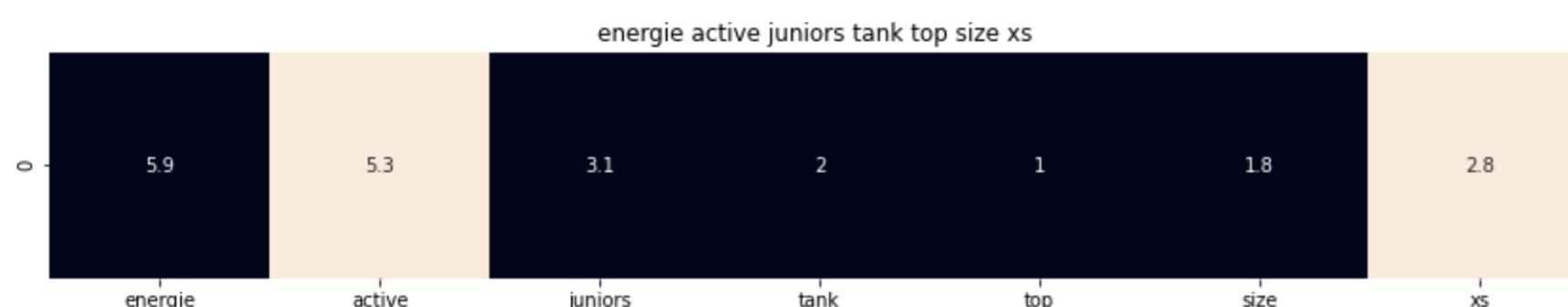
ASIN : B01KM2OLIW  
Brand : Voguegirl  
euclidean distance from the given image : 15.073128747546114



ASIN : B01NGRZPQW  
Brand : Lauren by Ralph Lauren  
euclidean distance from the given image : 15.122329251244038



ASIN : B074G5G5RK  
Brand : ERMANNO SCERVINO  
euclidean distance from the given image : 15.182130236391776



ASIN : B071RSQFKQ  
Brand : Energie  
euclidean distance from the given image : 15.205561495747867



ASIN : B00KF2N5PU  
Brand : Vietsbay  
euclidean distance from the given image : 15.208661081568088

## [9] Text Semantics based product similarity

```
In [35]: M
1 from gensim.models import Word2Vec
2 from gensim.models import KeyedVectors
3 import pickle
4
5 # in this project we are using a pretrained model by google
6 # its 3.3G file, once you load this into your memory
7 # it occupies ~9GB, so please do this step only if you have >12G of ram
8 # we will provide a pickle file which contains a dict ,
9 # and it contains all our corpus words as keys and model[word] as values
10 # To use this code-snippet, download "GoogleNews-vectors-negative300.bin"
11 # from https://drive.google.com/file/d/0B7XkCwPI5KDYNLNUUTLSS2lpQmM/edit
12 # it's 1.9GB in size.
13
14 ...
15 model = KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=True)
16 ...
17
18 #if you do NOT have RAM >= 12GB, use the code below.
19 with open('word2vec_model', 'rb') as handle:
20     model = pickle.load(handle)
21
```

```
In [36]: M
1 # Utility functions
2
3 def get_word_vec(sentence, doc_id, m_name):
4     # sentence : title of the apparel
5     # doc_id: document id in our corpus
6     # m_name: model information it will take two values
7     # if m_name == 'avg', we will append the model[i], w2v representation of word i
8     # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)
9     vec = []
10    for i in sentence.split():
11        if i in vocab:
12            if m_name == 'weighted' and i in idf_title_vectorizer.vocabulary_:
13                vec.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[i]] * model[i])
14            elif m_name == 'avg':
15                vec.append(model[i])
16        else:
17            # if the word in our corpus is not there in the google word2vec corpus, we are just ignoring it
18            vec.append(np.zeros(shape=(300,)))
19    # we will return a numpy array of shape (#number of words in title * 300) 300 = Len(w2v_model[word])
20    # each row represents the word2vec representation of each word (weighted/avg) in given sentence
21    return np.array(vec)
22
23 def get_distance(vec1, vec2):
24     # vec1 = np.array(#number_of_words_title1 * 300), each row is a vector of length 300 corresponds to each word in give title
25     # vec2 = np.array(#number_of_words_title2 * 300), each row is a vector of length 300 corresponds to each word in give title
26
27     final_dist = []
28     # for each vector in vec1 we calculate the distance(euclidean) to all vectors in vec2
29     for i in vec1:
30         dist = []
31         for j in vec2:
32             # np.linalg.norm(i-j) will result the euclidean distance between vectors i, j
33             dist.append(np.linalg.norm(i-j))
34         final_dist.append(np.array(dist))
35     # final_dist = np.array(#number of words in title1 * #number of words in title2)
36     # final_dist[i,j] = euclidean distance between vectors i, j
37     return np.array(final_dist)
38
39
40 def heat_map_w2v(sentence1, sentence2, url, doc_id1, doc_id2, model):
41     # sentence1 : title1, input apparel
42     # sentence2 : title2, recommended apparel
43     # url: apparel image url
44     # doc_id1: document id of input apparel
45     # doc_id2: document id of recommended apparel
46     # model: it can have two values, 1. avg 2. weighted
47
48     s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/avg) of Length 300 corresponds to each word in give title
49     s1_vec = get_word_vec(sentence1, doc_id1, model)
50     s2_vec = np.array(#number_of_words_title2 * 300), each row is a vector(weighted/avg) of Length 300 corresponds to each word in give title
51     s2_vec = get_word_vec(sentence2, doc_id2, model)
52
53     # s1_s2_dist = np.array(#number of words in title1 * #number of words in title2)
54     # s1_s2_dist[i,j] = euclidean distance between words i, j
55     s1_s2_dist = get_distance(s1_vec, s2_vec)
56
57
58     # devide whole figure into 2 parts 1st part displays heatmap 2nd part displays image of apparel
59     gs = gridspec.GridSpec(2, 2, width_ratios=[4,1], height_ratios=[2,1])
60     fig = plt.figure(figsize=(15,15))
61
62     ax = plt.subplot(gs[0])
63     # plotting the heatmap based on the pairwise distances
64     ax = sns.heatmap(np.round(s1_s2_dist,4), annot=True)
65     # set the x axis labels as recommended apparels title
66     # set the y axis labels as input apparels title
67     try :
68         # np.concatenate((sentence2.split(),[sentence2.split()[0]]))
69         # np.concatenate((sentence1.split(),[sentence1.split()[0]]))
70         ax.set_xticklabels(sentence2.split())
71         # set the y axis Labels as input apparels title
72         ax.set_yticklabels(sentence1.split())
73         # set title as recommended apparels title
74         ax.set_title(sentence2)
75     except:
76         ax.set_xticklabels(np.concatenate((sentence2.split(),[sentence2.split()[0]])))
77         # set the y axis labels as input apparels title
78         ax.set_yticklabels(np.concatenate((sentence1.split(),[sentence1.split()[0]])))
79         # set title as recommended apparels title
80         ax.set_title(sentence2)
81
82     ax = plt.subplot(gs[1])
83     # we remove all grids and axis labels for image
84     ax.grid(False)
85     ax.set_xticks([])
86     ax.set_yticks([])
87     display_img(url, ax, fig)
88
89 plt.show()
```

```
In [37]: M
1 # vocab stores all the words that are there in google w2v model
2 # vocab = model.wv.vocab.keys() # if you are using Google word2Vec
3
4 vocab = model.keys()
5 # this function will add the vectors of each word and returns the avg vector of given sentence
6 def build_avg_vec(sentence, num_features, doc_id, m_name):
7     # sentence: its title of the apparel
8     # num_features: the length of word2vec vector, its values = 300
9     # m_name: model information it will take two values
10    # if m_name == 'avg', we will append the model[i], w2v representation of word i
11    # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)
12
13    featureVec = np.zeros((num_features,), dtype="float32")
14    # we will initialize a vector of size 300 with all zeros
15    # we add each word2vec(wordi) to this featureVec
16    nwords = 0
17
18    for word in sentence.split():
19        nwords += 1
20        if word in vocab:
21            if m_name == 'weighted' and word in idf_title_vectorizer.vocabulary_:
22                featureVec = np.add(featureVec, idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[word]] * model[word])
23            elif m_name == 'avg':
24                featureVec = np.add(featureVec, model[word])
25    if(nwords>0):
26        featureVec = np.divide(featureVec, nwords)
27    # returns the avg vector of given sentence, its of shape (1, 300)
28    return featureVec
```

## [9.2] Average Word2Vec product similarity.

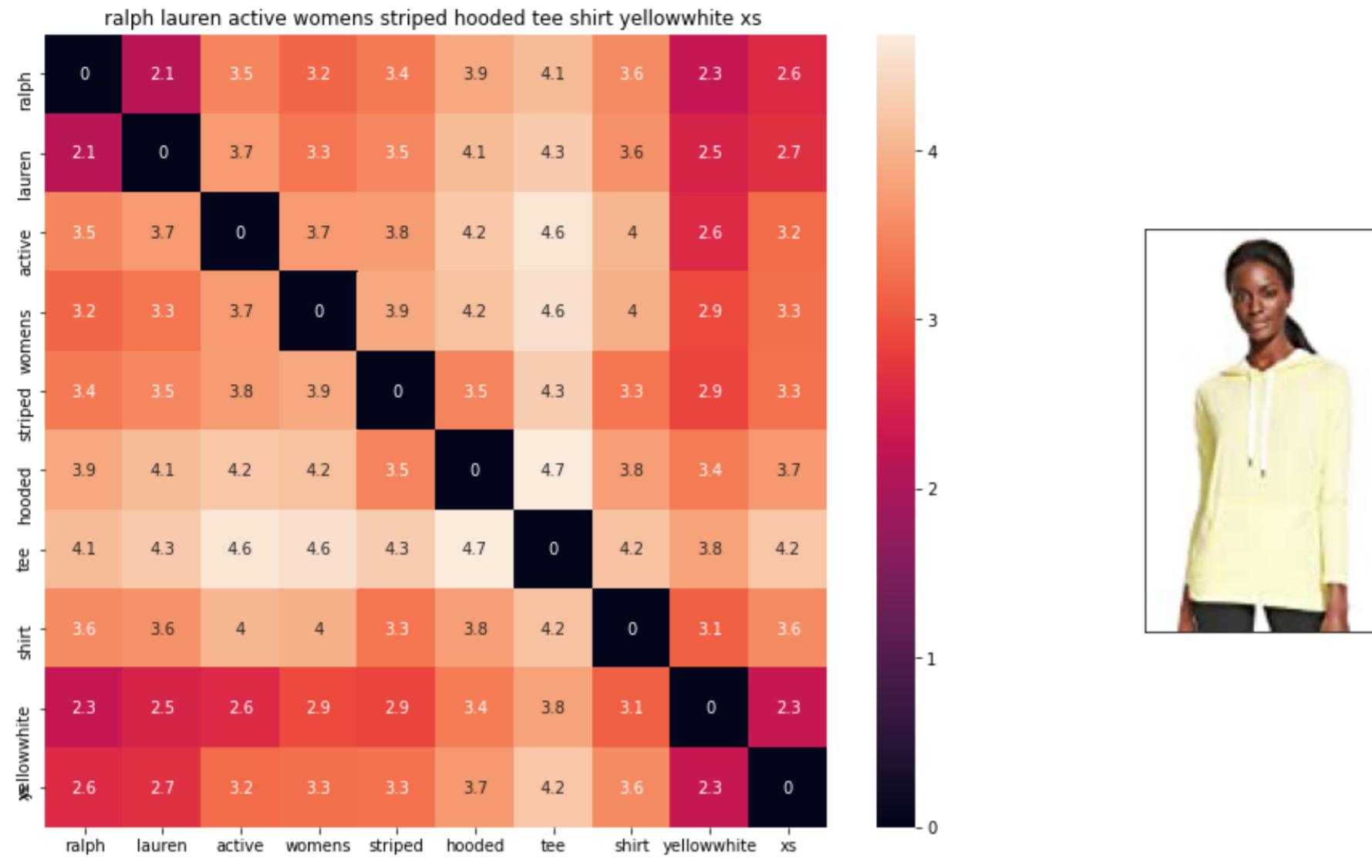
```
In [38]: M
1 doc_id = 0
2 w2v_title = []
3 # for every title we build a avg vector representation
4 for i in data['title']:
5     w2v_title.append(build_avg_vec(i, 300, doc_id, 'avg'))
6     doc_id += 1
7
8 # w2v_title = np.array(# number of doc in courpus * 300), each row corresponds to a doc
9 w2v_title = np.array(w2v_title)
```

In [62]:

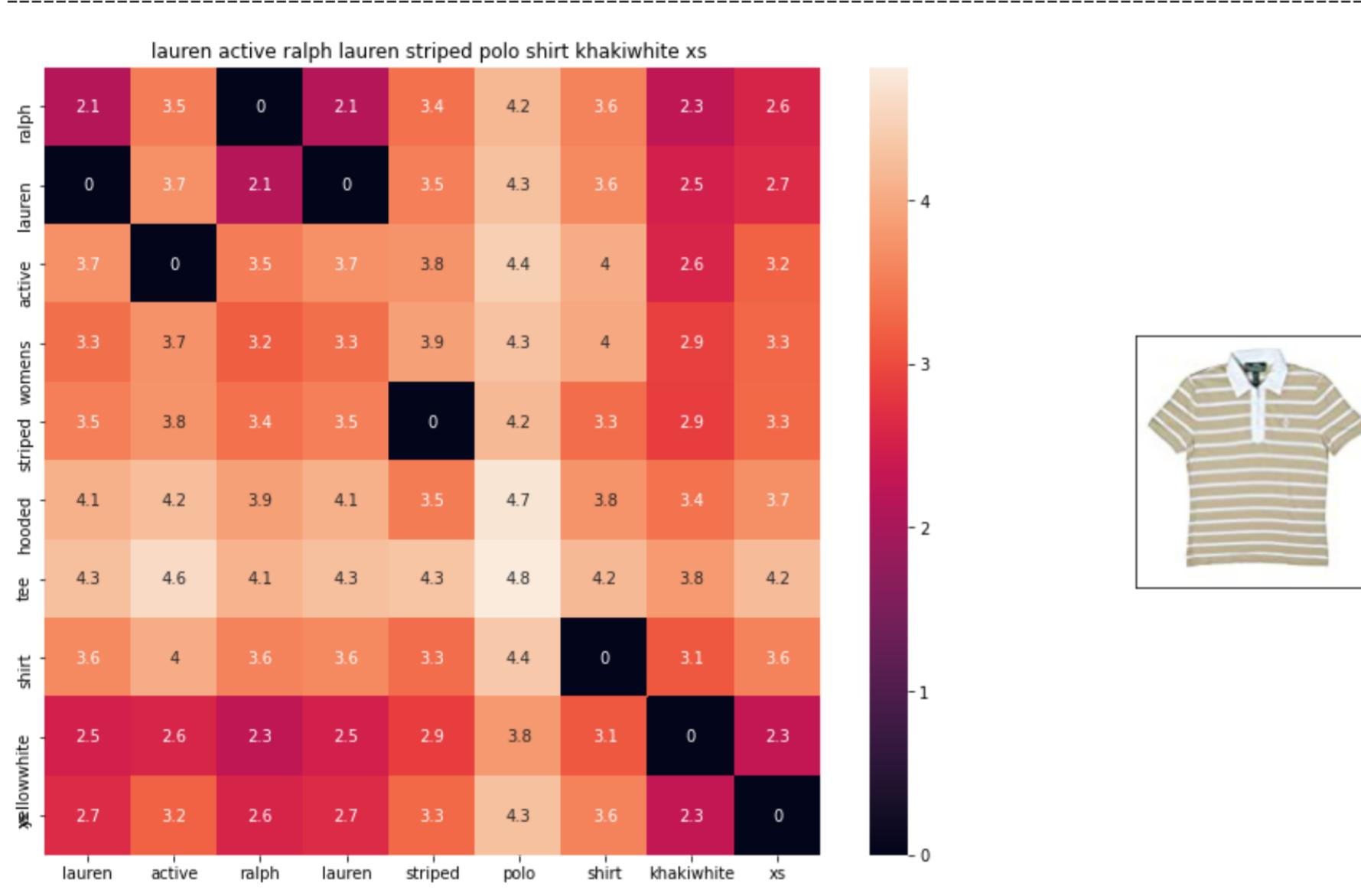
```

1 def avg_w2v_model(doc_id, num_results):
2     # doc_id: apparel's id in given corpus
3
4     # dist(x, y) = sqrt(dot(x, x) - 2 * dot(x, y) + dot(y, y))
5     pairwise_dist = pairwise_distances(w2v_title, w2v_title[doc_id].reshape(1,-1))
6
7     # np.argsort will return indices of 9 smallest distances
8     indices = np.argsort(pairwise_dist.flatten())[0:num_results]
9     #pdists will store the 9 smallest distances
10    pdists = np.sort(pairwise_dist.flatten())[0:num_results]
11
12    #data frame indices of the 9 smallest distance's
13    df_indices = list(data.index[indices])
14
15    for i in range(0, len(indices)):
16        heat_map_w2v(data['title'].loc[df_indices[i]],data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i], 'avg')
17        print('ASIN :',data['asin'].loc[df_indices[i]])
18        print('BRAND :',data['brand'].loc[df_indices[i]])
19        print ('euclidean distance from given input image :', pdists[i])
20        print('*'*125)
21
22
23 avg_w2v_model(12500, 20)
24 # in the give heat map, each cell contains the euclidean distance between words i, j

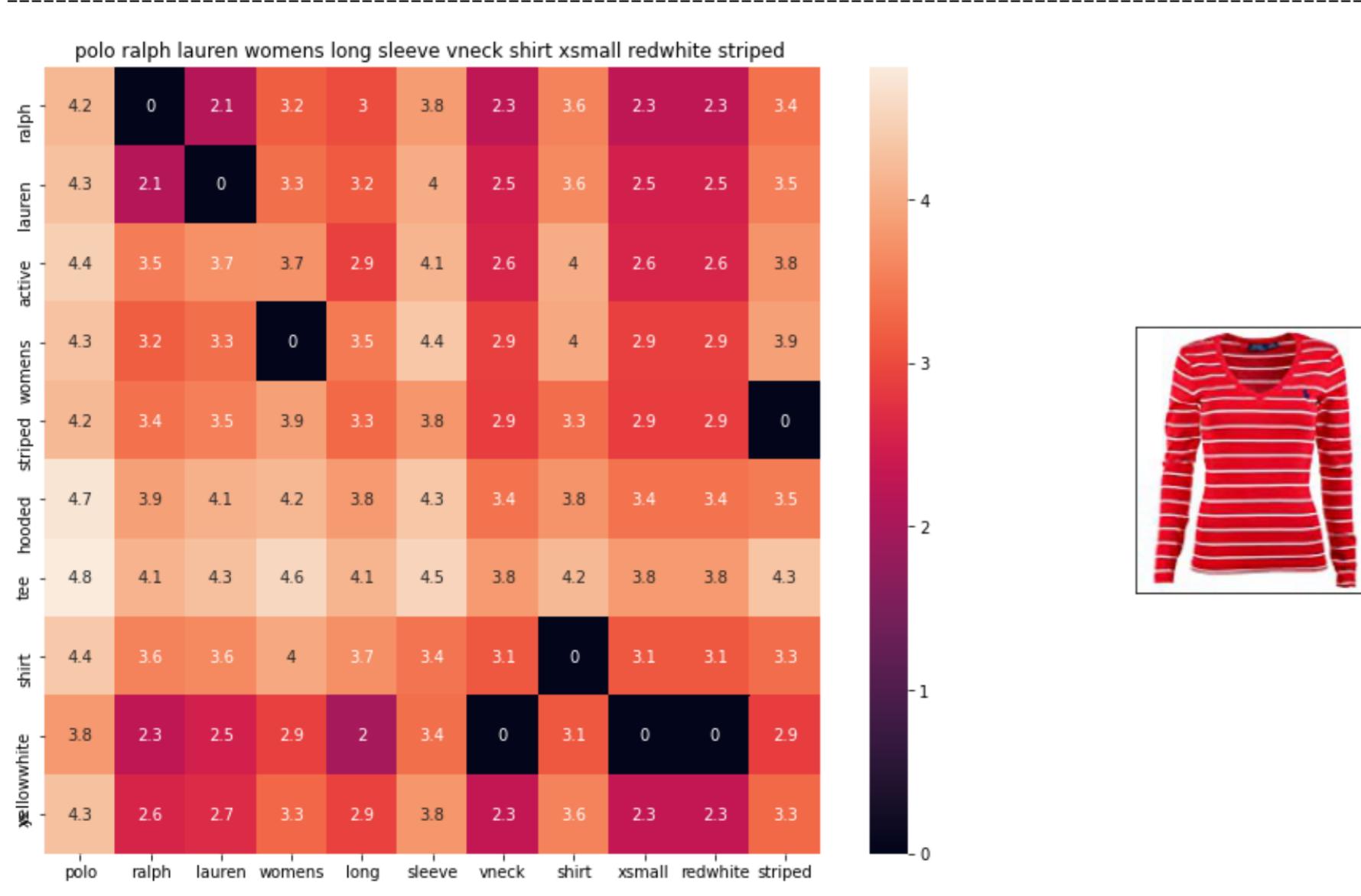
```



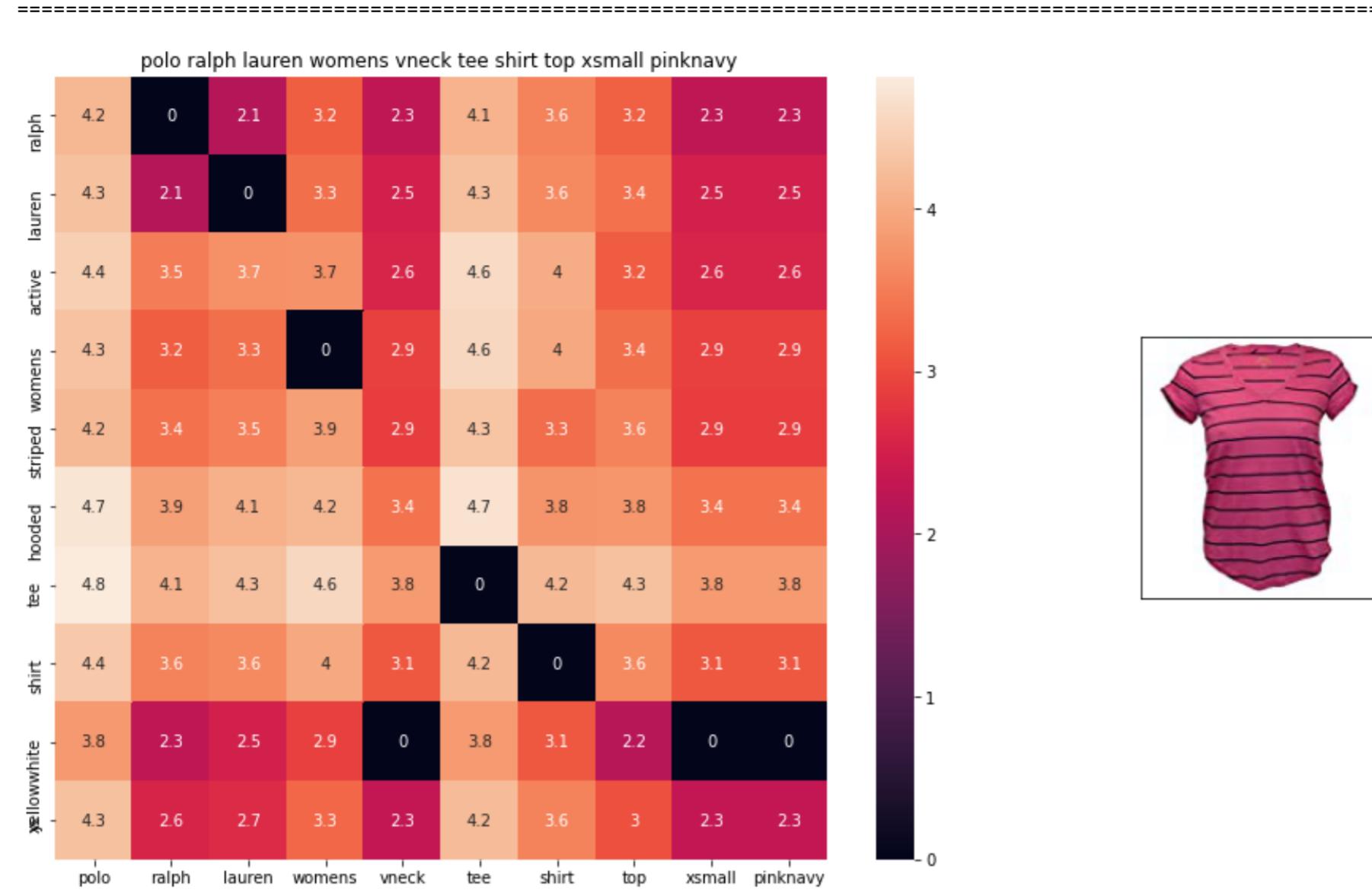
ASIN : B01JME4H6W  
BRAND : Ralph Lauren Active  
euclidean distance from given input image : 0.0



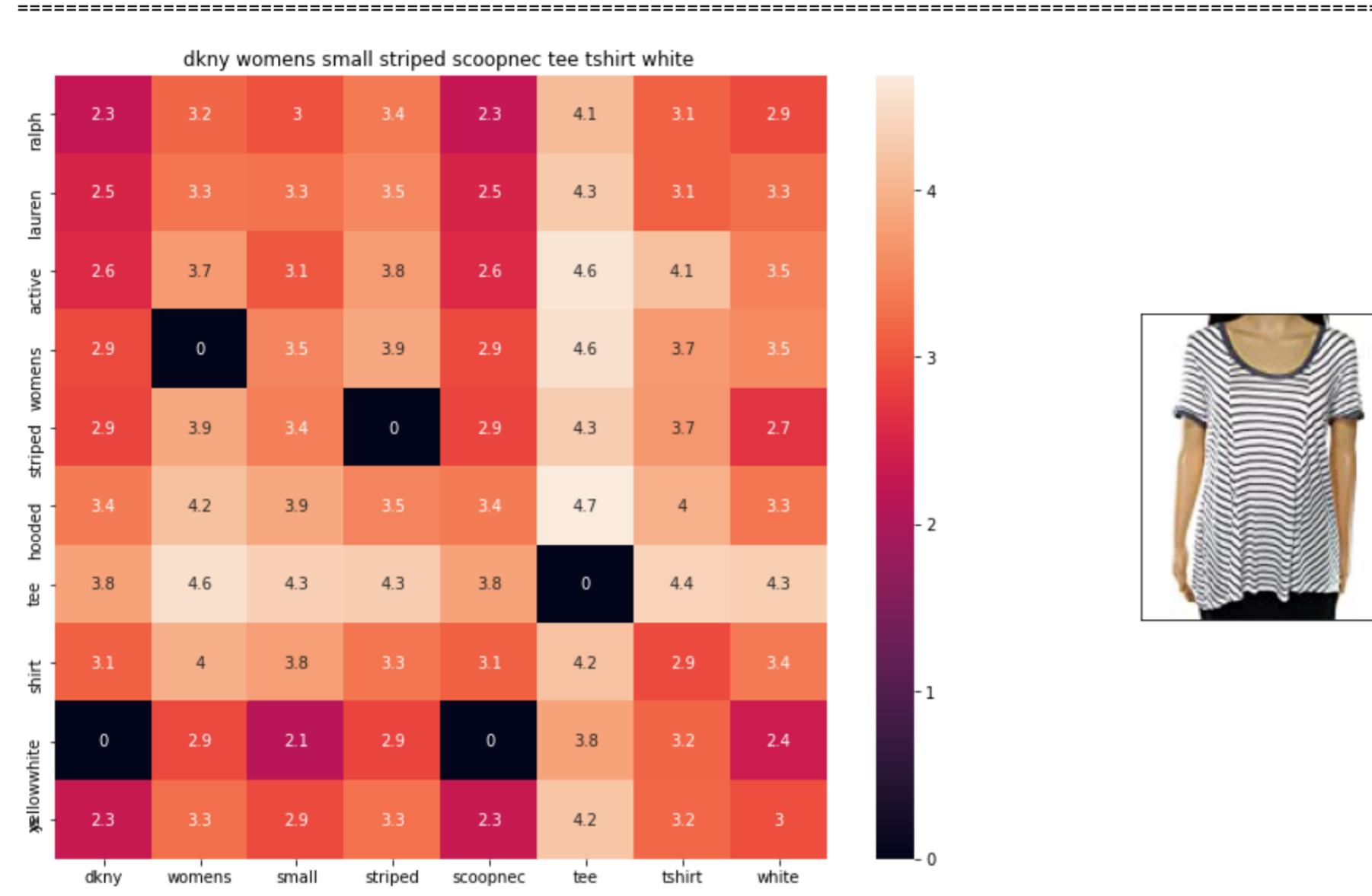
ASIN : B00ILGK6H2  
BRAND : Ralph Lauren Active  
euclidean distance from given input image : 0.70010304



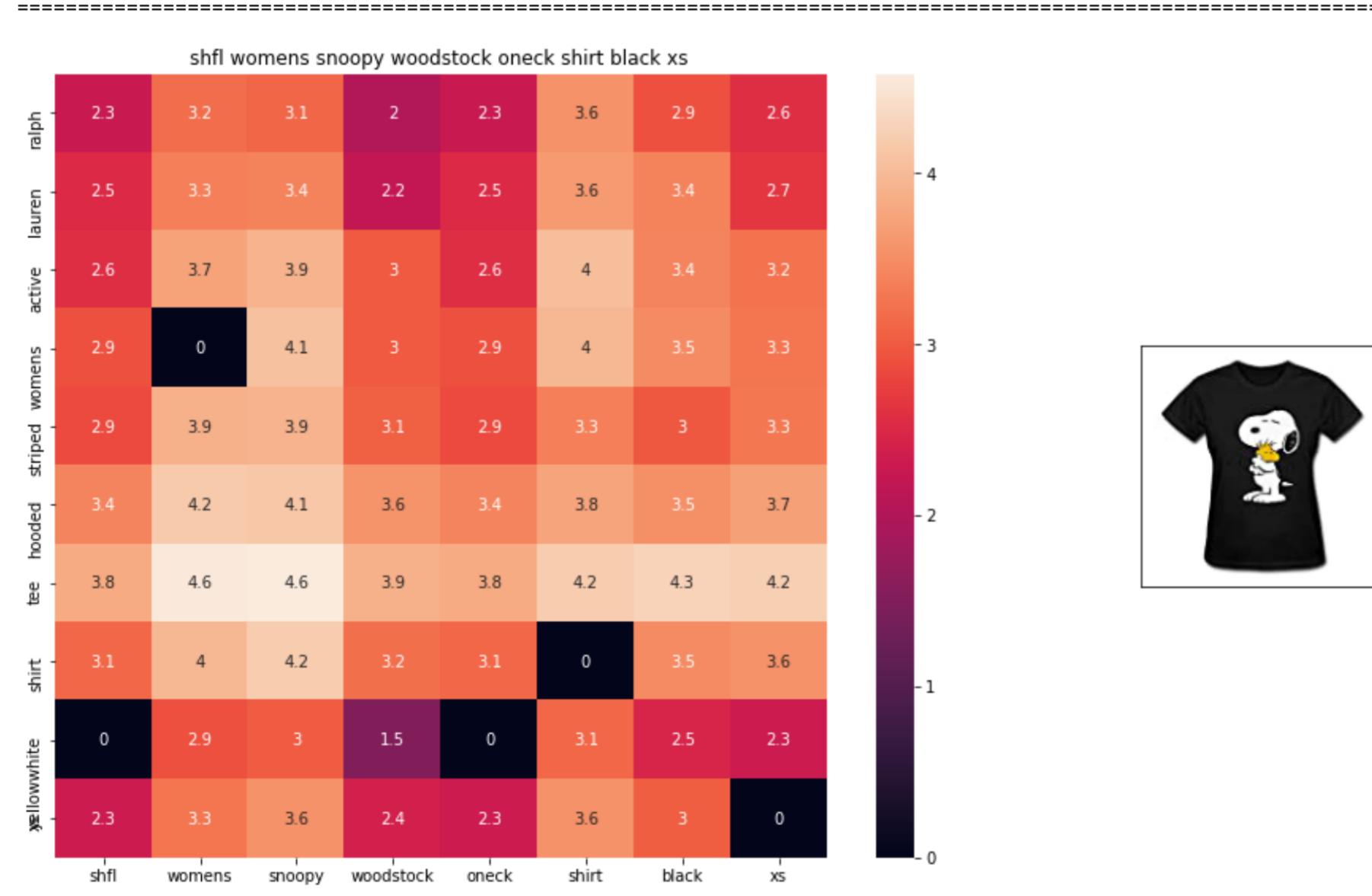
ASIN : B01MRL7J8F  
 BRAND : Polo Ralph Lauren  
 euclidean distance from given input image : 0.7125932



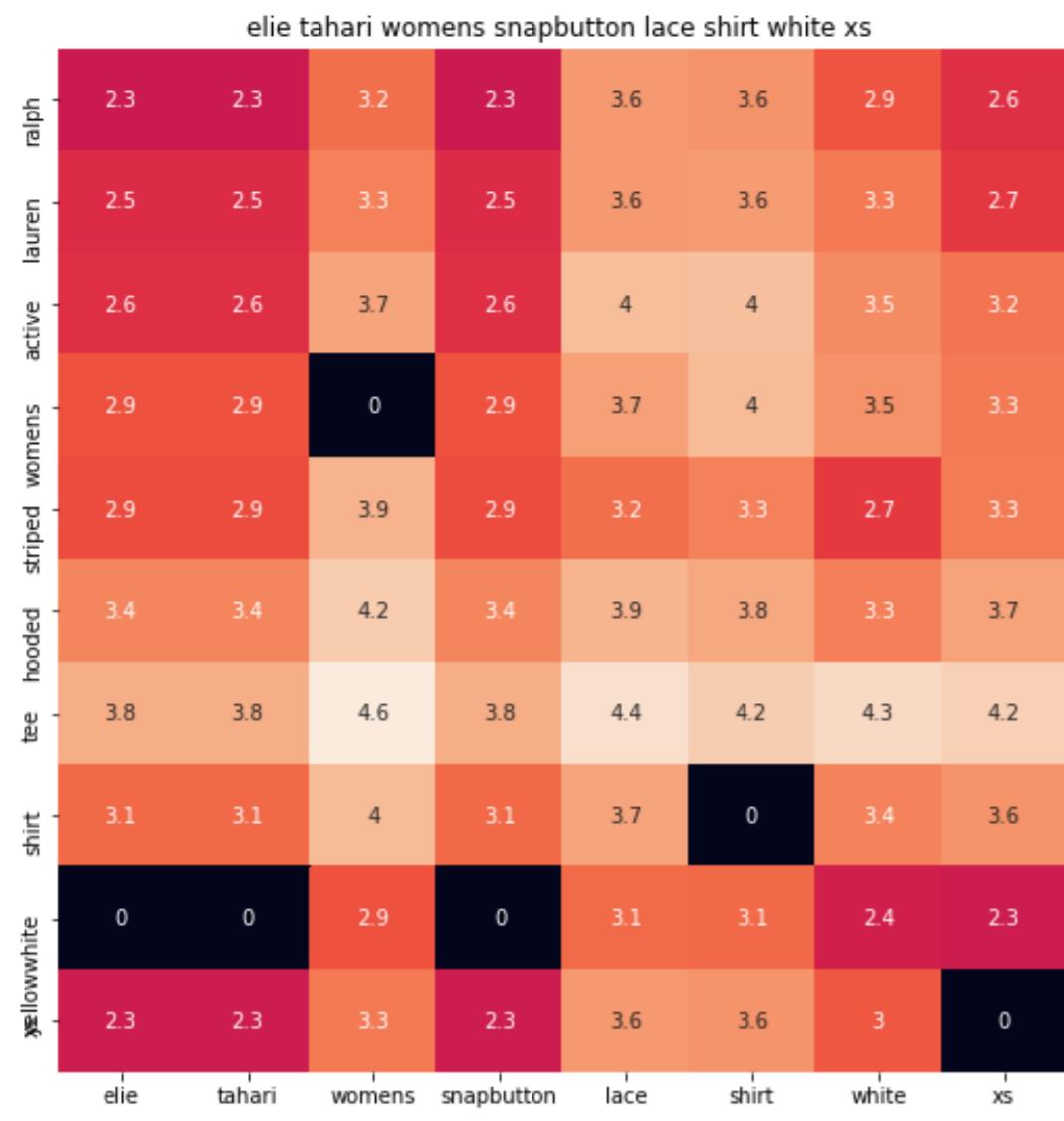
ASIN : B01ETUBK3W  
 BRAND : RALPH LAUREN  
 euclidean distance from given input image : 0.7143427



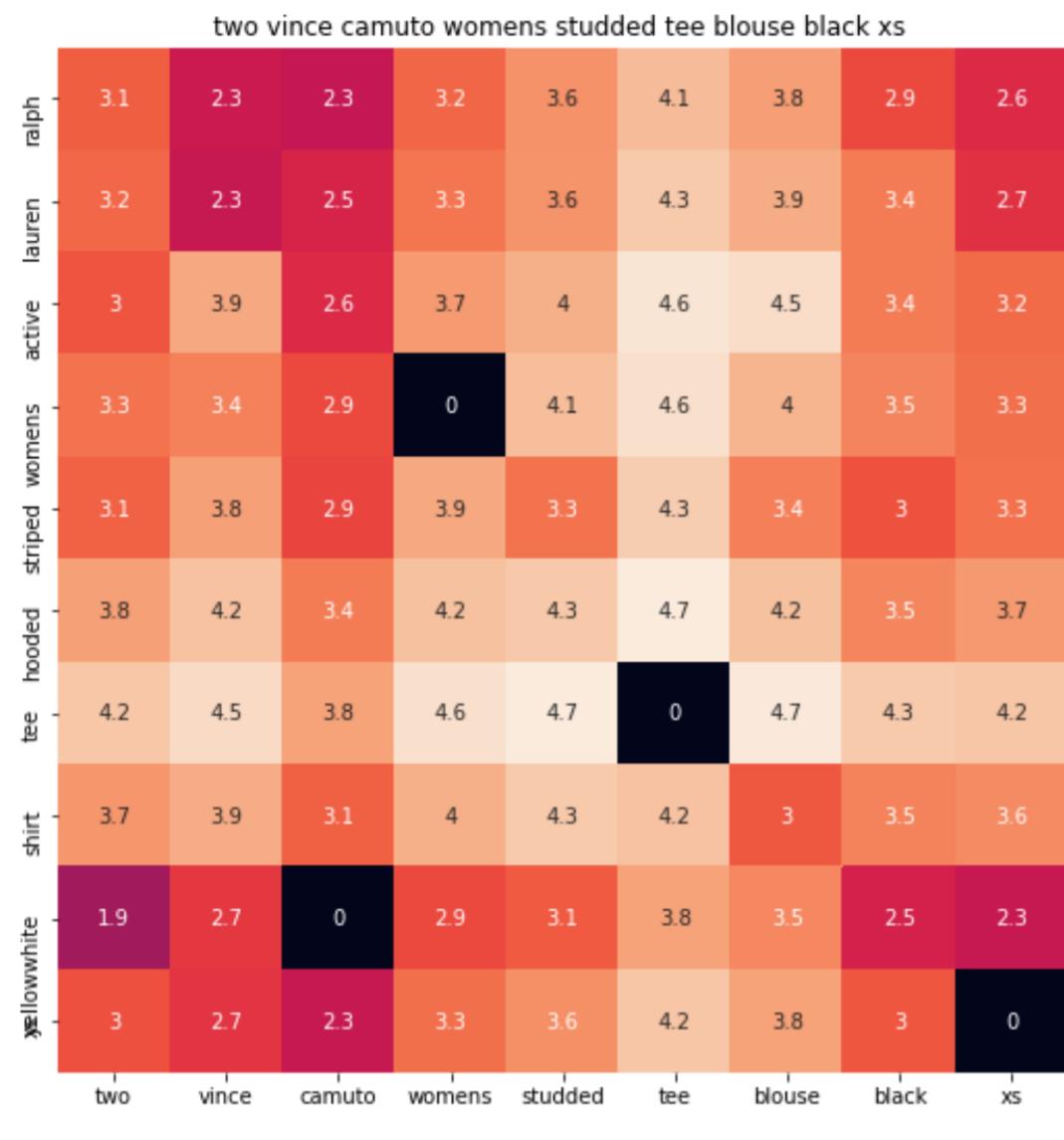
ASIN : B0725PWQFW  
 BRAND : DKNY  
 euclidean distance from given input image : 0.7170714



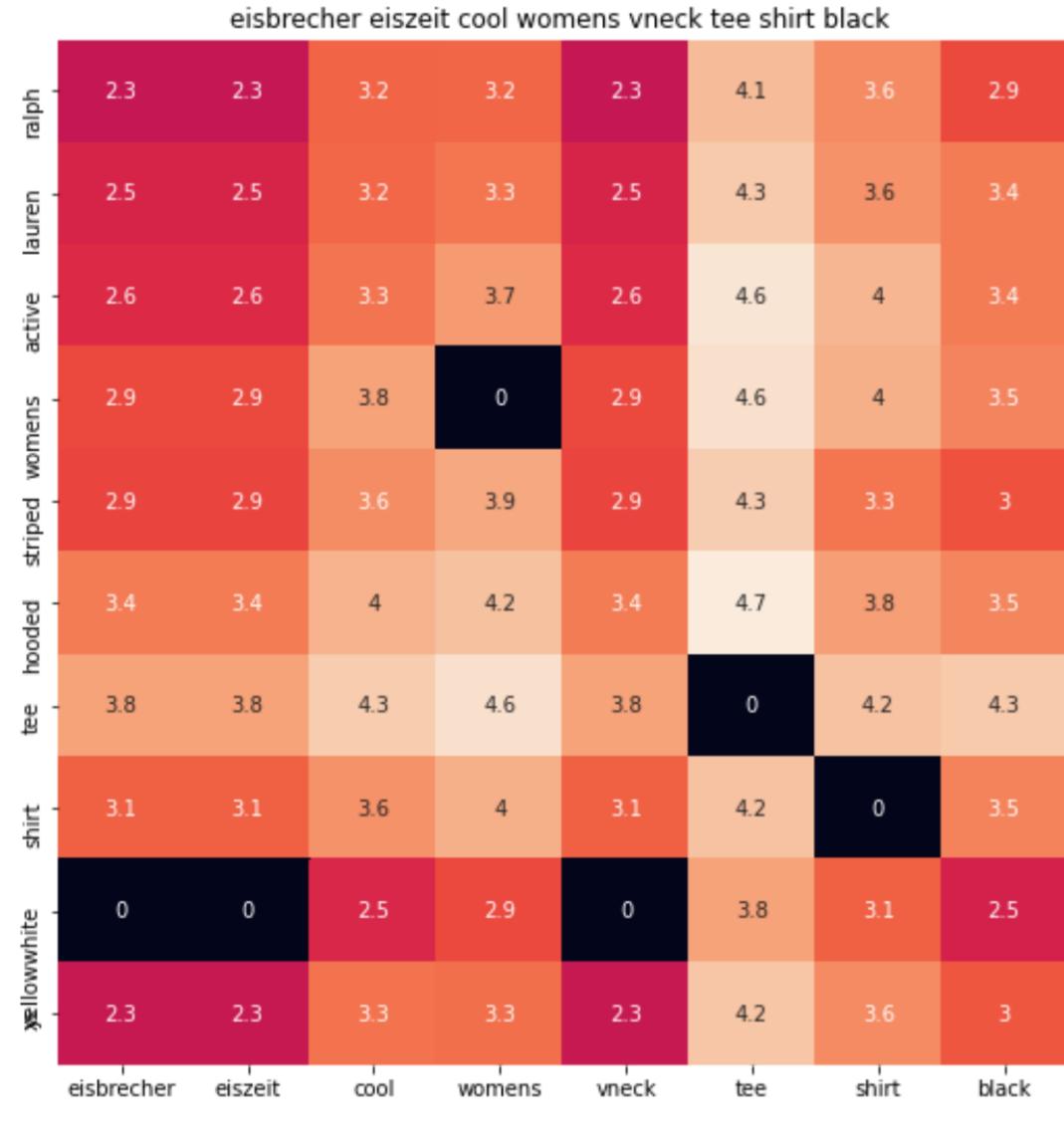
ASIN : B0166856BW  
 BRAND : SHFL  
 euclidean distance from given input image : 0.7533704



ASIN : B06X8YS632  
BRAND : Elie Tahari  
euclidean distance from given input image : 0.7700933



ASIN : B071GZJ5C8  
BRAND : Two by Vince Camuto  
euclidean distance from given input image : 0.7760781



ASIN : B01IT9KGS4  
BRAND : Bunny Angle  
euclidean distance from given input image : 0.77783227

george womens 34 sleeve shirt w front rouching xs 02 arctic white														
ralph	2.8	3.2	2.3	3.8	3.6	2.8	3.1	2.4	2.6	2.3	3.8	2.9	2.9	
lauren	2.8	3.3	2.5	4	3.6	2.7	3.2	2.5	2.7	2.5	4	3.3		
active	4.4	3.7	2.6	4.1	4	3.8	3.2	2.9	3.2	2.6	4.1	3.5		
striped	4	0	2.9	4.4	4	3.6	3.4	2.9	3.3	2.9	4.2	3.5		
womens	4.3	3.9	2.9	3.8	3.3	3.7	3.2	2.6	3.3	2.9	4.1	3.7		
hooded	4.8	4.2	3.4	4.3	3.8	4.3	3.6	3.3	3.7	3.4	4.4	3.3		
tee	5	4.6	3.8	4.5	4.2	4.6	4.1	3.7	4.2	3.8	4.9	4.3		
shirt	4.5	4	3.1	3.4	0	4.1	3.5	2.9	3.6	3.1	4.5	3.4		
white	3.5	2.9	0	3.4	3.1	2.8	2.1	1.3	2.3	0	3.3	2.4		
yellow	3.5	3.3	2.3	3.8	3.6	2.8	3	2.3	0	2.3	4	3		
black	3.5	3.3	2.3	3.8	3.6	2.8	3	2.3	0	2.3	4	3		
george	2.8	3.2	2.3	3.8	3.6	2.8	3	2.3	0	2.3	4	3		
womens	3.4	3.8	2.9	4.4	4	3.6	3.4	2.9	3.3	2.9	4.2	3.5		
sleeve	4.5	4.6	3.8	4.5	4.2	4.6	4.1	3.7	4.2	3.8	4.9	4.3		
shirt	4.8	4.2	3.4	4.3	3.8	4.3	3.6	3.3	3.7	3.4	4.4	3.3		
w	5	4.6	3.8	4.5	4.2	4.6	4.1	3.7	4.2	3.8	4.9	4.3		
front	4.8	4.2	3.4	4.3	3.8	4.3	3.6	3.3	3.7	3.4	4.4	3.3		
rouching	4.3	3.9	2.9	3.8	3.3	3.7	3.2	2.6	3.3	2.9	4.1	3.5		
xs	4.3	3.9	2.9	3.8	3.3	3.7	3.2	2.6	3.3	2.9	4.1	3.5		
02	4.8	4.2	3.4	4.3	3.8	4.3	3.6	3.3	3.7	3.4	4.4	3.3		
arctic	5	4.6	3.8	4.5	4.2	4.6	4.1	3.7	4.2	3.8	4.9	4.3		
white	4.8	4.2	3.4	4.3	3.8	4.3	3.6	3.3	3.7	3.4	4.4	3.3		



ASIN : B01MSHGL5V

BRAND : George

euclidean distance from given input image : 0.7824235

lovegifting lady womens norwich city fc sign oneck long sleeve raglan tee shirt xxlarge														
ralph	2.3	3.2	3.2	2	3.1	3	3.3	2.3	3	3.8	2.6	4.1	3.6	2.3
lauren	2.5	3.6	3.3	2.2	3.4	3	3.5	2.5	3.2	4	2.7	4.3	3.6	2.5
active	2.6	3.8	3.7	2.9	3.5	4	3.4	2.6	2.9	4.1	3.2	4.6	4	2.6
striped	2.9	3.5	0	2.9	3.7	3.7	3.8	2.9	3.5	4.4	3.1	4.6	4	2.9
womens	2.9	3.8	3.9	3.1	3.5	4	3.6	2.9	3.3	3.8	2.5	4.3	3.3	2.9
hooded	3.4	4.2	4.2	3.6	4	4.4	4	3.4	3.8	4.3	3.4	4.7	3.8	3.4
tee	3.8	4.5	4.6	3.9	4.5	5	4.3	3.8	4.1	4.5	3.9	0	4.2	3.8
shirt	3.1	4.1	4	3.2	3.8	4.1	3.5	3.1	3.7	3.4	3	4.2	0	3.1
white	0	2.9	2.9	1.3	2.3	3.1	2.5	0	2	3.4	2	3.8	3.1	0
yellow	2.3	3.7	3.3	2.2	3.4	2.9	3.4	2.3	2.9	3.8	2.6	4.2	3.6	2.3
lady	2.3	3.2	3.2	2.2	3.4	2.9	3.4	2.3	2.9	3.8	2.6	4.1	3.6	2.3
womens	2.9	3.8	3.9	3.1	3.5	4	3.6	2.9	3.3	3.8	2.5	4.3	3.3	2.9
norwich	3.4	4.2	4.2	3.6	4	4.4	4	3.4	3.8	4.3	3.4	4.7	3.8	3.4
city	3.8	4.5	4.6	3.9	4.5	5	4.3	3.8	4.1	4.5	3.9	0	4.2	3.8
fc	3.1	4.1	4	3.2	3.8	4.1	3.5	3.1	3.7	3.4	3	4.2	0	3.1
sign	0	2.9	2.9	1.3	2.3	3.1	2.5	0	2	3.4	2	3.8	3.1	0
oneck	2.3	3.2	3.2	2.2	3.4	2.9	3.4	2.3	2.9	3.8	2.6	4.1	3.6	2.3
long	3.8	4.5	4.6	3.9	4.5	5	4.3	3.8	4.1	4.5	3.9	0	4.2	3.8
sleeve	3.1	4.1	4	3.2	3.8	4.1	3.5	3.1	3.7	3.4	3	4.2	0	3.1
raglan	0	2.9	2.9	1.3	2.3	3.1	2.5	0	2	3.4	2	3.8	3.1	0
tee	2.3	3.2	3.2	2.2	3.4	2.9	3.4	2.3	2.9	3.8	2.6	4.1	3.6	2.3
shirt	3.8	4.5	4.6	3.9	4.5	5	4.3	3.8	4.1	4.5	3.9	0	4.2	3.8
xxlarge	3.1	4.1	4	3.2	3.8	4.1	3.5	3.1	3.7	3.4	3	4.2	0	3.1



ASIN : B01M347E01

BRAND : LOVEGIFTTO LADY

euclidean distance from given input image : 0.7842254

polo ralph lauren womens vneck tee shirt top xs small royal lilac														
ralph	4.2	0	2.1	3.2	3.2	2.3	4.1	3.6	3.2	2.3	3.7	3.6	2.3	
lauren	4.3	2.1	0	3.3	2.5</td									

BRAND : RALPH LAUREN  
euclidean distance from given input image : 0.7862537

solow womens solow hilo sleeveless tshirt xs white									
	ralph	2.3	3.2	2.3	2.3	4.2	3.1	2.6	2.9
lauren	2.5	3.3	2.5	2.4	4.2	3.1	2.7	3.3	
active	2.6	3.7	2.6	3	4.6	4.1	3.2	3.5	
striped womens	2.9	0	2.9	3.1	4.1	3.7	3.3	3.5	
hooded	3.4	4.2	3.4	3.6	4.1	4	3.7	3.3	
tee	3.8	4.6	3.8	3.9	4.8	4.4	4.2	4.3	
shirt	3.1	4	3.1	3.4	3.7	2.9	3.6	3.4	
yellowwhite	0	2.9	0	1.7	3.8	3.2	2.3	2.4	
solow	2.3	3.3	2.3	2.3	4.1	3.2	0	3	



ASIN : B01M292SR6  
BRAND : SOLOW  
euclidean distance from given input image : 0.7898404

macbeth collection womens striped tshirt xs green								
	ralph	2.3	3.3	3.2	3.4	3.1	2.6	3.3
lauren	2.5	3.6	3.3	3.5	3.1	2.7	3.5	
active	2.6	3.6	3.7	3.8	4.1	3.2	3.6	
striped womens	2.9	3.6	0	3.9	3.7	3.3	3.8	
hooded	3.4	4.2	4.2	3.5	4	3.7	3.9	
tee	3.8	4.3	4.6	4.3	4.4	4.2	3.7	
shirt	3.1	3.9	4	3.3	2.9	3.6	3.6	
yellowwhite	0	2.5	2.9	2.9	3.2	2.3	2.7	
macbeth	2.3	3.3	3.3	3.3	3.2	0	3.4	
collection								
womens								
striped								
tshirt								
xs								
green								



ASIN : B01E3CPLJK  
BRAND : Macbeth Collection  
euclidean distance from given input image : 0.79283816

fantastic womens jameson irish whiskey jjs logo raglan long sleeves black baseball tee shirt															
	ralph	3.5	3.2	2.3	3.2	2.3	2.3	3.9	2.6	3	4.1	2.9	3.4	4.1	3.6
lauren	3.4	3.3	2.5	3.2	2.5	2.5	4.1	2.7	3.2	4.2	3.4	3.8	4.3	3.6	
active	3.6	3.7	2.6	4.2	2.6	2.6	4.4	3.2	2.9	4.2	3.4	3.7	4.6	4	
striped womens	3.8	0	2.9	3.9	2.9	2.9	4.3	3.1	3.5	4.5	3.5	3.9	4.6	4	
hooded	3.9	3.9	2.9	4.3	2.9	2.9	3.9	2.5	3.3	3.8	3	4	4.3	3.3	
tee	4.2	4.2	3.4	4.7	3.4	3.4	4.3	3.4	3.8	4.3	3.5	4.3	4.7	3.8	
shirt	4.5	4.6	3.8	4.8	3.8	3.8	4.8	3.9	4.1	4.7	4.3	4.3	0	4.2	
yellowwhite	3.8	4	3.1	4.3	3.1	3.1	3.8	3	3.7	3.8	3.5	4	4.2	0	
fantastic	2.8	2.9	0	3.3	0	0	3.5	2	2	3.5	2.5	3	3.8	3.1	
womens	3.4	3.3	2.3	3.3	2.3	2.3	3.9	2.6	2.9	4.1	3	3.7	4.2	3.6	
jameson															
irish															
whiskey															
jjs															
logo															
raglan															
long sleeves															
black															
baseball															
tee															
shirt															



ASIN : B01MCXK7C1  
BRAND : Raglan Baseball Shirt  
euclidean distance from given input image : 0.7957387

=====

	sophie	finzi	ny	womens	aline	striped	cotton	shirt	sz	xl	blackwhite	190160e
ralph	1.9	2.3	2.8	3.2	2.2	3.4	3.9	3.6	3.3	2.8	2.3	2.3
lauren	2	2.5	3.1	3.3	2.4	3.5	4.1	3.6	3.4	2.9	2.5	2.5
active	3.1	2.6	4.1	3.7	2.9	3.8	4.2	4	4	3.5	2.6	2.6
striped	2.8	2.9	3.9	0	3	3.9	4.1	4	3.7	3.5	2.9	2.9
womens	3.1	2.9	4.3	3.9	2.9	0	3.9	3.3	3.6	3.6	2.9	2.9
hooded	3.6	3.4	4.6	4.2	3.5	3.5	4.7	3.8	4.2	4	3.4	3.4
tee	4.1	3.8	4.9	4.6	3.9	4.3	4.8	4.2	4.9	4.5	3.8	3.8
shirt	3.4	3.1	4.2	4	3.2	3.3	4	0	4.3	3.7	3.1	3.1
yellowwhite	1.6	0	3.2	2.9	1.4	2.9	3.5	3.1	3.2	2.6	0	0
	2.2	2.3	3.2	3.3	2.3	3.3	3.9	3.6	3	2.5	2.3	2.3



ASIN : B01EKAMNVE  
BRAND : SOPHIE FINZI  
euclidean distance from given input image : 0.79599833

	womens	small	shirt	blue	fleur	de	lis	long	sleeve	tee	tshirt	
ralph	3.2	3	3.6	3	2.7	3.7	2.8	3	3.8	4.1	3.1	
lauren	3.3	3.3	3.6	3.3	2.7	3.9	3	3.2	4	4.3	3.1	
active	3.7	3.1	4	3.6	3.6	4.3	3.9	2.9	4.1	4.6	4.1	
striped	0	3.5	4	3.6	3.6	4.3	3.7	3.5	4.4	4.6	3.7	
womens	3.9	3.4	3.3	2.5	3.2	4.3	3.9	3.3	3.8	4.3	3.7	
hooded	4.2	3.9	3.8	3.5	3.9	4.6	4.3	3.8	4.3	4.7	4	
tee	4.6	4.3	4.2	4.2	4.5	5.1	4.6	4.1	4.5	0	4.4	
shirt	4	3.8	0	3.3	3.8	4.6	4.1	3.7	3.4	4.2	2.9	
yellowwhite	2.9	2.1	3.1	2.5	2.6	3.5	3	2	3.4	3.8	3.2	
	3.3	2.9	3.6	3	2.6	3.6	3.1	2.9	3.8	4.2	3.2	

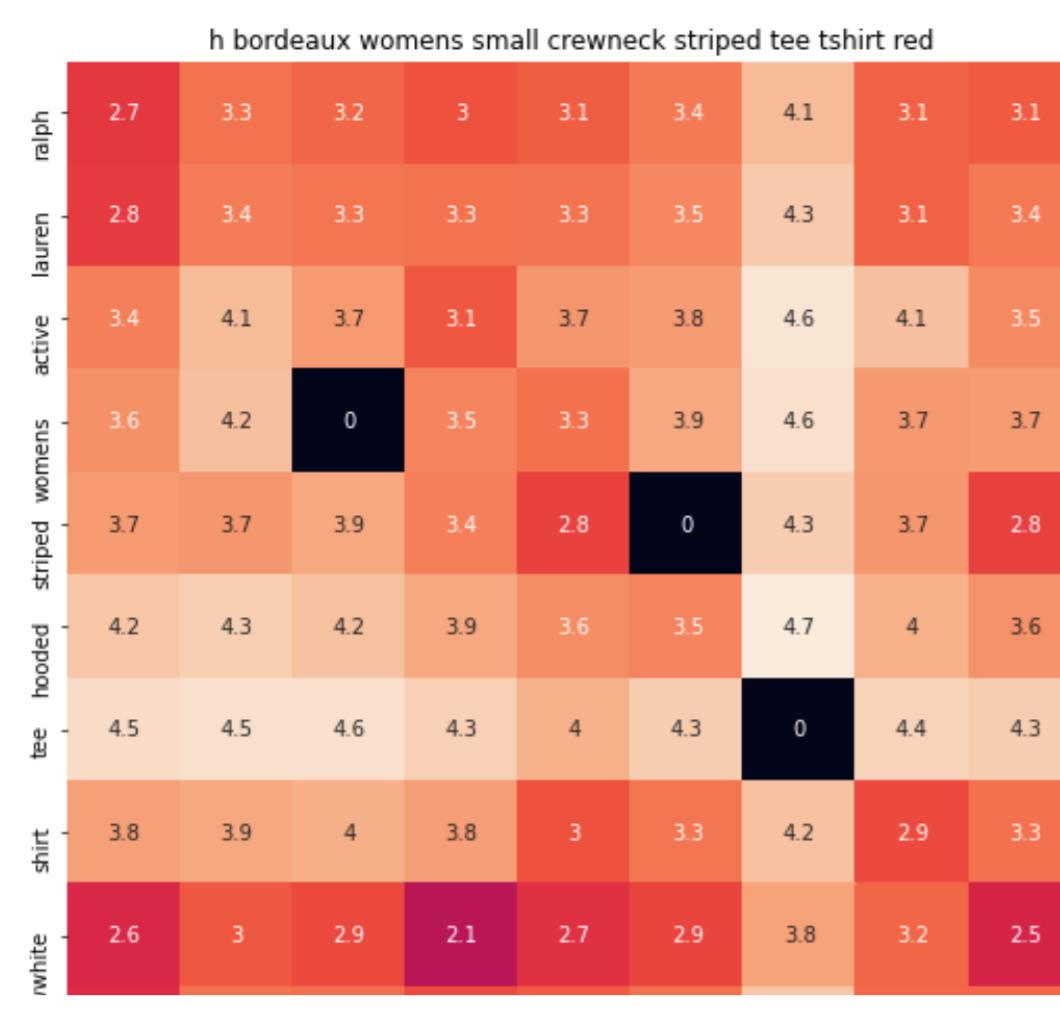


ASIN : B01D1CD420  
BRAND : URBAN X  
euclidean distance from given input image : 0.7979921

	eileen	fisher	womens	pullover	boatneck	tee	tshirt	orange	xl	
ralph	1.8	3.3	3.2	3.9	2.3	4.1	3.1	3.3	2.8	
lauren	2	3.7	3.3	4	2.4	4.3	3.1	3.6	2.9	
active	2.8	3.9	3.7	4.4	2.9	4.6	4.1	3.7	3.5	
striped	2.8	3.9	0	4.3	2.9	4.6	3.7	3.8	3.5	
womens	2.9	4	3.9	3.5	2.4	4.3	3.7	2.8	3.6	
hooded	3.5	4.5	4.2	4.1	3.2	4.7	4	3.8	4	
tee	3.8	4.7	4.6	4.8	3.7	0	4.4	4.4	4.5	
shirt	3.2	4.3	4	3.2	2.9	4.2	2.9	3.4	3.7	
yellowwhite	1.1	3.3	2.9	3.5	1.4	3.8	3.2	2.9	2.6	
	2.2	3.7	3.3	3.9	2.2	4.2	3.2	3.5	2.5	



ASIN : B0752Z5QLC  
BRAND : Eileen Fisher  
euclidean distance from given input image : 0.7992172



ASIN : B072FGSJ15

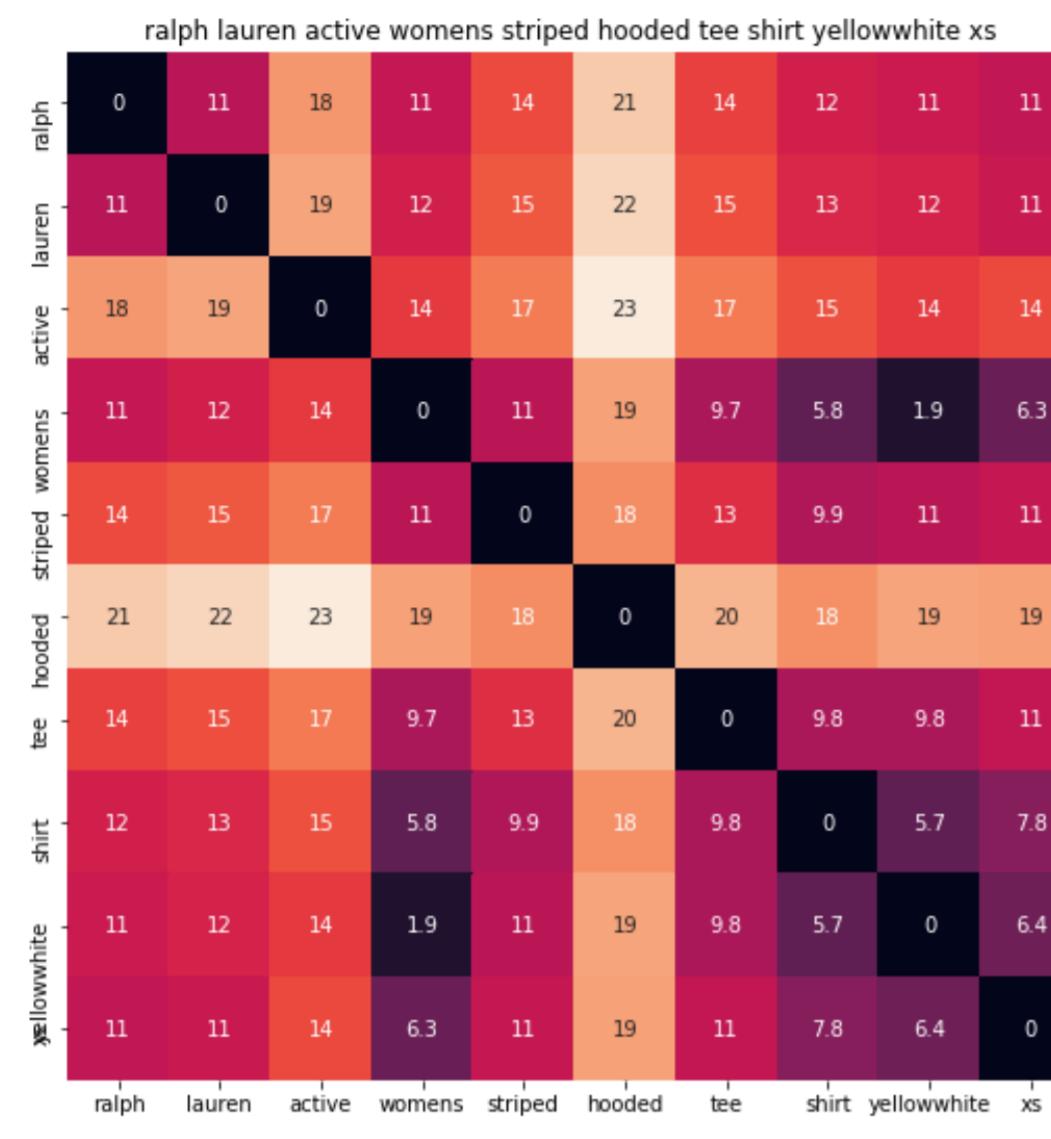
BRAND : H By Bordeaux

euclidean distance from given input image : 0.8040256

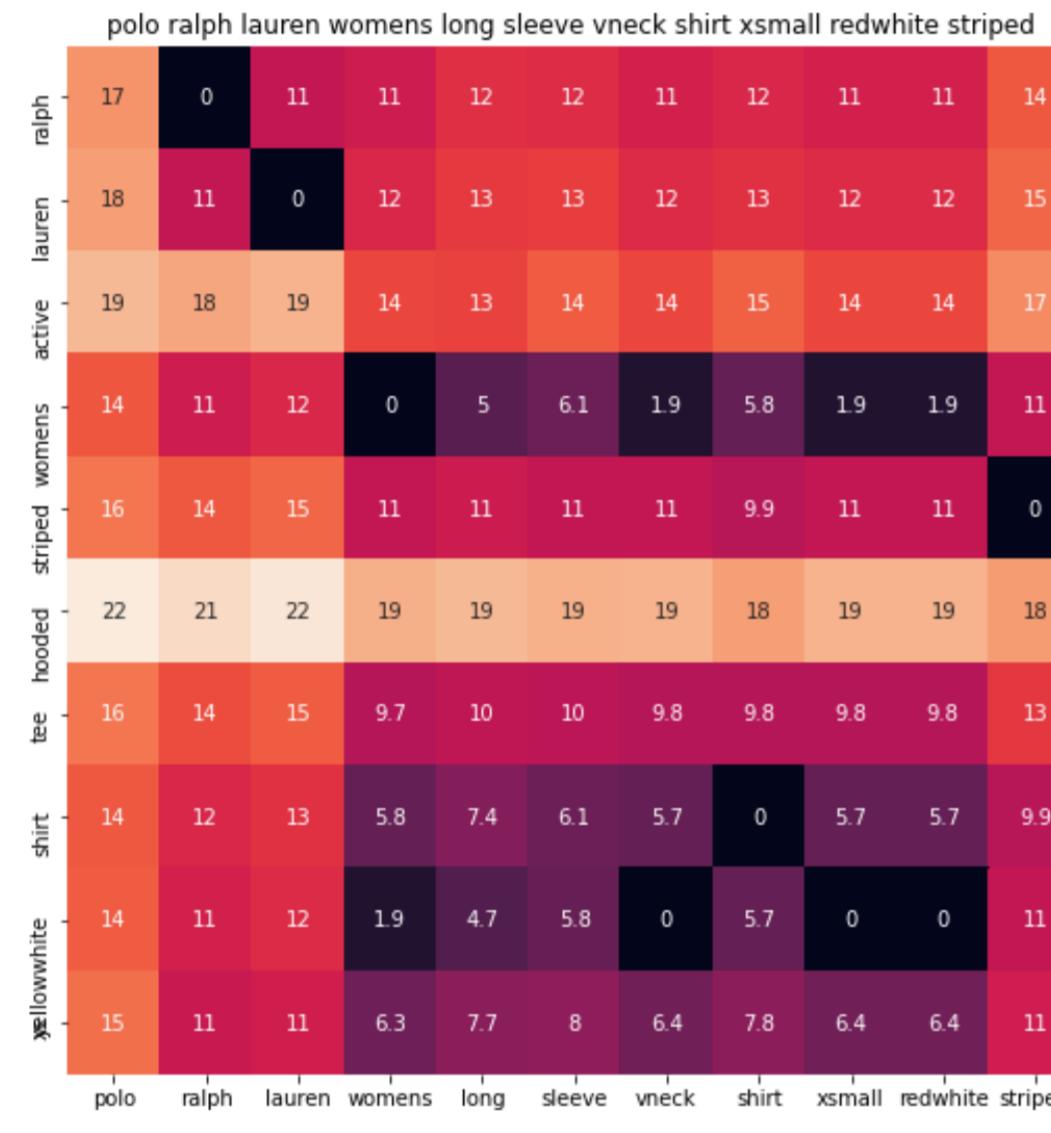
**[9.4] IDF weighted Word2Vec for product similarity**

```
In [46]: 1 doc_id = 0
2 w2v_title_weight = []
3 # for every title we build a weighted vector representation
4 for i in data['title']:
5     w2v_title_weight.append(build_avg_vec(i, 300, doc_id,'weighted'))
6     doc_id += 1
7 # w2v_title = np.array(# number of doc in courpus * 300), each row corresponds to a doc
8 w2v_title_weight = np.array(w2v_title_weight)
```

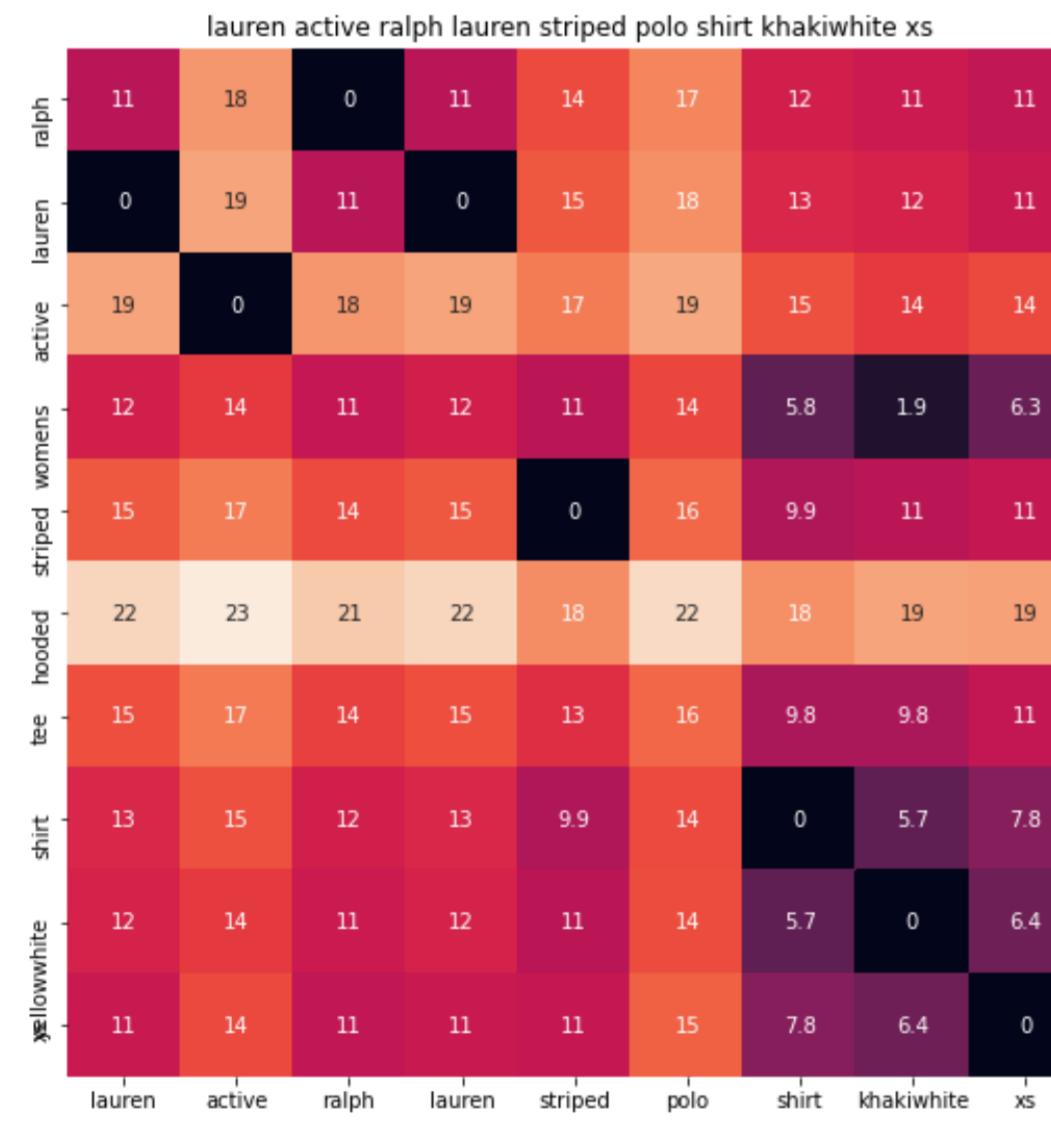
```
In [48]: M
1 def weighted_w2v_model(doc_id, num_results):
2     # doc_id: apparel's id in given corpus
3
4     # pairwise_dist will store the distance from given input apparel to all remaining apparels
5     # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y> / (||X|| * ||Y||)
6     # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
7     pairwise_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))
8
9     # np.argsort will return indices of 9 smallest distances
10    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
11    pdists = np.sort(pairwise_dist.flatten())[0:num_results]
12
13    #data frame indices of the 9 smallest distace's
14    df_indices = list(data.index[indices])
15
16    for i in range(0, len(indices)):
17        heat_map_w2v(data['title'].loc[df_indices[0]],data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i], 'weighted')
18    print('ASIN :',data['asin'].loc[df_indices[i]])
19    print('Brand :',data['brand'].loc[df_indices[i]])
20    print('euclidean distance from input :', pdists[i])
21    print('*'*125)
22
23
24 weighted_w2v_model(12500, 20)
25 #931
26 #12566
27 # in the give heat map, each cell contains the euclidean distance between words i, j
```



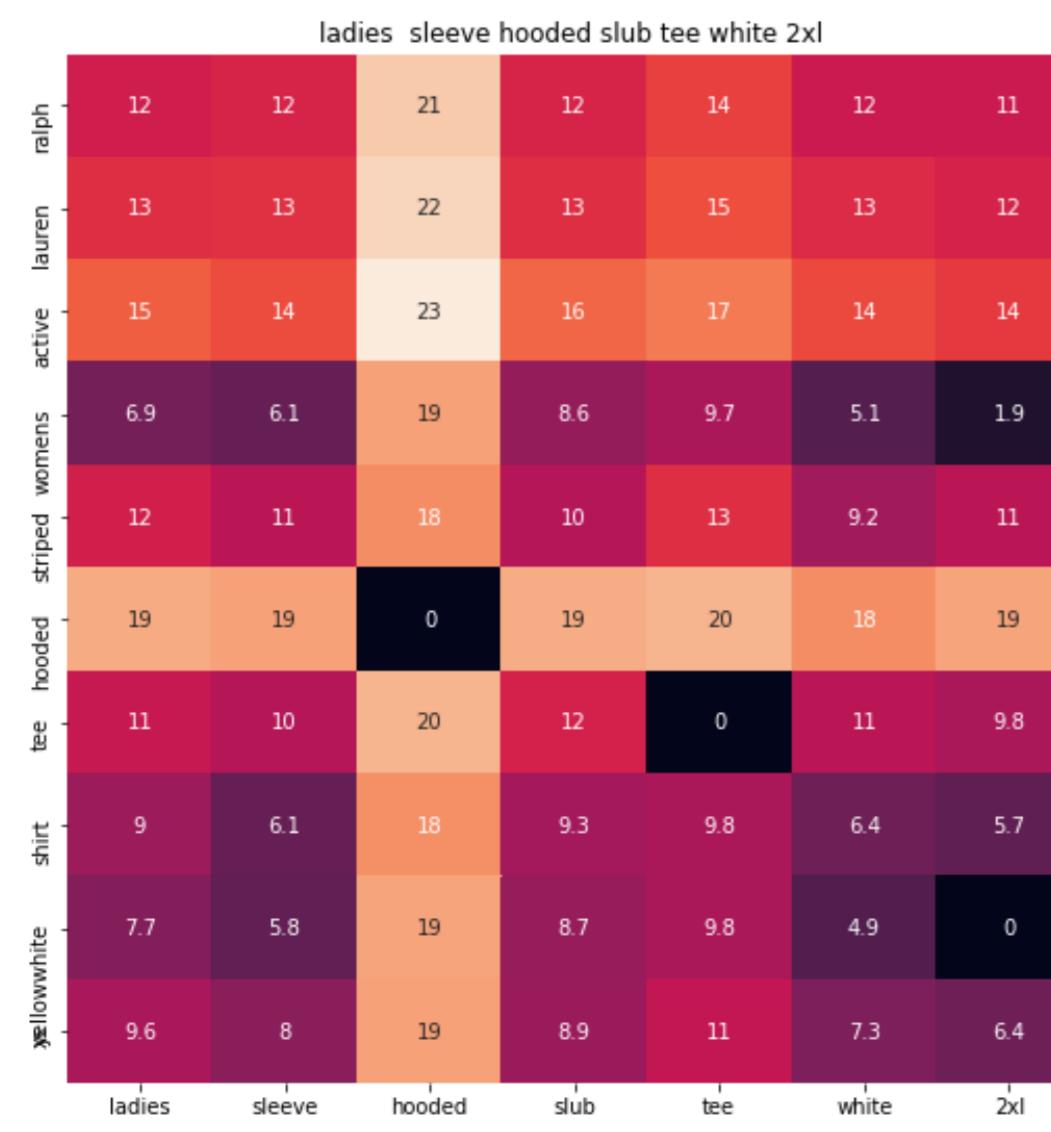
ASIN : B01JME4H6W  
Brand : Ralph Lauren Active  
euclidean distance from input : 0.0



ASIN : B01MRL7J8F  
Brand : Polo Ralph Lauren  
euclidean distance from input : 2.8538036



ASIN : B00ILGK6H2  
Brand : Ralph Lauren Active  
euclidean distance from input : 2.8810923



ASIN : B00C08F6XW

Brand : J. America  
euclidean distance from input : 2.992326

tommy hilfiger womens striped splitneck blouse white xl

	tommy	hilfiger	womens	striped	splitneck	blouse	white	xl
ralph	12	11	11	14	11	12	12	11
lauren	12	12	12	15	12	13	13	12
active	21	14	14	17	14	15	14	15
striped	15	19	0	11	1.9	5.7	5.1	7
womens	17	11	11	0	11	9.6	9.2	12
hooded	23	19	19	18	19	18	18	19
tee	17	9.8	9.7	13	9.8	10	11	12
shirt	16	5.7	5.8	9.9	5.7	5.2	6.4	8.4
yellowwhite	16	0	1.9	11	0	5.8	4.9	7.2
	14	6.4	6.3	11	6.4	7.8	7.3	6.9



ASIN : B071VBTXCP  
Brand : Tommy Hilfiger  
euclidean distance from input : 2.9987054

lauren active ralph lauren short sleeve collared shirt blackwhite small

	lauren	active	ralph	lauren	short	sleeve	collared	shirt	blackwhite	small
ralph	11	18	0	11	13	12	22	12	11	12
lauren	0	19	11	0	14	13	22	13	12	13
active	19	0	18	19	14	14	23	15	14	14
striped	12	14	11	12	6.5	6.1	20	5.8	1.9	5.2
womens	15	17	14	15	12	11	19	9.9	11	11
hooded	22	23	21	22	20	19	24	18	19	19
tee	15	17	14	15	11	10	21	9.8	9.8	11
shirt	13	15	12	13	8.1	6.1	20	0	5.7	7.5
yellowwhite	12	14	11	12	6.2	5.8	20	5.7	0	5
	11	14	11	11	8.7	8	20	7.8	6.4	7.6



ASIN : B00HSX5ZAW  
Brand : Ralph Lauren Active  
euclidean distance from input : 3.0545192

tommy hilfiger ansley striped boatneck top blue small

	tommy	hilfiger	ansley	striped	boatneck	top	blue	small
ralph	12	11	11	14	13	12	12	12
lauren	12	12	12	15	13	13	13	13
active	21	14	14	17	16	13	15	14
striped	15	1.9	1.9	11	8.6	2.7	6.4	5.2
womens	17	11	11	0	9.3	11	8.7	11
hooded	23	19	19	18	18	19	18	19
tee	17	9.8	9.8	13	11	9.9	11	11
shirt	16	5.7	5.7	9.9	8.5	5.9	6.9	7.5
yellowwhite	16	0	0	11	8.9	2.3	6.3	5
	14	6.4	6.4	11	9	6.7	7.9	7.6



ASIN : B01LYYU8BB  
Brand : Tommy Hilfiger  
euclidean distance from input : 3.0882964

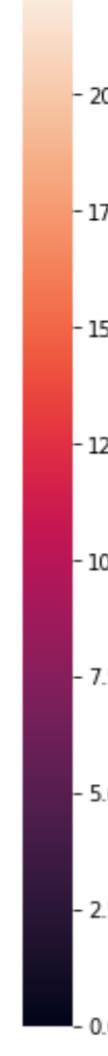
ralph lauren rrl double rl womens long sleeve striped work shirt blue white size 1 small

	ralph lauren	rrl	double	rl	womens	long	sleeve	striped	work	shirt	blue	white	size	1
ralph	0	11	11	19	21	11	12	12	14	17	12	12	12	12
lauren	11	0	12	19	21	12	13	13	15	17	13	13	13	13
active	18	19	14	19	27	14	13	14	17	16	15	15	14	14
striped	11	12	19	15	23	0	5	6.1	11	12	5.8	6.4	5.1	5.2
womens	14	15	11	17	25	11	11	11	0	16	9.9	8.7	9.2	11
hooded	21	22	19	23	29	19	19	19	18	23	18	18	18	19
tee	14	15	9.8	17	25	9.7	10	10	13	15	9.8	11	11	11
shirt	12	13	5.7	15	24	5.8	7.4	6.1	9.9	13	0	6.9	6.4	7.5
yellowwhite	11	12	0	14	24	1.9	4.7	5.8	11	12	5.7	6.3	4.9	5
	11	11	6.4	15	22	6.3	7.7	8	11	14	7.8	7.9	7.3	7.6



ASIN : B01K5BRKGK  
Brand : RRL  
euclidean distance from input : 3.0914545

polo ralph lauren womens vneck tee shirt top xsmall pinknavy										
ralph	17	0	11	11	11	14	12	12	11	11
lauren	18	11	0	12	12	15	13	13	12	12
active	19	18	19	14	14	17	15	13	14	14
striped	14	11	12	0	19	9.7	5.8	2.7	1.9	1.9
womens	16	14	15	11	11	13	9.9	11	11	11
hooded	22	21	22	19	19	20	18	19	19	19
tee	16	14	15	9.7	9.8	0	9.8	9.9	9.8	9.8
shirt	14	12	13	5.8	5.7	9.8	0	5.9	5.7	5.7
white	14	11	12	1.9	0	9.8	5.7	2.3	0	0
xsmall	15	11	11	6.3	6.4	11	7.8	6.7	6.4	6.4
pinknavy										



ASIN : B01ETUBK3W  
Brand : RALPH LAUREN  
euclidean distance from input : 3.1127374

dkny womens small striped scoopnec tee tshirt white										
ralph	11	11	12	14	11	14	11	12		
lauren	12	12	13	15	12	15	11	13		
active	14	14	14	17	14	17	15	14		
striped	1.9	0	5.2	11	1.9	9.7	6.8	5.1		
womens	11	11	11	0	11	13	11	9.2		
hooded	19	19	19	18	19	20	18	18		
tee	9.8	9.7	11	13	9.8	0	11	11		
shirt	5.7	5.8	7.5	9.9	5.7	9.8	6	6.4		
white	0	1.9	5	11	0	9.8	7	4.9		
xsmall	6.4	6.3	7.6	11	6.4	11	7.7	7.3		
pinknavy										



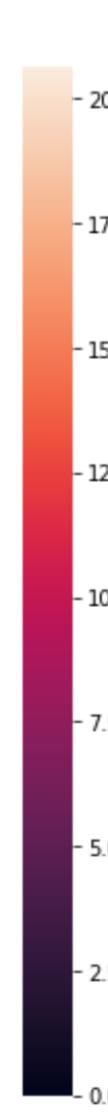
ASIN : B0725PWQFW  
Brand : DKNY  
euclidean distance from input : 3.1371267

tommy hilfiger white womens small striped blouse blue										
ralph	12	11	12	11	12	14	12	12		
lauren	12	12	13	12	13	15	13	13		
active	21	14	14	14	14	17	15	15		
striped	15	1.9	5.1	0	5.2	11	5.7	6.4		
womens	17	11	9.2	11	11	0	9.6	8.7		
hooded	23	19	18	19	19	18	18	18		
tee	17	9.8	11	9.7	11	13	10	11		
shirt	16	5.7	6.4	5.8	7.5	9.9	5.2	6.9		
white	16	0	4.9	1.9	5	11	5.8	6.3		
xsmall	14	6.4	7.3	6.3	7.6	11	7.8	7.9		
blue										



ASIN : B0759NSBV6  
Brand : Tommy Hilfiger  
euclidean distance from input : 3.1382675

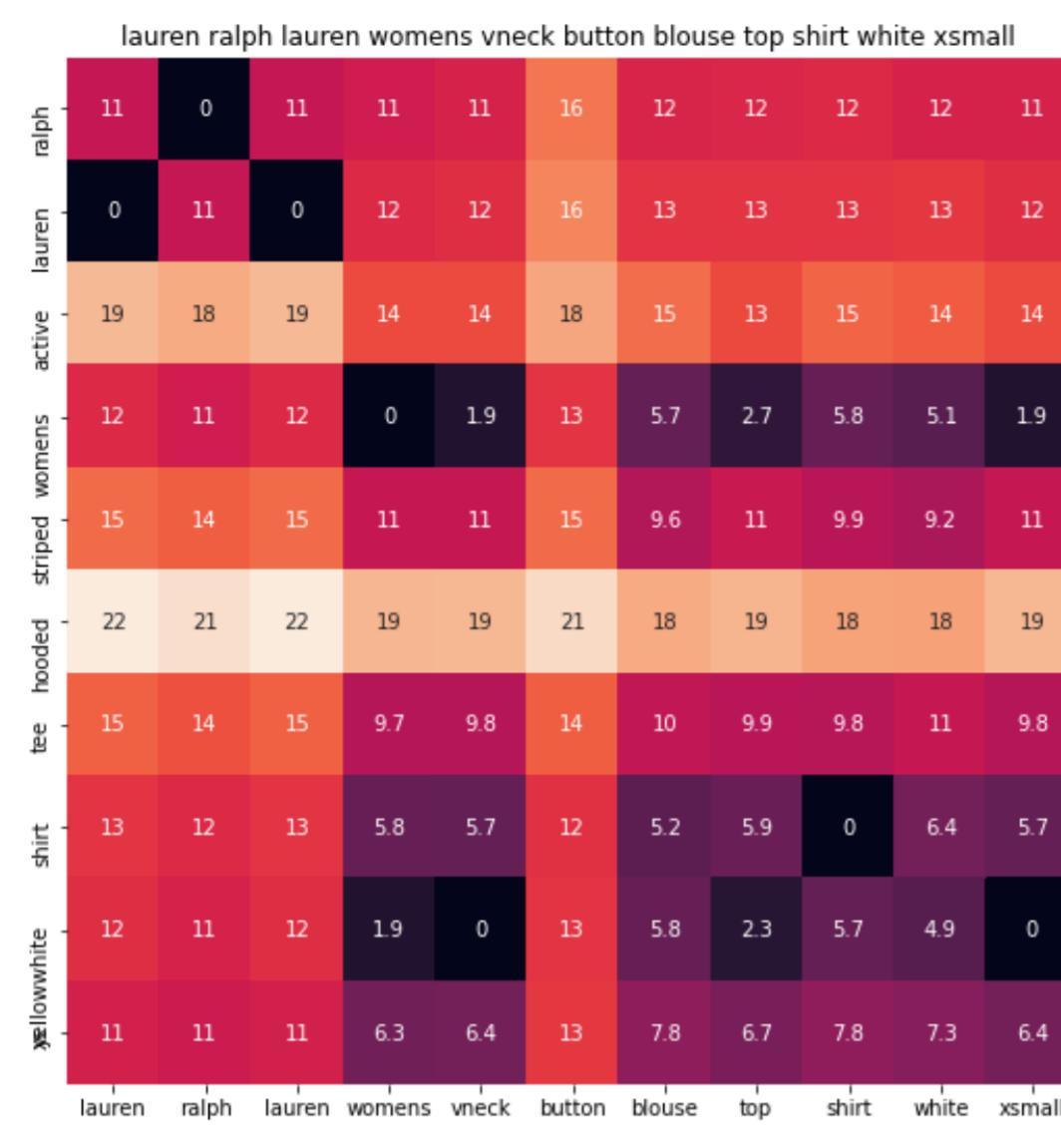
chloe k white womens striped colorblock blouse blue xs										
ralph	11	12	11	14	13	12	12	11		
lauren	10	13	12	15	12	13	13	11		
active	19	14	14	17	16	15	15	14		
striped	12	5.1	0	11	8.5	5.7	6.4	6.3		
womens	14	9.2	11	0	9.9	9.6	8.7	11		
hooded	21	18	19	18	18	18	18	19		
tee	14	11	9.7	13	12	10	11	11		
shirt	12	6.4	5.8	9.9	9	5.2	6.9	7.8		
white	12	4.9	1.9	11	8.9	5.8	6.3	6.4		
xsmall	11	7.3	6.3	11	9.3	7.8	7.9	0		
blue										



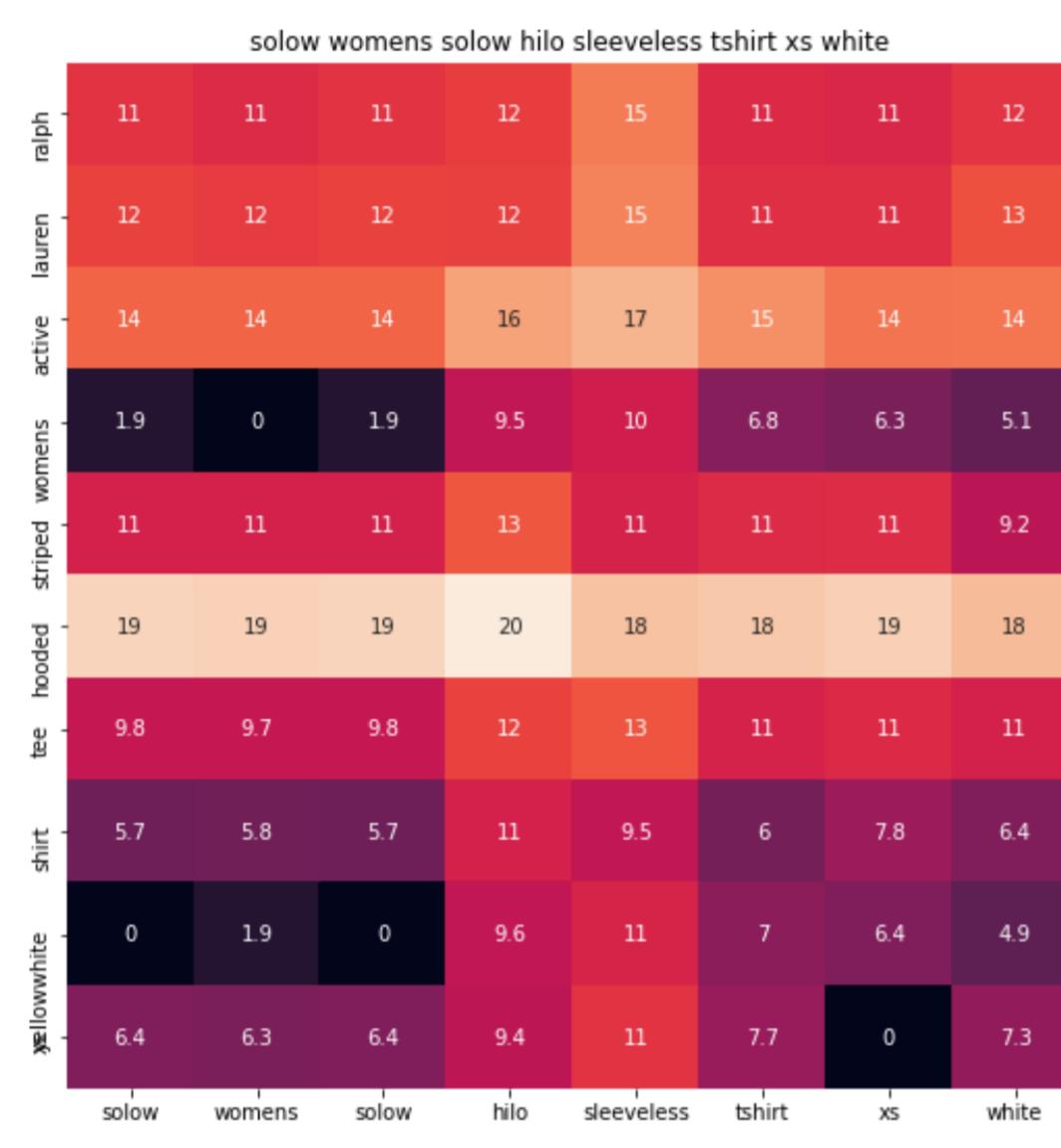
ASIN : B074QTKXPP  
Brand : Chloe K.  
euclidean distance from input : 3.1397839



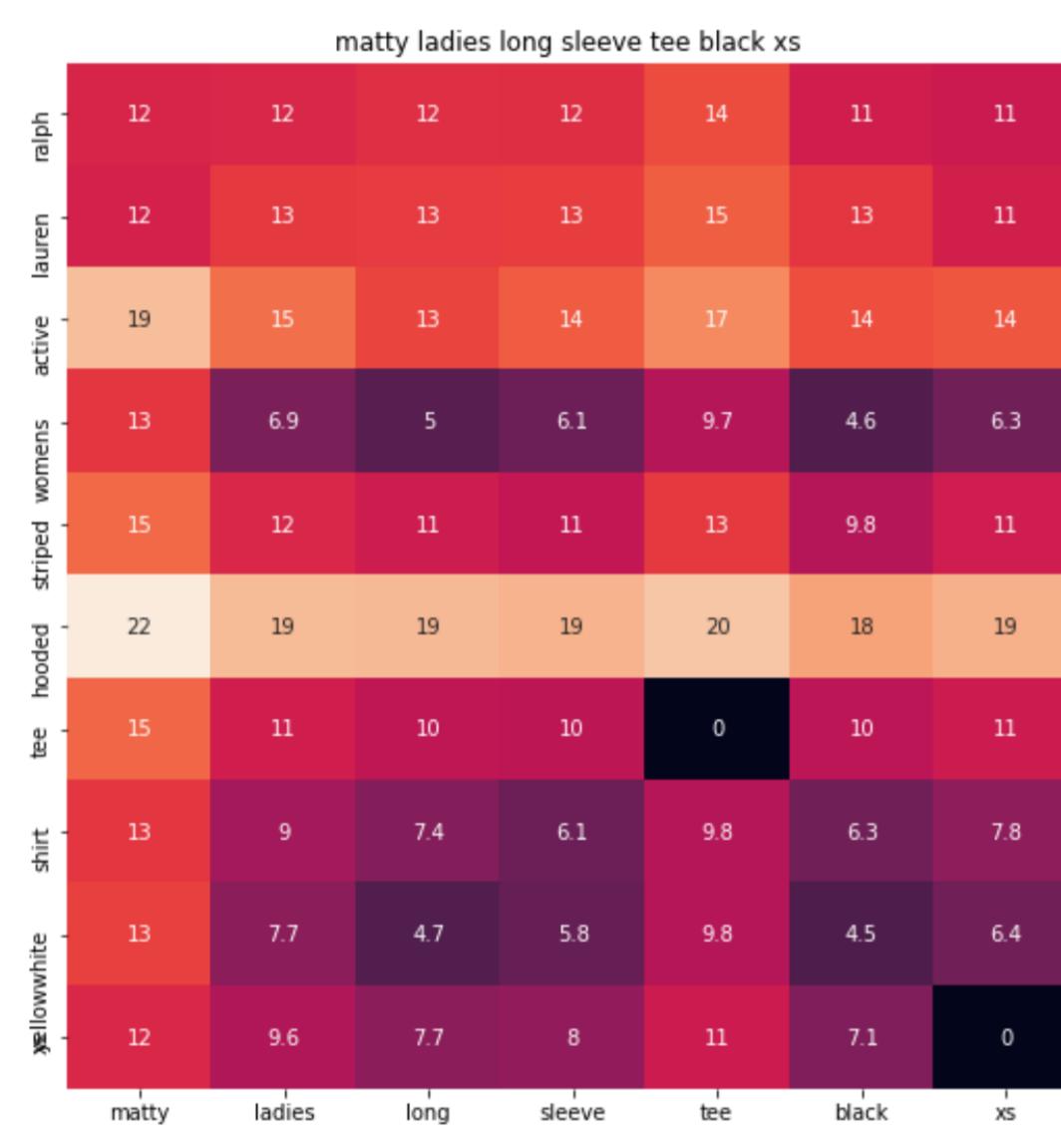
ASIN : B0759NLYJP  
Brand : Tommy Hilfiger  
euclidean distance from input : 3.1414907



ASIN : B0713MCCFX  
Brand : RALPH LAUREN  
euclidean distance from input : 3.1532543



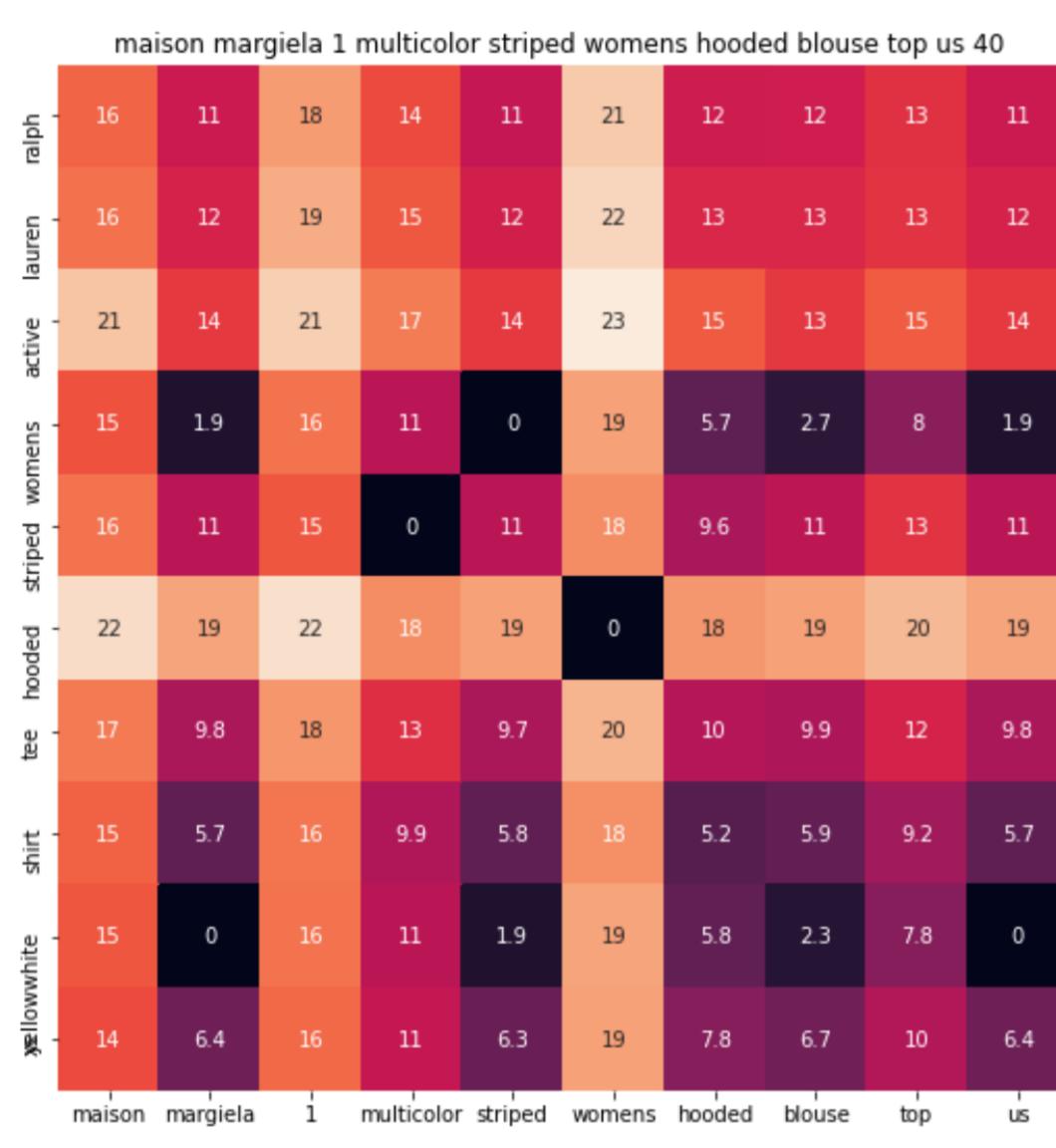
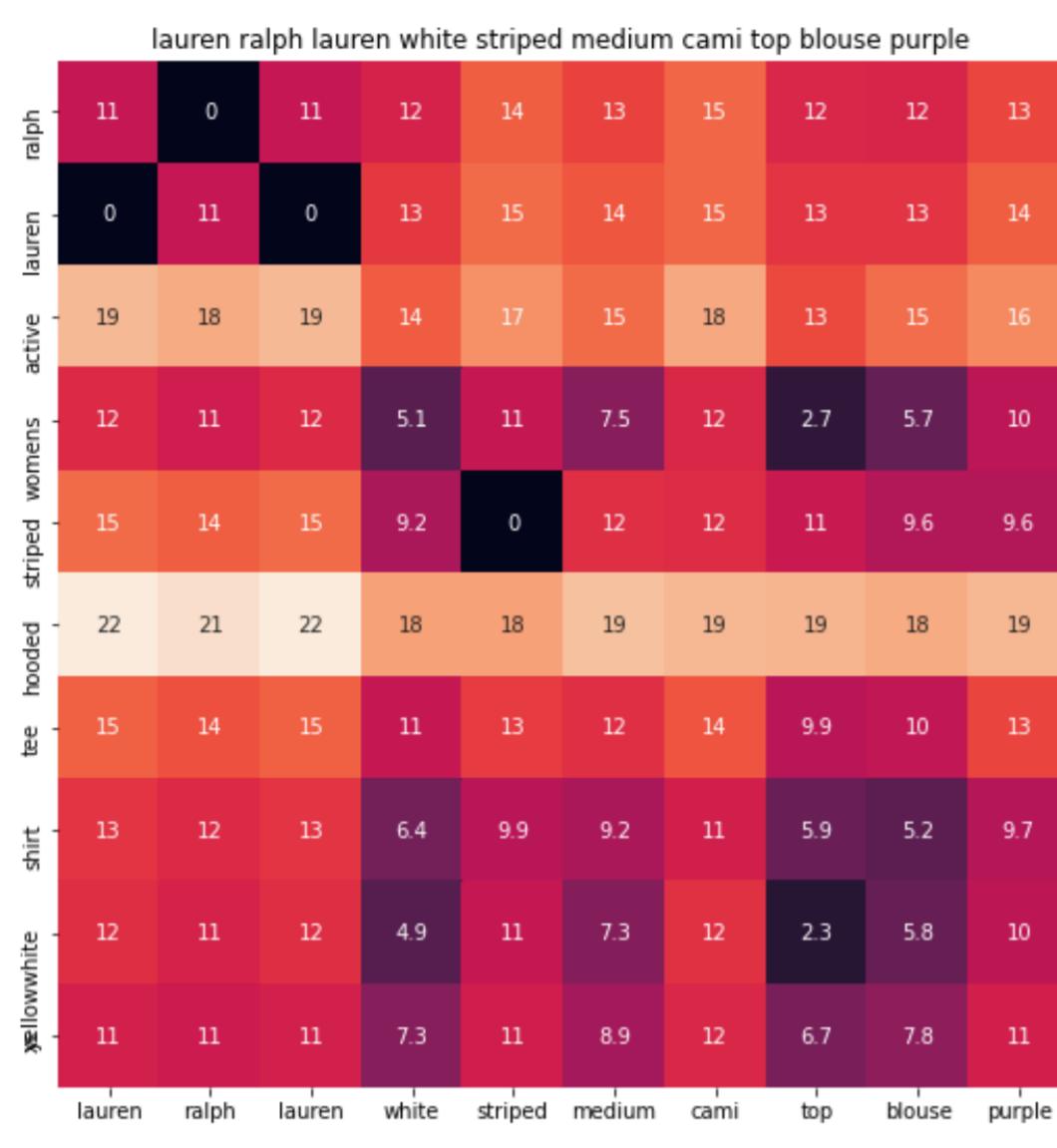
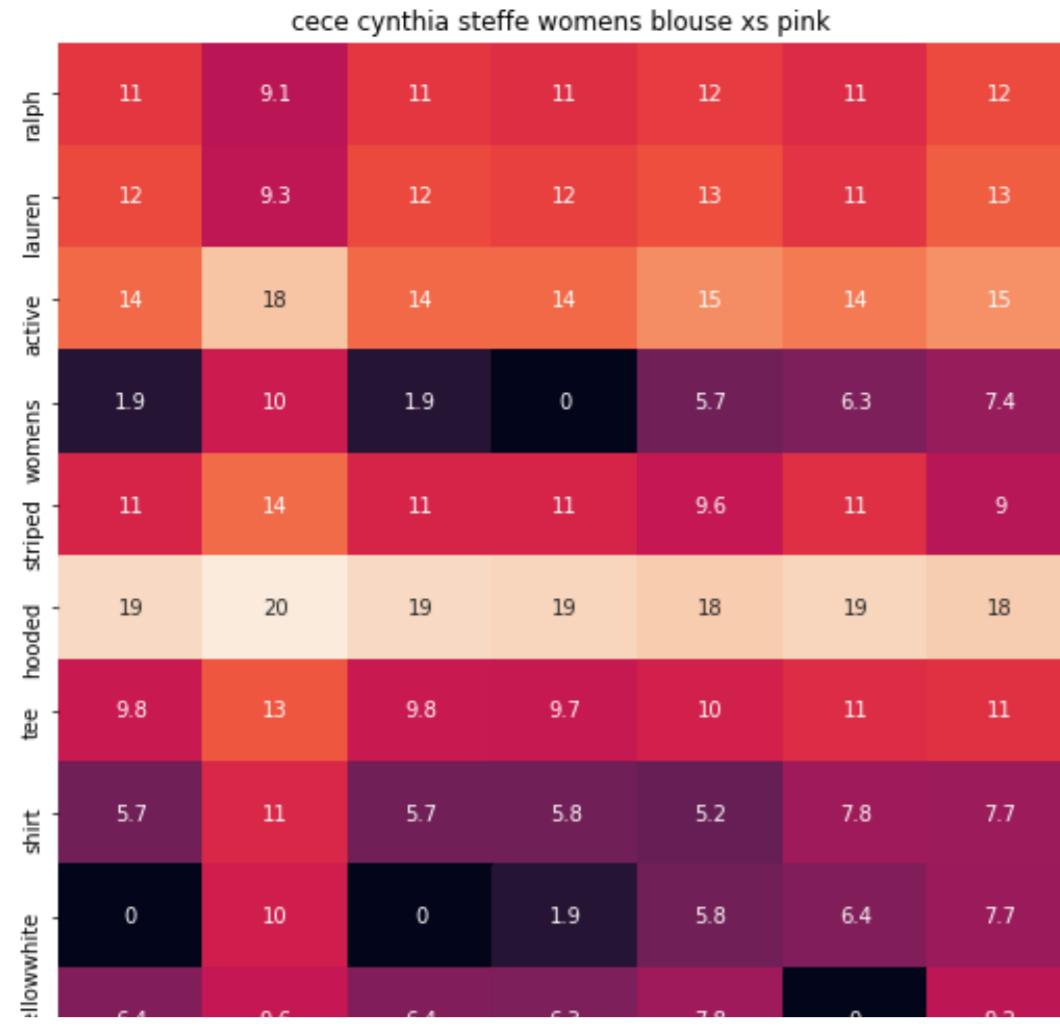
ASIN : B01M292SR6  
Brand : SOLOW  
euclidean distance from input : 3.1820793



ASIN : B01N4714YM  
Brand : Matty M  
euclidean distance from input : 3.2157245



ASIN : B014EII2U  
Brand : Tommy Hilfiger  
euclidean distance from input : 3.2291455

**Observations:**

- BOW, IDF, TFIDF models were able to capture shirts with brand name and size well.
- Word-to-vec model is able to capture hooded shirts well unlike BOW and tfidf, idf

**[9.6] Weighted similarity using brand and color.**

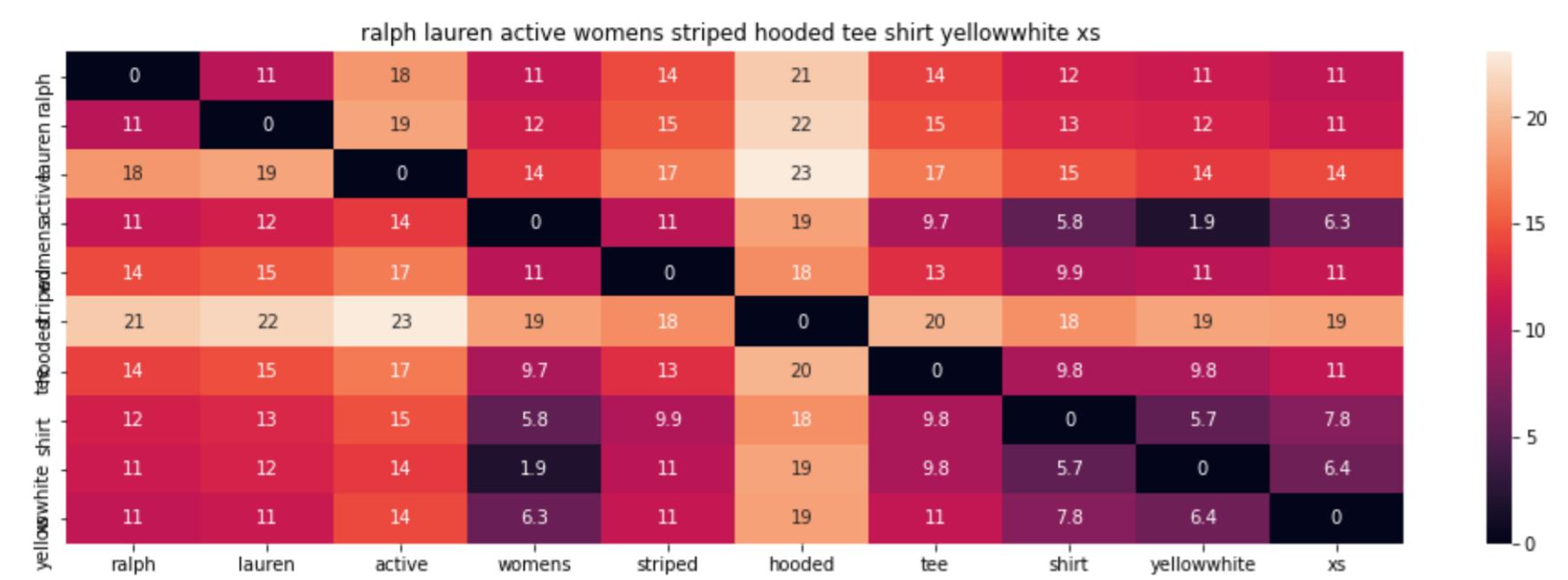
```
In [48]: 1 # some of the brand values are empty.
2 # Need to replace Null with string "NULL"
3 data['brand'].fillna(value="Not given", inplace=True )
4
5 # replace spaces with hyphen
6 brands = [x.replace(" ", "-") for x in data['brand'].values]
7 types = [x.replace(" ", "-") for x in data['product_type_name'].values]
8 colors = [x.replace(" ", "-") for x in data['color'].values]
9
10 brand_vectorizer = CountVectorizer()
11 brand_features = brand_vectorizer.fit_transform(brands)
12
13 type_vectorizer = CountVectorizer()
14 type_features = type_vectorizer.fit_transform(types)
15
16 color_vectorizer = CountVectorizer()
17 color_features = color_vectorizer.fit_transform(colors)
18
19 extra_features = hstack((brand_features, type_features, color_features)).tocsr()
```

```
In [49]: M 1 def heat_map_w2v_brand(sentance1, sentance2, url, doc_id1, doc_id2, df_id1, df_id2, model):
2     # sentance1 : title1, input apparel
3     # sentance2 : title2, recommended apparel
4     # url: apparel image url
5     # doc_id1: document id of input apparel
6     # doc_id2: document id of recommended apparel
7     # df_id1: index of document1 in the data frame
8     # df_id2: index of document2 in the data frame
9     # model: it can have two values, 1. avg 2. weighted
10
11    #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/avg) of length 300 corresponds to each word in give title
12    s1_vec = get_word_vec(sentance1, doc_id1, model)
13    #s2_vec = np.array(#number_of_words_title2 * 300), each row is a vector(weighted/avg) of length 300 corresponds to each word in give title
14    s2_vec = get_word_vec(sentance2, doc_id2, model)
15
16    # s1_s2_dist = np.array(#number of words in title1 * #number of words in title2)
17    # s1_s2_dist[i,j] = euclidean distance between words i, j
18    s1_s2_dist = get_distance(s1_vec, s2_vec)
19
20    data_matrix = [[['Asin', 'Brand', 'Color', 'Product type'],
21                   [data['asin'].loc[df_id1], brands[doc_id1], colors[doc_id1], types[doc_id1]], # input apparel's features
22                   [data['asin'].loc[df_id2], brands[doc_id2], colors[doc_id2], types[doc_id2]]] # recommended apparel's features
23
24    colorscale = [[0, '#1d004d'], [.5, '#f2e5ff'], [1, '#f2e5d1']] # to color the headings of each column
25
26    # we create a table with the data_matrix
27    table = ff.create_table(data_matrix, index=True, colorscale=colorscale)
28    # plot it with plotly
29    plotly.offline.iplot(table, filename='simple_table')
30
31    # devide whole figure space into 25 * 1:10 grids
32    gs = gridspec.GridSpec(25, 15)
33    fig = plt.figure(figsize=(25,5))
34
35    # in Last 25 * 10:15 grids we display image
36
37    # in first 25*10 grids we plot heatmap
38    ax1 = plt.subplot(gs[:, :-5])
39    # plotting the heap map based on the pairwise distances
40    ax1 = sns.heatmap(np.round(s1_s2_dist,6), annot=True)
41    # set the x axis labels as recommended apparels title
42    ax1.set_xticklabels(sentance2.split())
43    # set the y axis labels as input apparels title
44    ax1.set_yticklabels(sentance1.split())
45    # set title as recommended apparels title
46    ax1.set_title(sentance2)
47
48    ax2 = plt.subplot(gs[:, 10:16])
49    # we dont display grid Lins and axis Labels to images
50    ax2.grid(False)
51    ax2.set_xticks([])
52    ax2.set_yticks([])
53
54    # pass the url it display it
55    display_img(url, ax2, fig)
56
57    plt.show()
```

```
In [50]: M 1 def idf_w2v_brand(doc_id, w1, w2, num_results):
2     # doc_id: apparel's id in given corpus
3     # w1: weight for w2v features
4     # w2: weight for brand and color features
5
6     # pairwise_dist will store the distance from given input apparel to all remaining apparels
7     # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y> / (||X|| * ||Y||)
8     # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
9     idf_w2v_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))
10    ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id])
11    pairwise_dist = (w1 * idf_w2v_dist + w2 * ex_feat_dist)/float(w1 + w2)
12
13    # np.argsort will return indices of 9 smallest distances
14    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
15    #pdists will store the 9 smallest distances
16    pdists = np.sort(pairwise_dist.flatten())[0:num_results]
17
18    #data frame indices of the 9 smallest distace's
19    df_indices = list(data.index[indices])
20
21
22    for i in range(0, len(indices)):
23        heat_map_w2v_brand(data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i], df_indices[0], df_indices[i], 'weighted')
24        print('ASIN : ', data['asin'].loc[df_indices[i]])
25        print('Brand : ', data['brand'].loc[df_indices[i]])
26        print('euclidean distance from input : ', pdists[i])
27        print('*'*125)
```

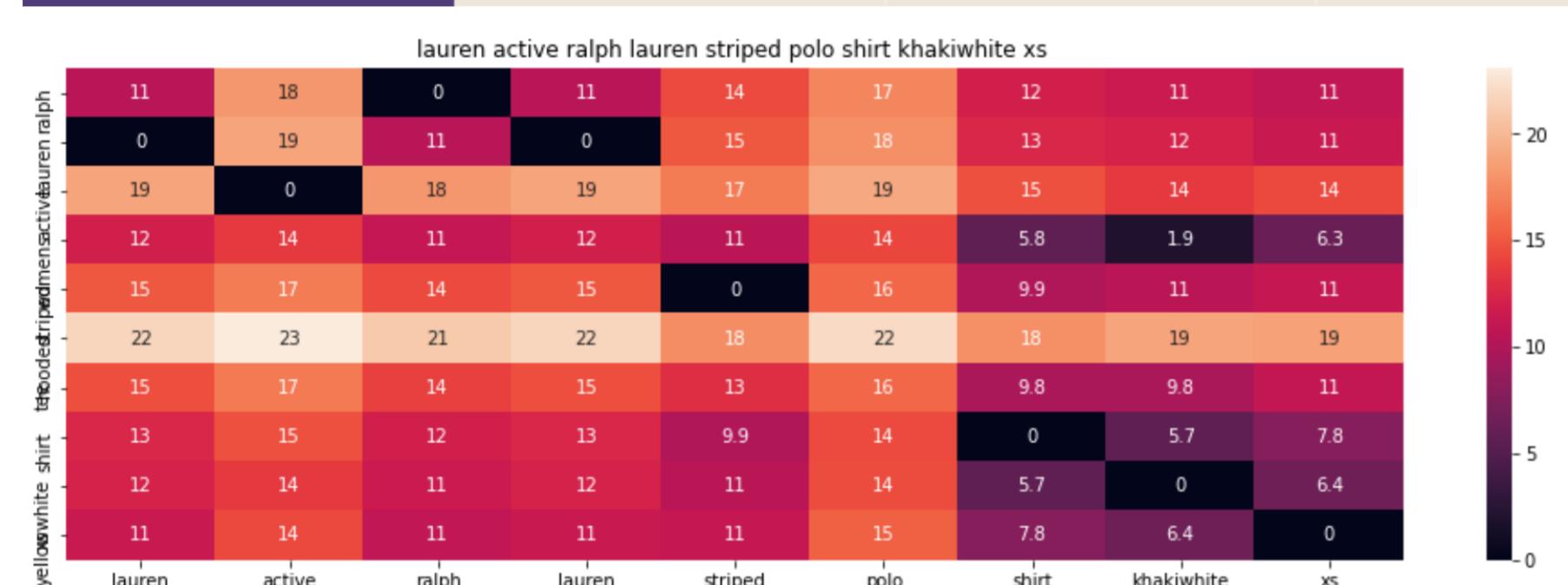
```
In [66]: 1 def idf_w2v_brand(doc_id, w1, w2, num_results):
2     # doc_id: apparel's id in given corpus
3     # w1: weight for w2v features
4     # w2: weight for brand and color features
5
6     # pairwise_dist will store the distance from given input apparel to all remaining apparels
7     # the metric we used here is cosine, the coside distance is measured as K(X, Y) = <X, Y> / (||X|| * ||Y||)
8     # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
9     idf_w2v_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))
10    ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id])
11    pairwise_dist = (w1 * idf_w2v_dist + w2 * ex_feat_dist) / float(w1 + w2)
12
13    # np.argsort will return indices of 9 smallest distances
14    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
15    #pdists will store the 9 smallest distances
16    pdists = np.sort(pairwise_dist.flatten())[0:num_results]
17
18    #data frame indices of the 9 smallest distance's
19    df_indices = list(data.index[indices])
20
21
22    for i in range(0, len(indices)):
23        heat_map_w2v_brand(data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i], df_indices[0], df_indices[i], 'weighted')
24        print('ASIN :', data['asin'].loc[df_indices[i]])
25        print('Brand :', data['brand'].loc[df_indices[i]])
26        print('euclidean distance from input :', pdists[i])
27        print('='*125)
28
29 idf_w2v_brand(12500, 5, 5, 20)
30 # in the give heat map, each cell contains the euclidean distance between words i, j
```

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT



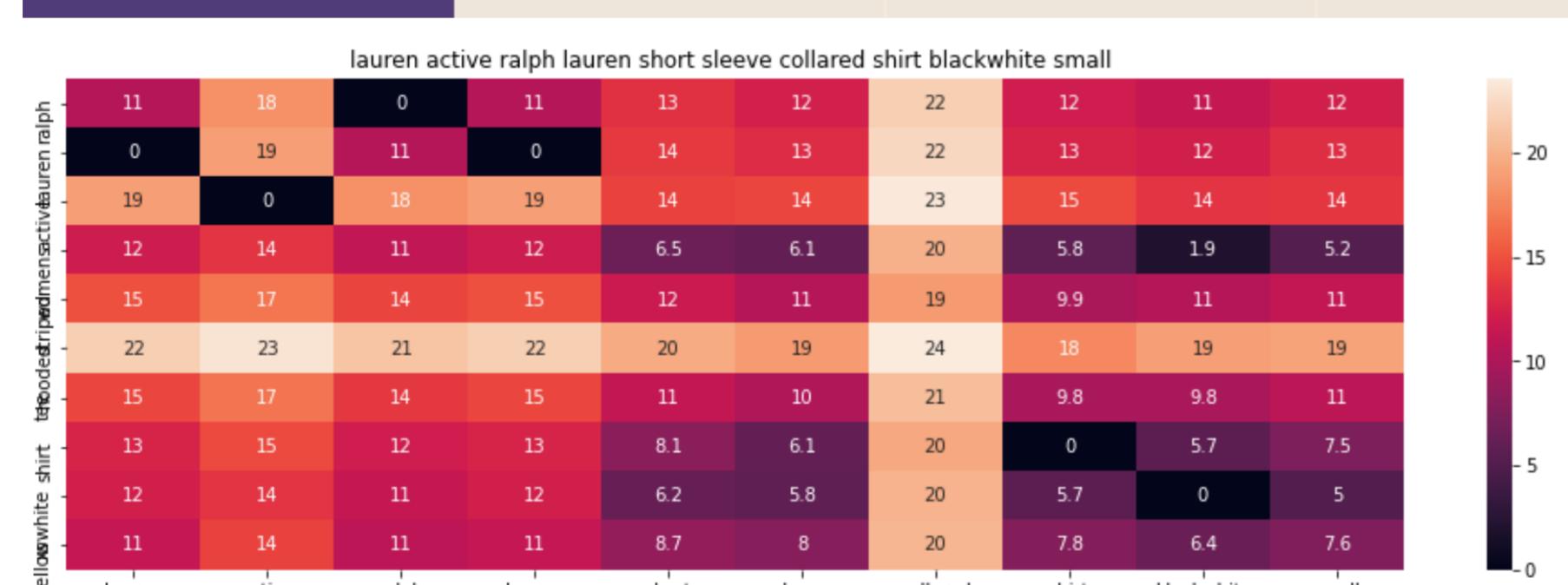
ASIN : B01JME4H6W  
Brand : Ralph Lauren Active  
euclidean distance from input : 0.0

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B00ILGK6H2	Ralph-Lauren-Active	Khaki/White	SHIRT



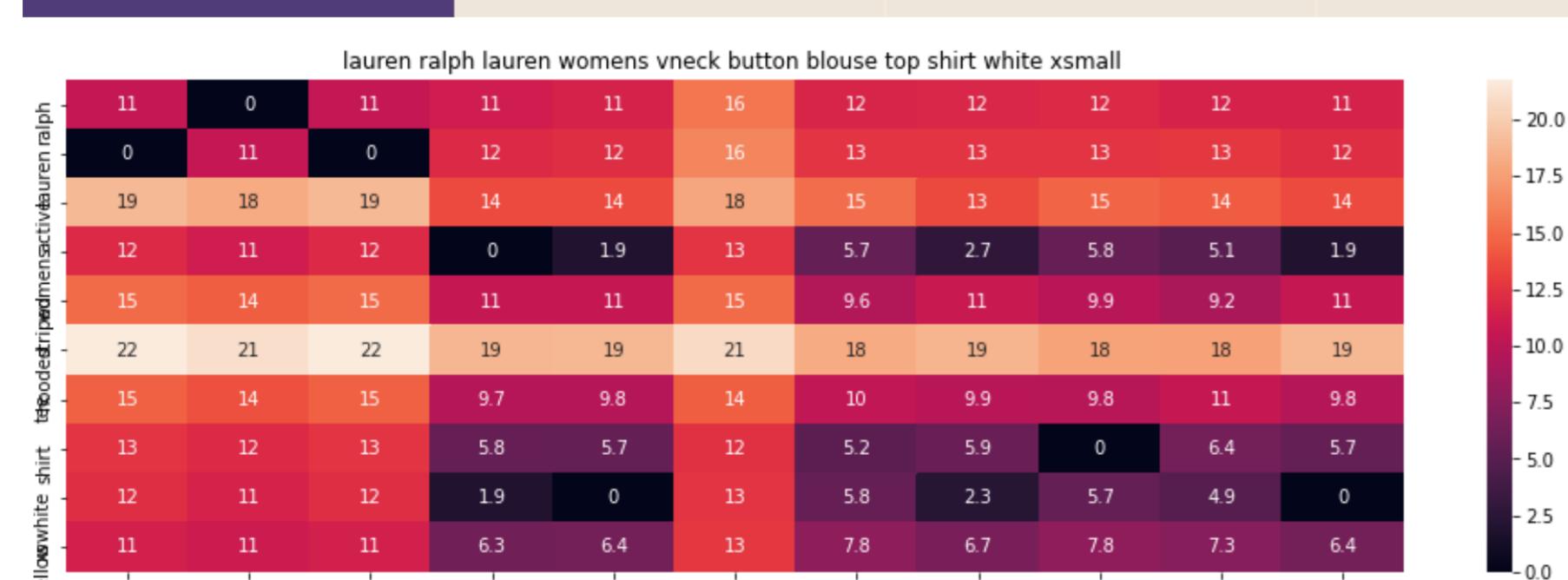
ASIN : B00ILGK6H2  
Brand : Ralph Lauren Active  
euclidean distance from input : 2.1476529123205808

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B00HSX5ZAW	Ralph-Lauren-Active	Black/White	SHIRT



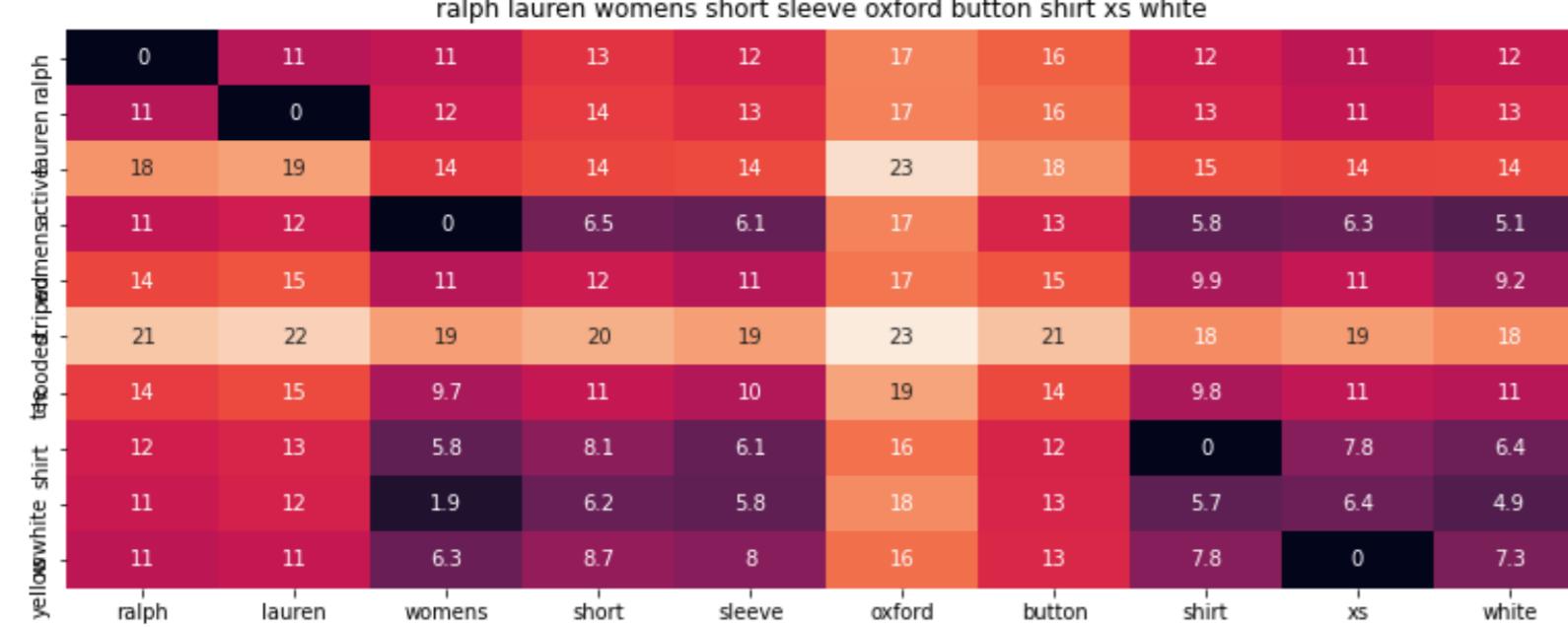
ASIN : B00HSX5ZAW  
Brand : Ralph Lauren Active  
euclidean distance from input : 2.2343664171118407

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B0713MCCFX	RALPH-LAUREN	White	SHIRT



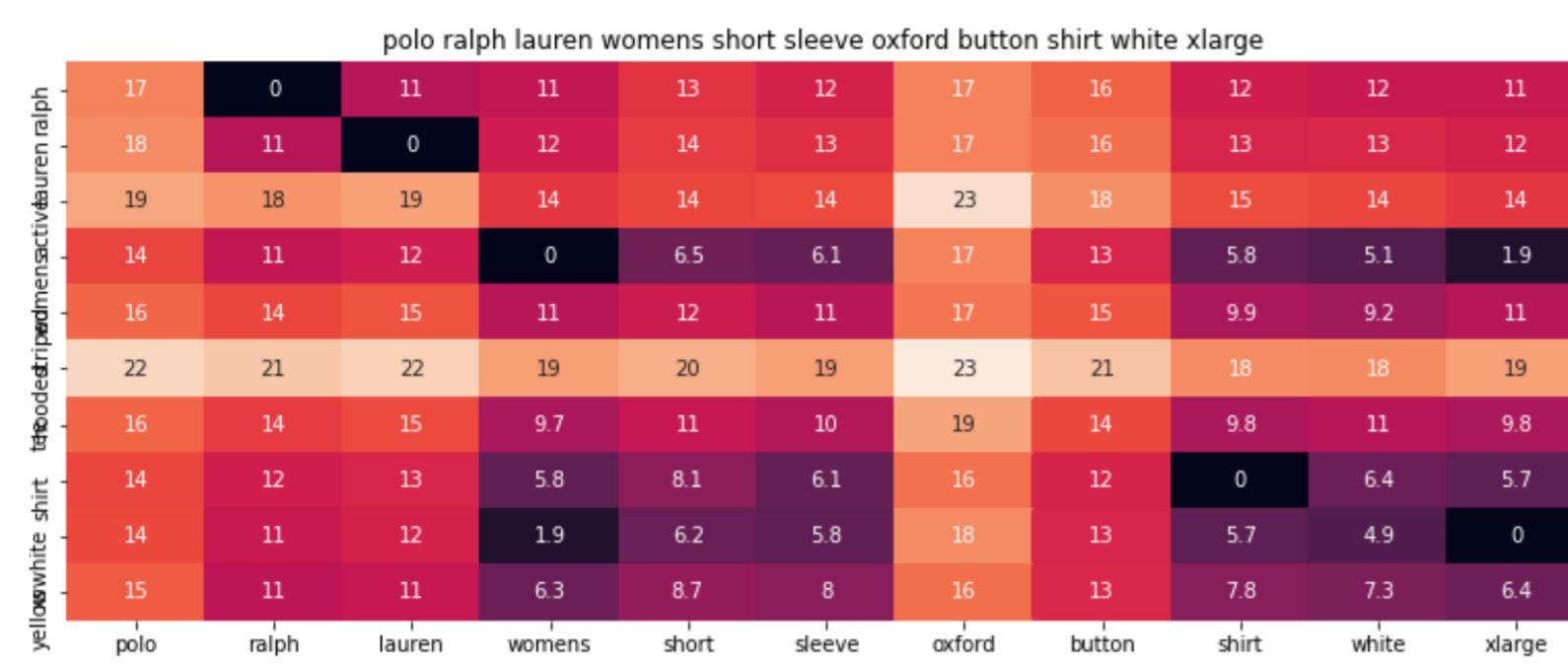
ASIN : B0713MCCFX  
Brand : RALPH LAUREN  
euclidean distance from input : 2.2837339403051997

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01FRIMR5U	RALPH-LAUREN	White	SHIRT



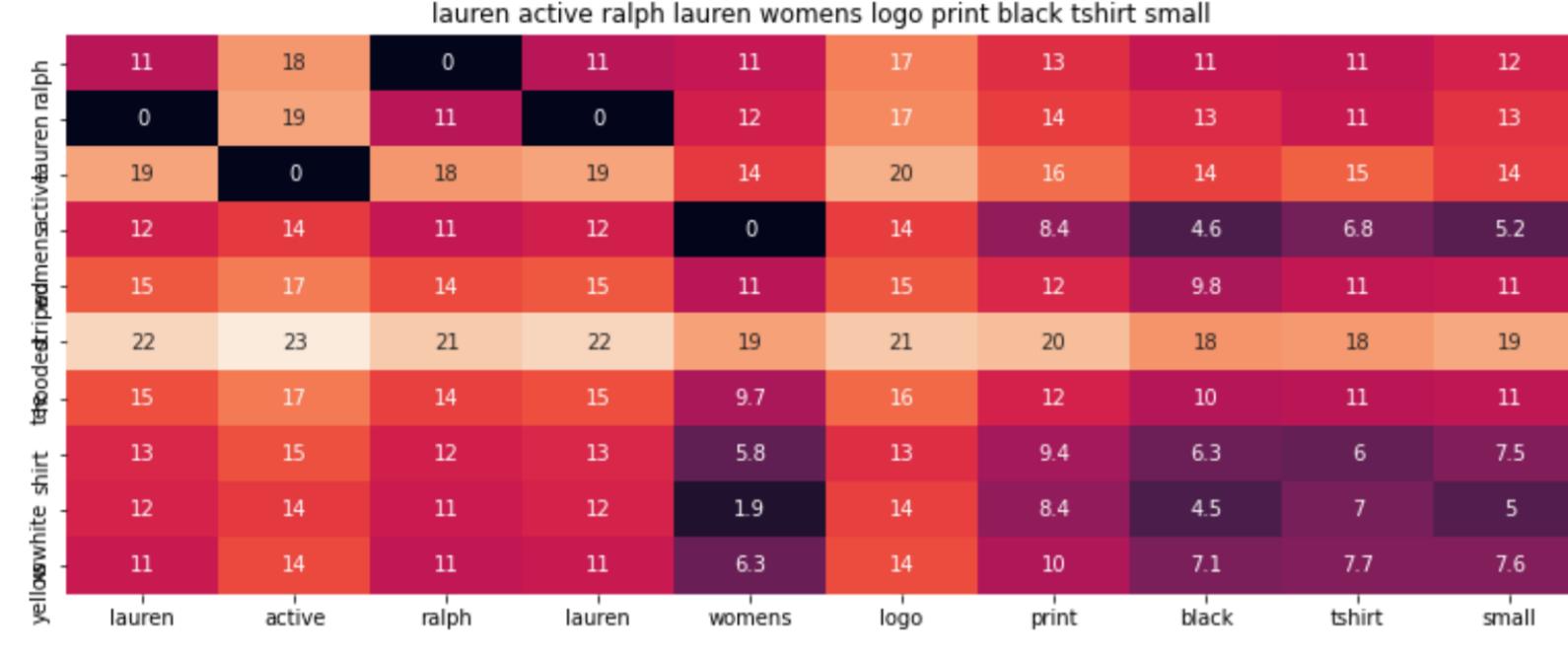
ASIN : B01FRIMRSU  
Brand : RALPH LAUREN  
euclidean distance from input : 2.372298050107446

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01DKUKFR4	RALPH-LAUREN	White	SHIRT



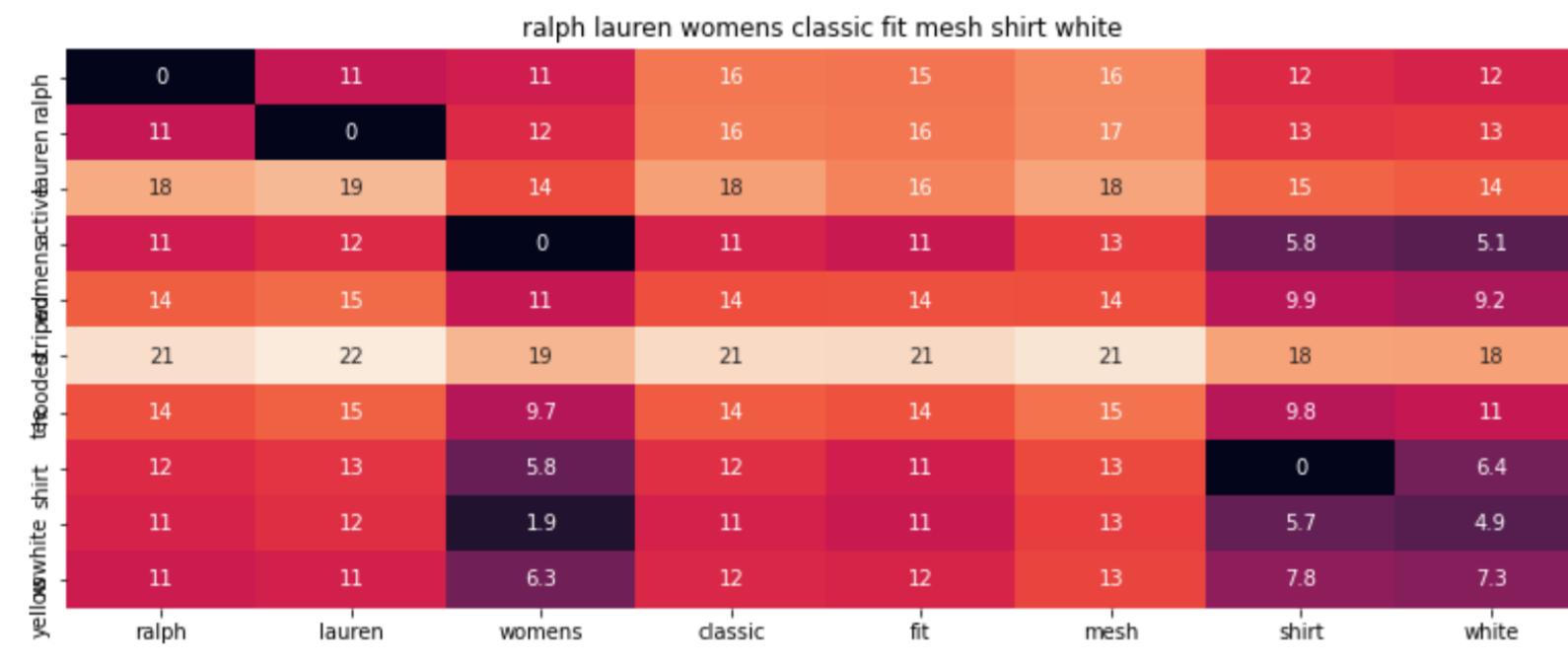
ASIN : B01DKUKFR4  
Brand : RALPH LAUREN  
euclidean distance from input : 2.3738574983496337

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01E1SZIGM	Ralph-Lauren-Active	Black	SHIRT



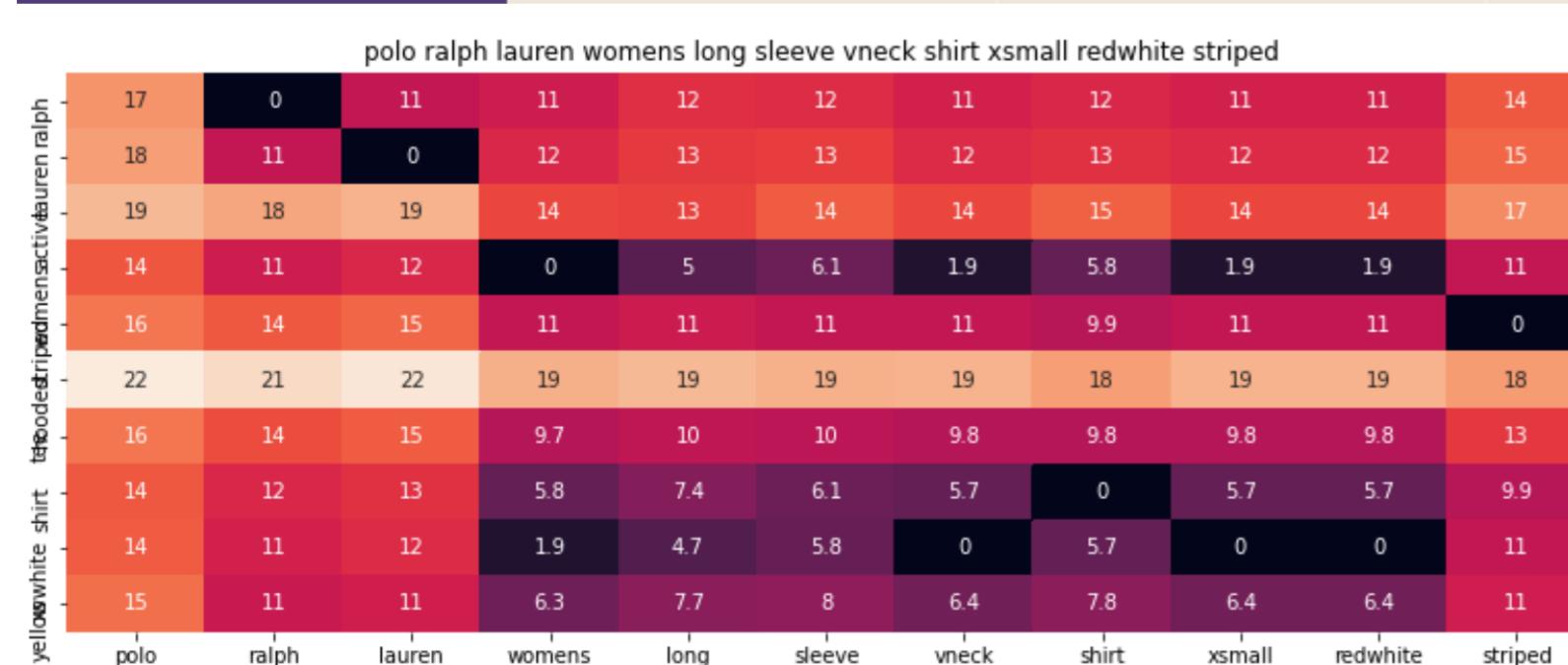
ASIN : B01E1SZIGM  
Brand : Ralph Lauren Active  
euclidean distance from input : 2.5163507974611963

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01KIXF87S	RALPH-LAUREN	White	SHIRT



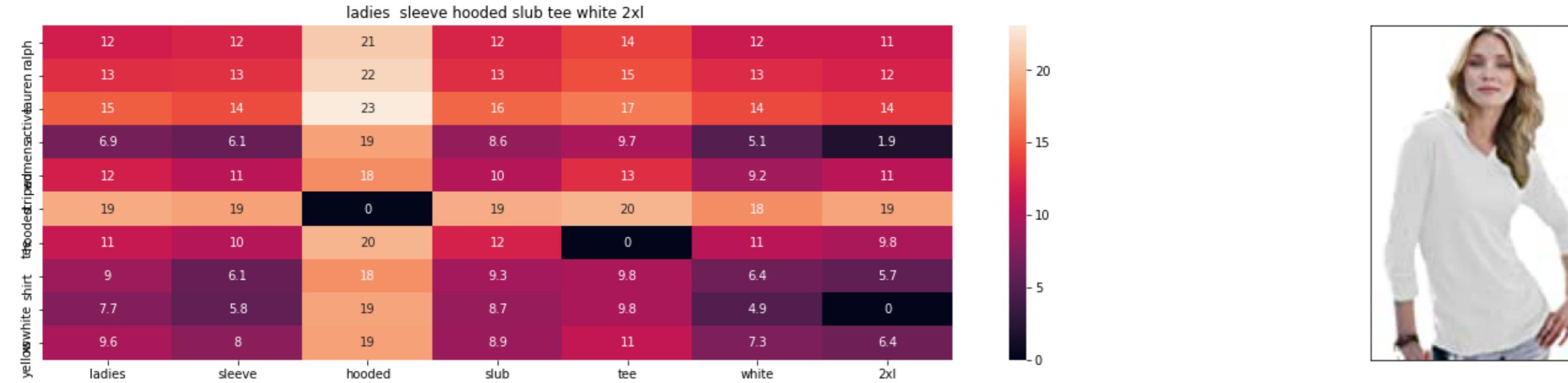
ASIN : B01KIXF87S  
Brand : RALPH LAUREN  
euclidean distance from input : 2.526565742673364

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01MRL7J8F	Polo-Ralph-Lauren	Red-White-Stripes	SHIRT



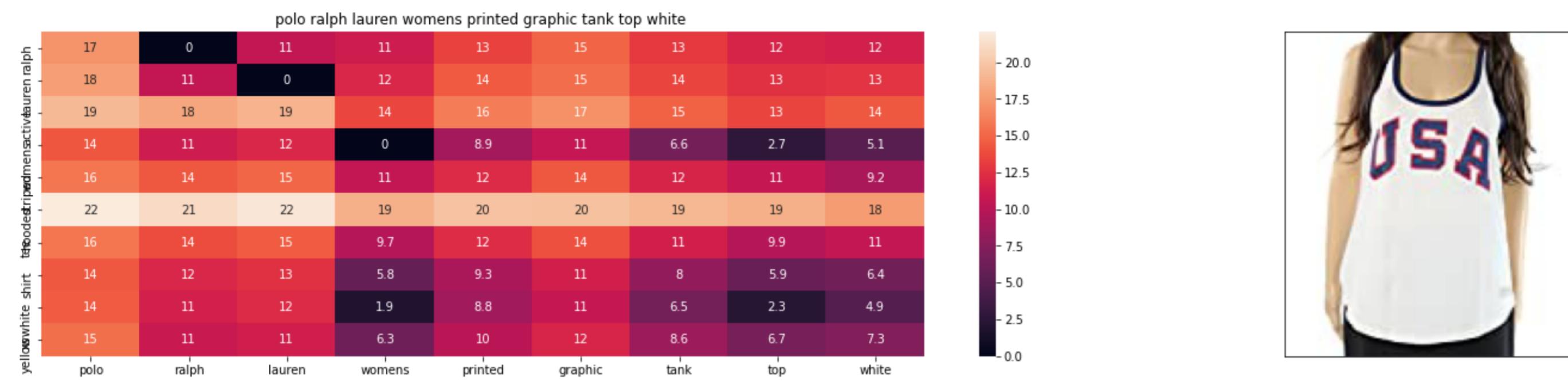
ASIN : B01MRL7J8F  
Brand : Polo Ralph Lauren  
euclidean distance from input : 2.5449358060716722

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B00CO8F6XW	J.-America	White	SHIRT



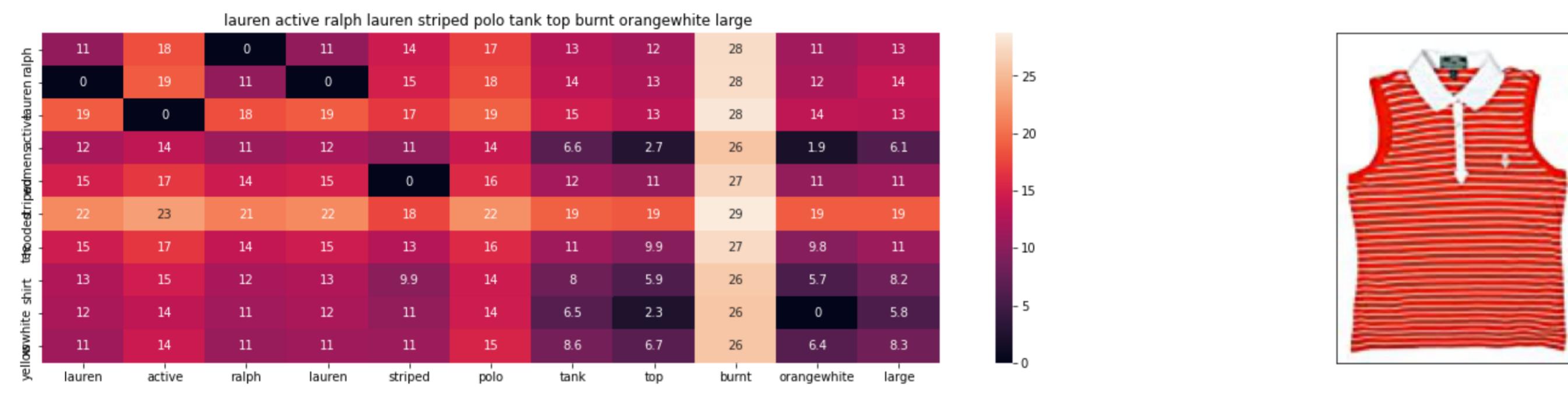
ASIN : B00C08F6XW  
Brand : J. America  
euclidean distance from input : 2.614196975505266

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01NAP3SFK	Polo-Ralph-Lauren	White	SHIRT



ASIN : B01NAP3SFK  
Brand : Polo Ralph Lauren  
euclidean distance from input : 2.615035108374282

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
BO0ILGH5OY	Ralph-Lauren-Active	Burnt-Orange/White	SHIRT



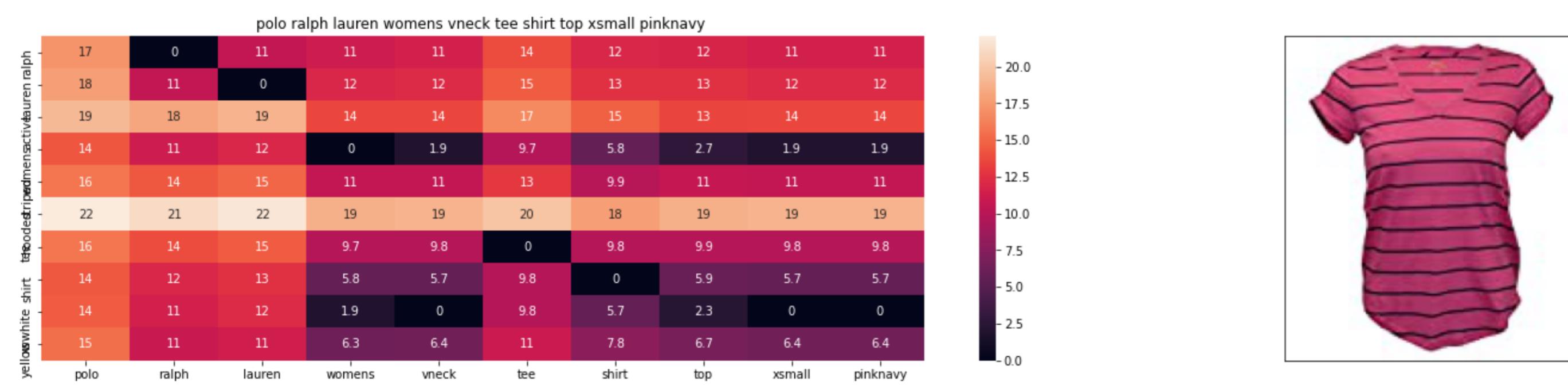
ASIN : B00ILGH50Y  
Brand : Ralph Lauren Active  
euclidean distance from input : 2.6280357873903957

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B06X9ZVRV1	RALPH-LAUREN	Black	SHIRT



ASIN : B06X9ZVRV1  
Brand : RALPH LAUREN  
euclidean distance from input : 2.6555479049682615

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01ETUBK3W	RALPH-LAUREN	Pink/Navy	SHIRT



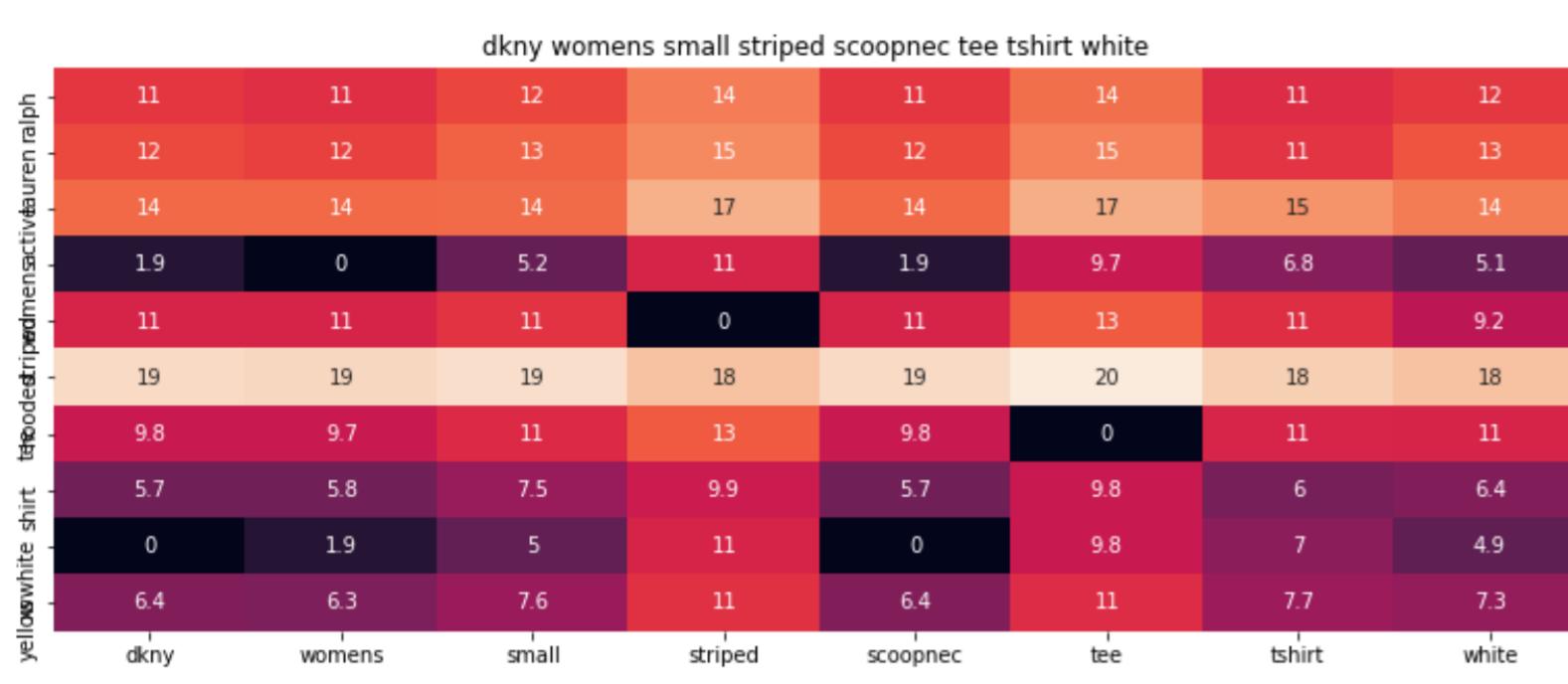
ASIN : B01ETUBK3W  
Brand : RALPH LAUREN  
euclidean distance from input : 2.6744027212022874

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
BO0IMMN726	RALPH-LAUREN	Orange	SHIRT



ASIN : B00IMMN726  
Brand : RALPH LAUREN  
euclidean distance from input : 2.677749824523926

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B0725PWQFW	DKNY	White	SHIRT



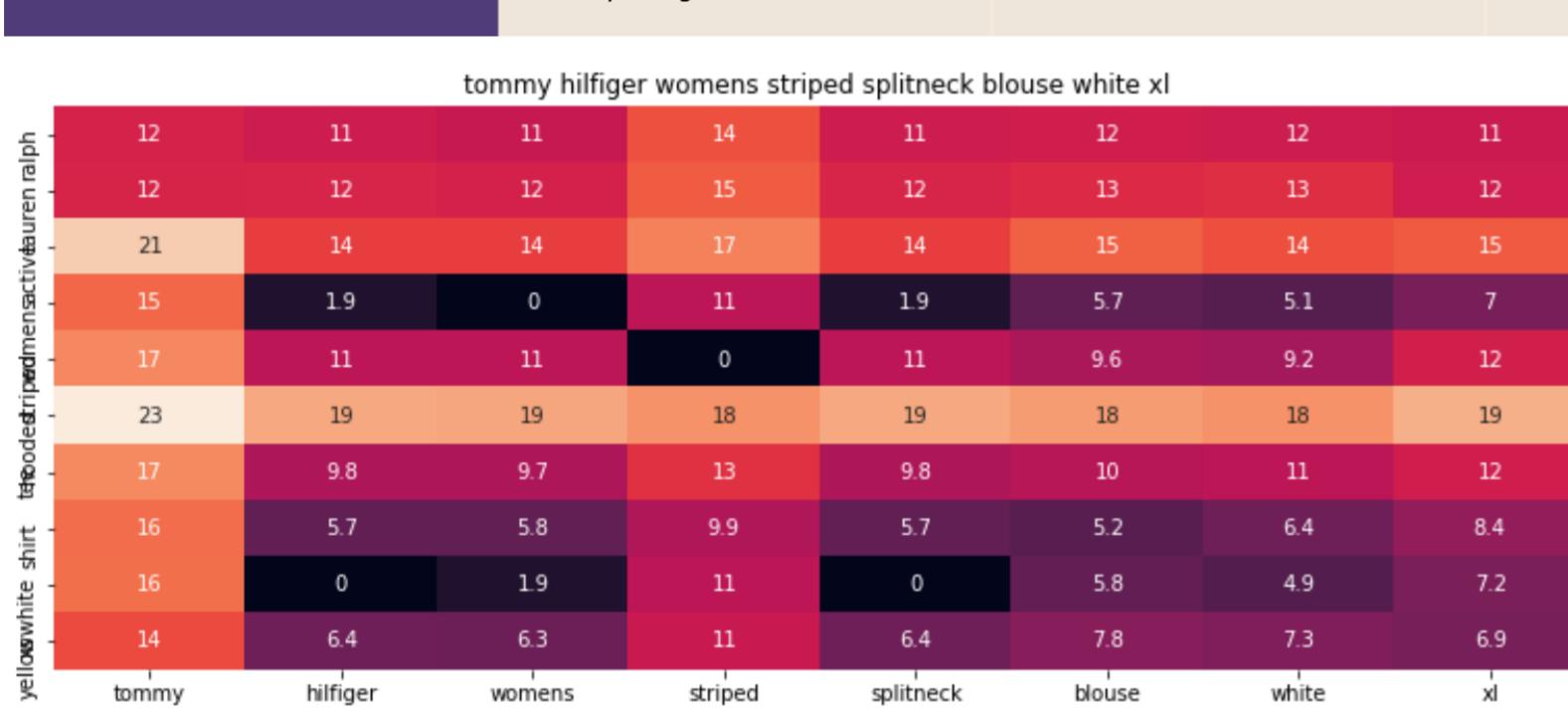
ASIN : B0725PWQFW  
Brand : DKNY  
euclidean distance from input : 2.686597354686174

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01M292SR6	SOLOW	White	SHIRT



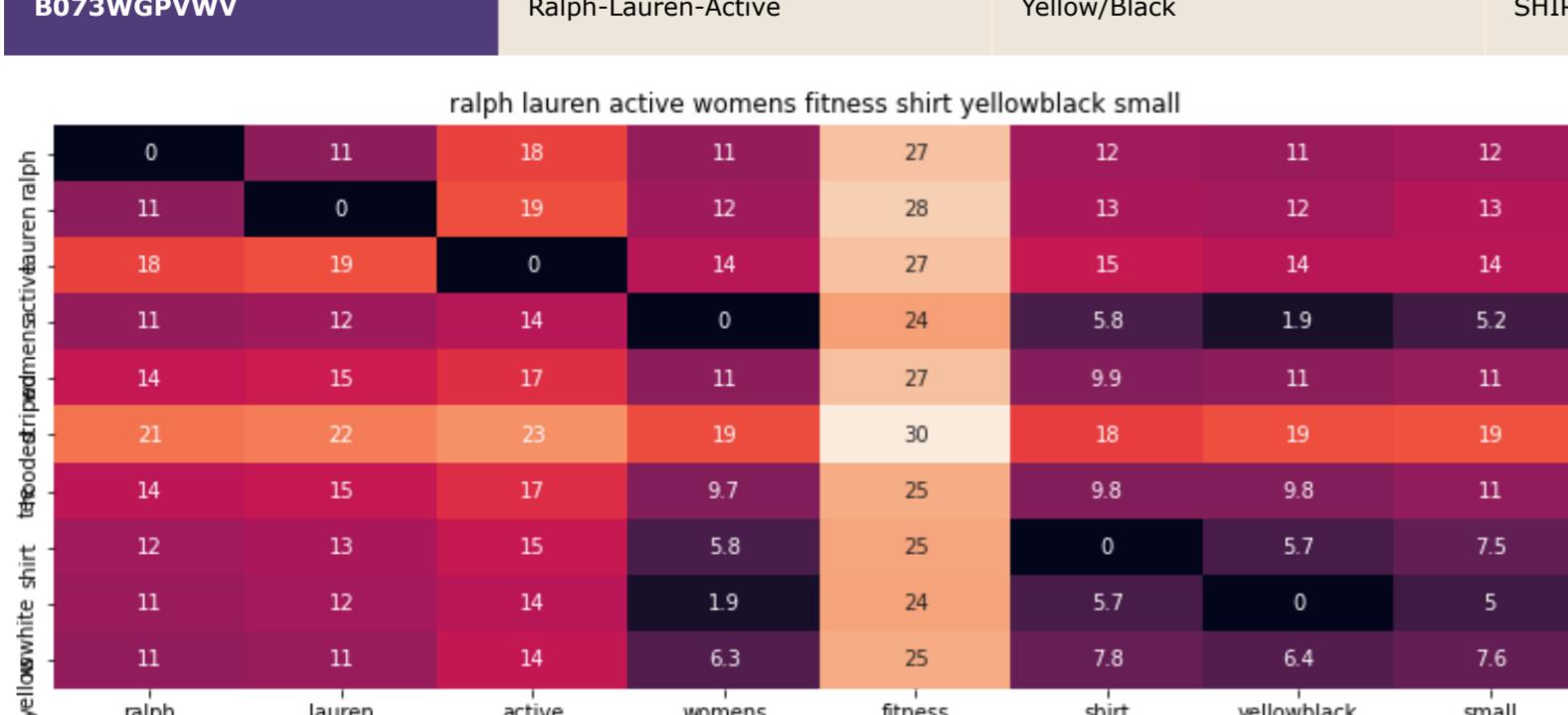
ASIN : B01M292SR6  
Brand : SOLOW  
euclidean distance from input : 2.7090736463426683

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B071VBTCPX	Tommy-Hilfiger	White	SHIRT



ASIN : B071VBTCPX  
Brand : Tommy Hilfiger  
euclidean distance from input : 2.7240975172656126

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B073WGPVWV	Ralph-Lauren-Active	Yellow/Black	SHIRT



ASIN : B073WGPVWV  
Brand : Ralph Lauren Active  
euclidean distance from input : 2.7376338960547115

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B06XKNW8TM	LEERYA	White	SHIRT

leerya women long sleeve tshirt blouse casual loose tops shirt tee xl white

ASIN : B06XKNW8TM  
Brand : LEERYA  
euclidean distance from input : 2.7859026029466722  
=====

## [10.2] Keras and Tensorflow to extract features

```
In [62]: N 1 import numpy as np
 2 from keras.preprocessing.image import ImageDataGenerator
 3 from keras.models import Sequential
 4 from keras.layers import Dropout, Flatten, Dense
 5 from keras import applications
 6 from sklearn.metrics import pairwise_distances
 7 import matplotlib.pyplot as plt
 8 import requests
 9 from PIL import Image
10 from IPython.display import display, Image, SVG, Math, YouTubeVideo
11 import pandas as pd
12 import pickle
```

```
In [2]: M
1 #Load the features and corresponding ASINS info.
2 bottleneck_features_train = np.load('16k_data_cnn_features.npy')
3 asins = np.load('16k_data_cnn_feature_asins.npy')
4 asins = list(asins)
5
6 # load the original 16K dataset
7 data = pd.read_pickle('pickels/16k_apperal_data_preprocessed')
8 df_asins = list(data['asin'])
9
10
11 #get similar products using CNN features (VGG-16)
12 def get_similar_products_cnn(doc_id, num_results):
13     doc_id = asins.index(df_asins[doc_id])
14     pairwise_dist = pairwise_distances(bottleneck_features_train, bottleneck_features_train[doc_id].reshape(1,-1))
15
16     indices = np.argsort(pairwise_dist.flatten())[0:num_results]
17     pdists = np.sort(pairwise_dist.flatten())[0:num_results]
18
19     for i in range(len(indices)):
20         rows = data[['medium_image_url','title']].loc[data['asin']==asins[indices[i]]]
21         for idx, row in rows.iterrows():
22             display(Image(url=row['medium_image_url'], embed=True))
23             print('Product Title: ', row['title'])
24             print('Euclidean Distance from input image: ', pdists[i])
25             print('Amazon Url: www.amazon.com/dp/' + asins[indices[i]])
26
27 get_similar_products_cnn(12500, 20)
28
```



Product Title: ralph lauren active womens striped hooded tee shirt yellowwhite xs  
 Euclidean Distance from input image: 4.862804e-06  
 Amazon Url: [www.amazon.com/dp/B01JME4H6W](http://www.amazon.com/dp/B01JME4H6W)



Product Title: alo ladies junior fit performance mesh polo shirt w1709 large sport athletic gold  
 Euclidean Distance from input image: 41.656384  
 Amazon Url: [www.amazon.com/dp/B00PH3DJC6](http://www.amazon.com/dp/B00PH3DJC6)



Product Title: sporttek ladies long sleeve vneck competitor teexs black lsl353ls  
 Euclidean Distance from input image: 41.85273  
 Amazon Url: [www.amazon.com/dp/B00J9UIK00](http://www.amazon.com/dp/B00J9UIK00)



Product Title: sugarlips womens relaxed fit seamless ribbed tank skin nude  
 Euclidean Distance from input image: 42.043793  
 Amazon Url: [www.amazon.com/dp/B00IJHSY54](http://www.amazon.com/dp/B00IJHSY54)



Product Title: white lesbian super flat chest builtin elastic band binder slim fit tank tops xl  
 Euclidean Distance from input image: 42.374218  
 Amazon Url: [www.amazon.com/dp/B00NBGYAH4](http://www.amazon.com/dp/B00NBGYAH4)



Product Title: miraclebody womens jersey slimming tunic top black  
 Euclidean Distance from input image: 42.843918  
 Amazon Url: [www.amazon.com/dp/B0059GPDE](http://www.amazon.com/dp/B0059GPDE)



Product Title: 100 cotton camisole 4pack size extra large 3x  
 Euclidean Distance from input image: 43.36531  
 Amazon Url: [www.amazon.com/dp/B01B9RBRVS](http://www.amazon.com/dp/B01B9RBRVS)



Product Title: 100 cotton camisole 4pack assorted size extra large 3x  
 Euclidean Distance from input image: 43.36531  
 Amazon Url: [www.amazon.com/dp/B01B9REAHQ](http://www.amazon.com/dp/B01B9REAHQ)



Product Title: alo sport womens 3button mesh polo shirt crlna blue medium  
 Euclidean Distance from input image: 43.60738  
 Amazon Url: [www.amazon.com/dp/B001M7XQ40](http://www.amazon.com/dp/B001M7XQ40)



Product Title: one femme womens solid cotton embroidered top xlalge white 001  
 Euclidean Distance from input image: 43.652073  
 Amazon Url: [www.amazon.com/dp/B01DLSOL1G](http://www.amazon.com/dp/B01DLSOL1G)



Product Title: alo sport ladies bamboo racerback tank pinkwhite xs  
 Euclidean Distance from input image: 43.812096  
 Amazon Url: [www.amazon.com/dp/B004J8LKP8](http://www.amazon.com/dp/B004J8LKP8)



Product Title: bb dakota womens plus size tereza top optic white 2x  
 Euclidean Distance from input image: 44.02845  
 Amazon Url: [www.amazon.com/dp/B00JG6EBA](http://www.amazon.com/dp/B00JG6EBA)



Product Title: slinky 34slv ruffle scoopneck tee 532020 white l  
 Euclidean Distance from input image: 44.035717  
 Amazon Url: [www.amazon.com/dp/B0759W1F8F](http://www.amazon.com/dp/B0759W1F8F)



Product Title: red house ladies french cuff noniron pinpoint oxford white xs  
 Euclidean Distance from input image: 44.181805  
 Amazon Url: [www.amazon.com/dp/B0090A7JX2](http://www.amazon.com/dp/B0090A7JX2)



Product Title: feel piece womens charlotte ribbed hi lo knit tunic top sz xss black 270075e  
 Euclidean Distance from input image: 44.194263  
 Amazon Url: [www.amazon.com/dp/B074T2TTPL](http://www.amazon.com/dp/B074T2TTPL)



Product Title: milkku womens long sleeve cold shoulder shirts cross backless party top black  
 Euclidean Distance from input image: 44.239902  
 Amazon Url: [www.amazon.com/dp/B01I2WW6PE](http://www.amazon.com/dp/B01I2WW6PE)



Product Title: milkku womens long sleeve cold shoulder shirts party top black  
 Euclidean Distance from input image: 44.239902  
 Amazon Url: [www.amazon.com/dp/B01I2WW7NK](http://www.amazon.com/dp/B01I2WW7NK)



Product Title: women39s ua coldgear174 infrared crew  
 Euclidean Distance from input image: 44.285942  
 Amazon Url: [www.amazon.com/dp/B00GOAMNLU](http://www.amazon.com/dp/B00GOAMNLU)



Product Title: cloth stone split back washed tee medium stone  
 Euclidean Distance from input image: 44.40766  
 Amazon Url: [www.amazon.com/dp/B06XTBW9MQ](http://www.amazon.com/dp/B06XTBW9MQ)



Product Title: guayabita womens alba black blouse  
 Euclidean Distance from input image: 44.513775  
 Amazon Url: [www.amazon.com/dp/B01JM4Z03M](http://www.amazon.com/dp/B01JM4Z03M)

#### Observations

- Using CNN architecture vgg-16 , we used image as our input to produce similar products,we see that we are able to capture similar shirts well .We can see that we are able to capture long-sleeves, plain colors and yellow colored products well.

#### Assignment

----> let's use all the features text,brand,color,image ,give weights to all the features calculate the euclidean distance

```
In [68]: 1 # some of the brand values are empty.
2 # Need to replace Null with string "NULL"
3 data['brand'].fillna(value="Not given", inplace=True )
4
5 # replace spaces with hyphen
6 brands = [x.replace(" ", "-") for x in data['brand'].values]
7 color = [x.replace(" ", "-") for x in data['color'].values]
8
9 brand_vectorizer = CountVectorizer()
10 brand_features = brand_vectorizer.fit_transform(brands)
11
12 color_vectorizer = CountVectorizer()
13 color_features = color_vectorizer.fit_transform(color)
14
15 vg16_features = np.load('16k_data_cnn_features.npy')
16
17 extra_features = hstack((w2v_title_weight,brand_features, color_features,vg16_features)).tocsr()
```

```
In [89]: M 1 def heat_map_w2v_brand(sentance1, sentance2, url, doc_id1, doc_id2, df_id1, df_id2, model):  
2     # sentance1 : title1, input apparel  
3     # sentance2 : title2, recommended apparel  
4     # url: apparel image url  
5     # doc_id1: document id of input apparel  
6     # doc_id2: document id of recommended apparel  
7     # df_id1: index of document1 in the data frame  
8     # df_id2: index of document2 in the data frame  
9     # model: it can have two values, 1. avg 2. weighted  
10    #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/avg) of length 300 corresponds to each word in give title  
11    s1_vec = get_word_vec(sentance1, doc_id1, model)  
12    #s2_vec = np.array(#number_of_words_title2 * 300), each row is a vector(weighted/avg) of length 300 corresponds to each word in give title  
13    s2_vec = get_word_vec(sentance2, doc_id2, model)  
14  
15    # s1_s2_dist = np.array(#number of words in title1 * #number of words in title2)  
16    # s1_s2_dist[i,j] = euclidean distance between words i, j  
17    s1_s2_dist = get_distance(s1_vec, s2_vec)  
18  
19    data_matrix = [['Asin','Brand', 'Color', 'Product type'],  
20                  [data['asin'].loc[df_id1].brands[doc_id1], color[doc_id1], types[doc_id1]], # input apparel's features  
21                  [data['asin'].loc[df_id2].brands[doc_id2], color[doc_id2], types[doc_id2]]] # recommended apparel's features  
22  
23    colorscale = [[0, '#1d004d'], [.5, '#f2e5ff'], [1, '#f2e5d1']] # to color the headings of each column  
24  
25    # we create a table with the data_matrix  
26    table = ff.create_table(data_matrix, index=True, colorscale=colorscale)  
27    # plot it with plotly  
28    plotly.offline.iplot(table, filename='simple_table')  
29  
30    # devide whole figure space into 25 * 1:10 grids  
31    gs = gridspec.GridSpec(25, 15)  
32    fig = plt.figure(figsize=(25,5))  
33    # in first 25*10 grids we plot heatmap  
34    ax1 = plt.subplot(gs[:, :-5])  
35    # plotting the heatmap based on the pairwise distances  
36    ax1 = sns.heatmap(np.round(s1_s2_dist,3), annot=True)  
37  
38    # np.concatenate((sentance2.split(), [sentance2.split()[0]]))  
39    # np.concatenate((sentance1.split(), [sentance1.split()[0]]))  
40    ax1.set_xticklabels(sentance2.split())  
41    # set the y axis labels as input apparels title  
42    ax1.set_yticklabels(sentance1.split())  
43    # set title as recommended apparels title  
44    ax1.set_title(sentance2)  
45  
46    # set title as recommended apparels title  
47    ax1.set_title(sentance2)  
48    #print(len)  
49    # in last 25 * 10:15 grids we display image  
50    ax2 = plt.subplot(gs[:, 10:16])  
51  
52    # we dont display grid lines and axis labels to images  
53    ax2.grid(False)  
54    ax2.set_xticks([])  
55    ax2.set_yticks([])  
56  
57    # pass the url it display it  
58    display_img(url, ax2, fig)  
59  
60    plt.show()  
61  
62
```

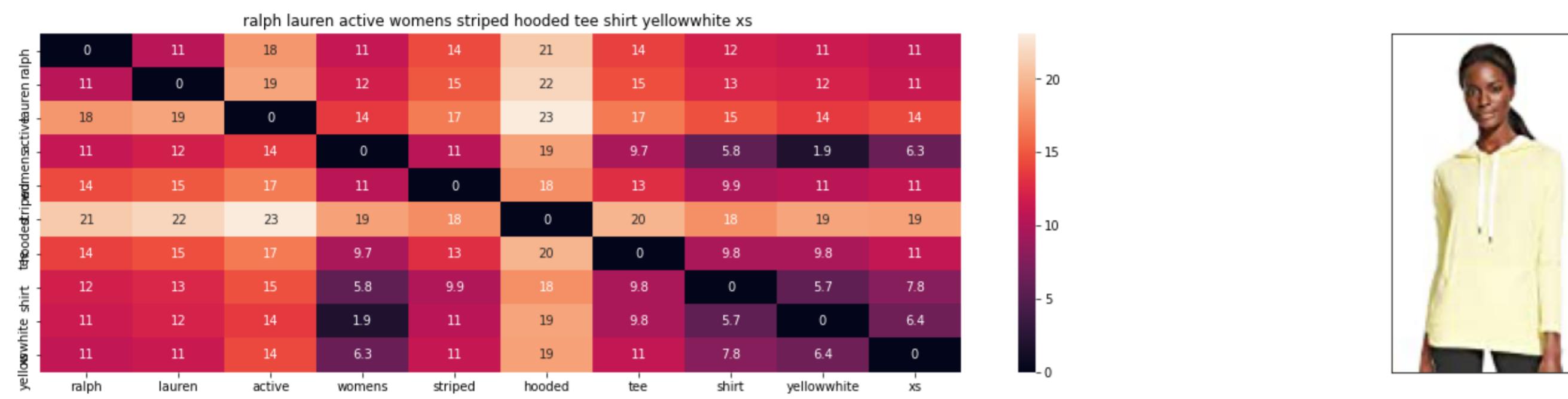
In [99]:

```

1 def idf_w2v_brand(doc_id, w1, w2, w3, w4, num_results):
2     # doc_id: apparel's id in given corpus
3     # w1: weight for w2v features
4     # w2: weight for brand and color features
5
6     # pairwise_dist will store the distance from given input apparel to all remaining apparels
7     # the metric we used here is cosine, the coside distance is measured as K(X, Y) = <X, Y> / (||X|| * ||Y||)
8     # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
9     idf_w2v_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))
10    color_dist = pairwise_distances(color_features, color_features[doc_id])
11    brand_dist = pairwise_distances(brand_features, brand_features[doc_id])
12    vg16_dist = pairwise_distances(vg16_features, vg16_features[doc_id].reshape(1,-1))
13
14    pairwise_dist = (w1 * idf_w2v_dist + w2 * color_dist + w3 * brand_dist + w4 * vg16_dist)/float(w1 + w2 + w3 + w4)
15
16    # np.argsort will return indices of 9 smallest distances
17    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
18    # pdists will store the 9 smallest distances
19    pdists = np.sort(pairwise_dist.flatten())[0:num_results]
20
21    # data frame indices of the 9 smallest distance's
22    df_indices = list(data.index[indices])
23
24
25    for i in range(0, len(indices)):
26        if data['asin'].loc[df_indices[i]] != 'B00PVS29HB':
27            heat_map_w2v_brand(data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]],
28            data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i], df_indices[i], 'weighted')
29            print('ASIN :', data['asin'].loc[df_indices[i]])
30            print('Brand :', data['brand'].loc[df_indices[i]])
31            print('euclidean distance from input :', round(pdists[i], 3))
32            print('=*125')
33    idf_w2v_brand(12500, 10, 5, 5, 15, 20)
34    # in the give heat map, each cell contains the euclidean distance between words i, j

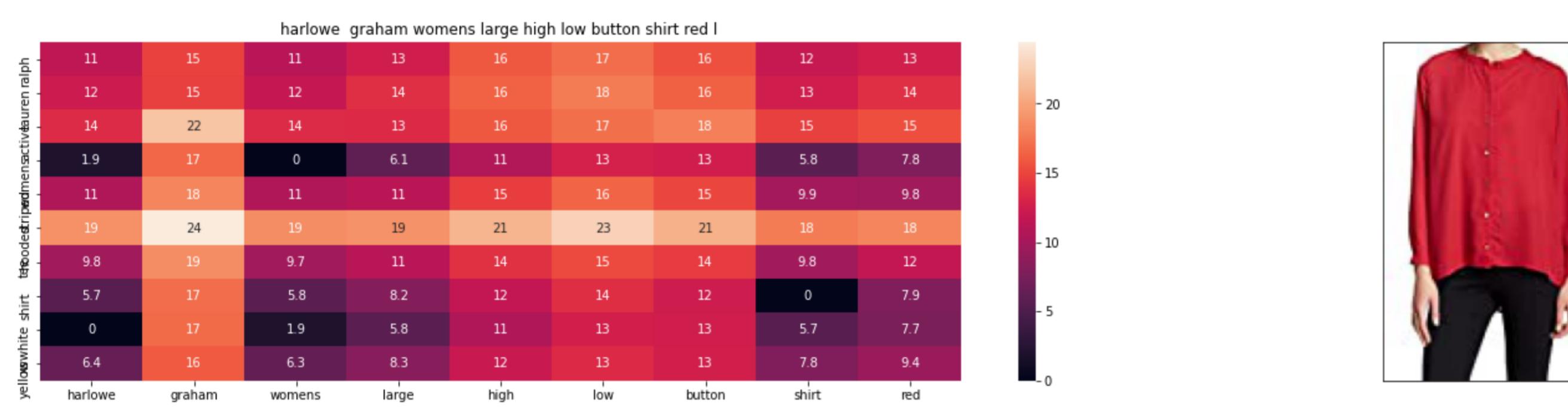
```

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT



ASIN : B01JME4H6W  
Brand : Ralph Lauren Active  
euclidean distance from input : 0.0

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B074T32Q46	Harlowe-&Graham	Red	SHIRT



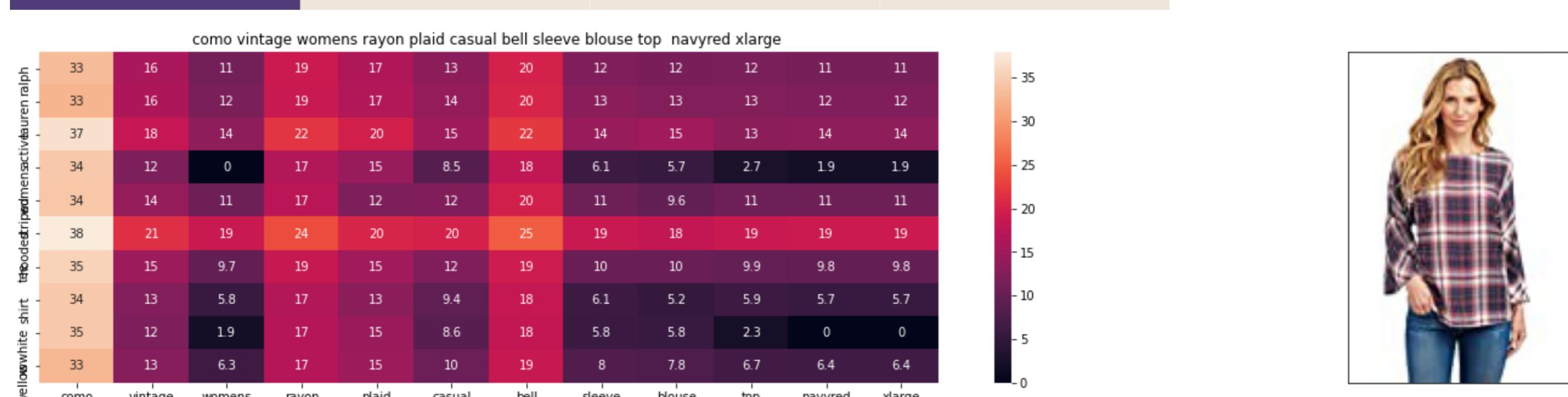
ASIN : B074T32Q46  
Brand : Harlowe & Graham  
euclidean distance from input : 1.802

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01H5L3CNI	Shawhua	Blue	SHIRT



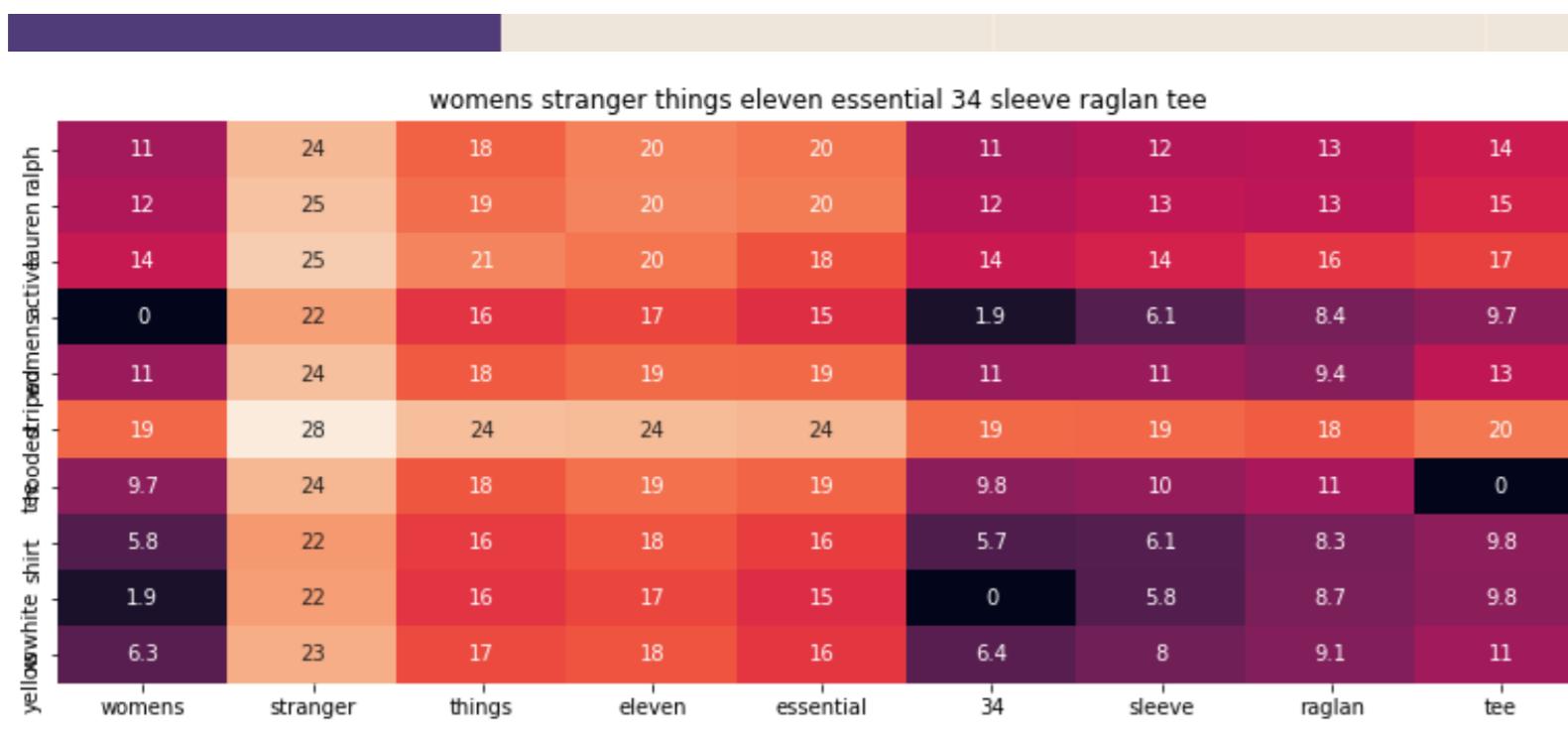
ASIN : B01H5L3CNI  
Brand : Shawhua  
euclidean distance from input : 1.86

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B074TM1HJB	Como-vintage	Navy/Red	SHIRT



ASIN : B074TM1HJB  
Brand : Como vintage  
euclidean distance from input : 1.93

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01LVWWXY2	JOWO88	Black	SHIRT

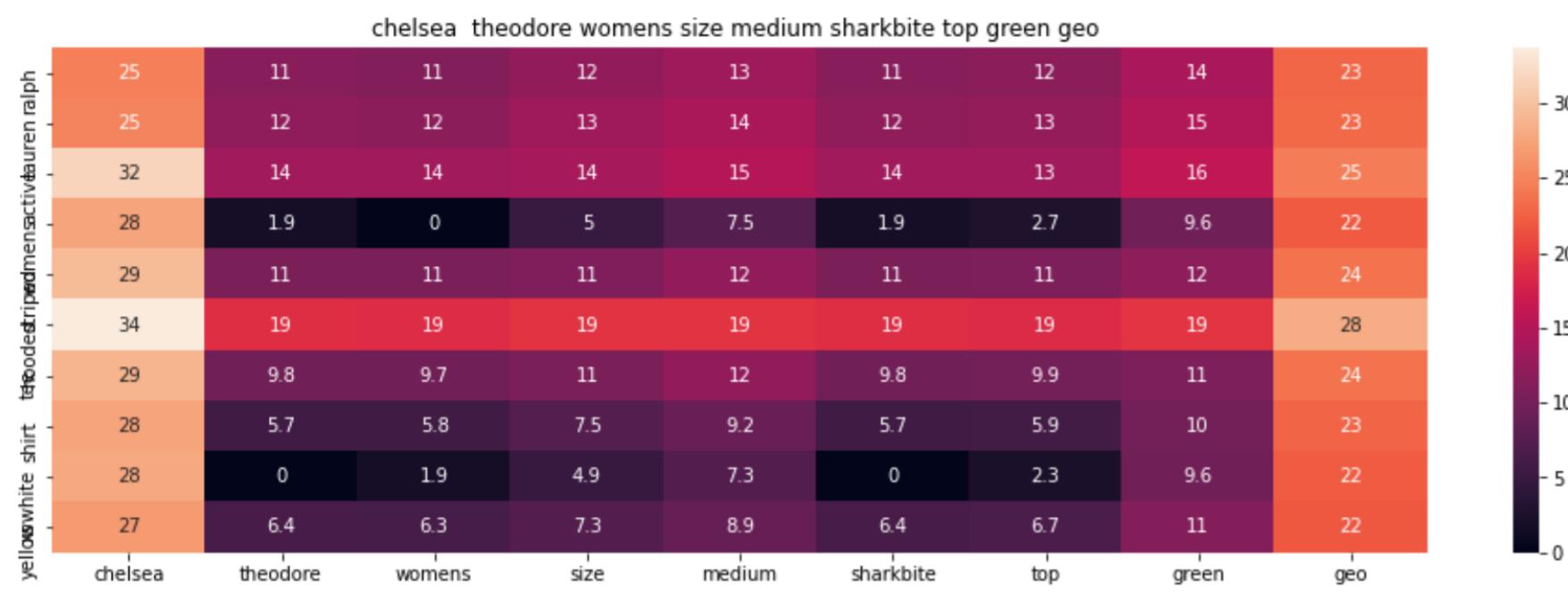


ASIN : B01LVWXY2

Brand : JONW088

euclidean distance from input : 1.943

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B0759NHC93	Chelsea-&-Theodore	Green-Geo	SHIRT



ASIN : B0759NHC93

Brand : Chelsea &amp; Theodore

euclidean distance from input : 2.012

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01BR1Q4PU	Vanssty	Pink	BOOKS_1973_AND_LATER

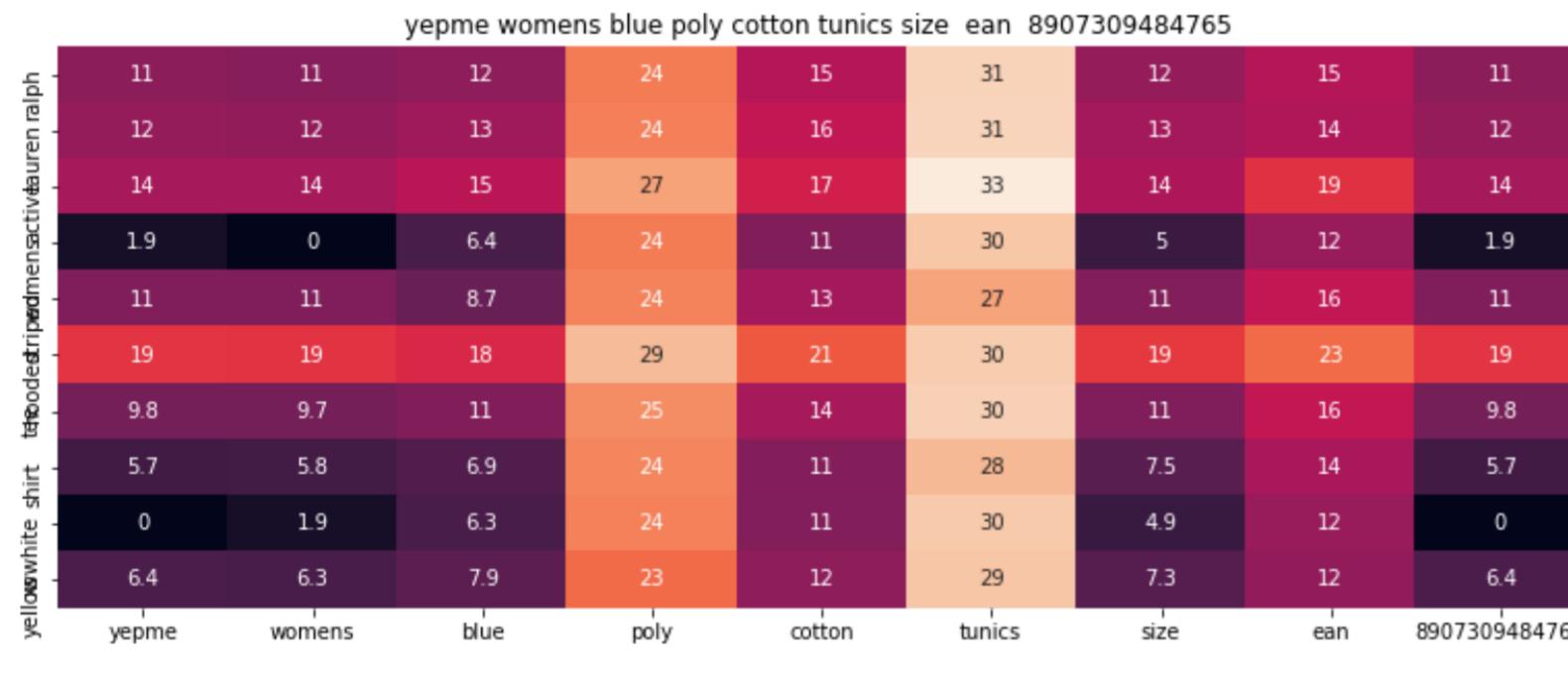


ASIN : B01BR1Q4PU

Brand : Vanssty

euclidean distance from input : 2.0

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01M12O3WS	Yepme	Blue	SHIRT

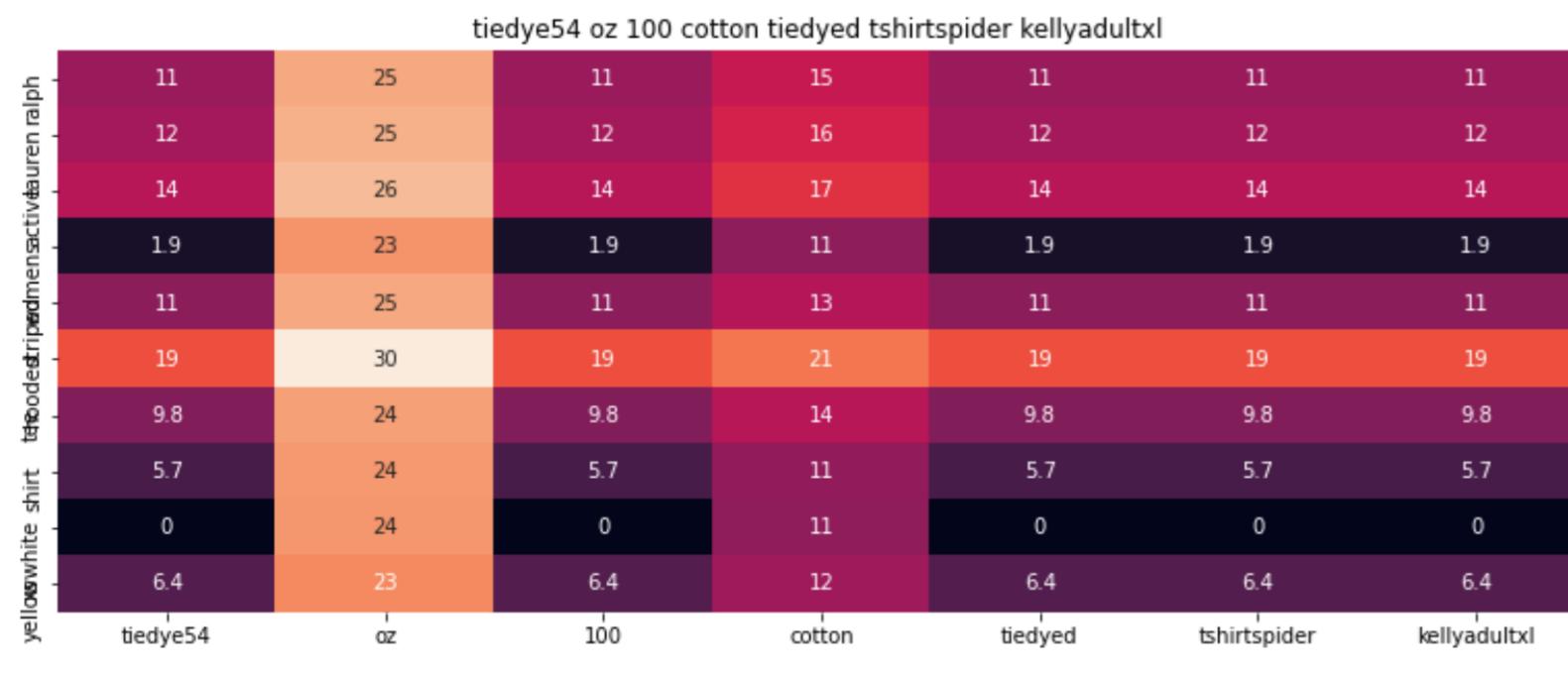


ASIN : B01M12O3WS

Brand : Yepme

euclidean distance from input : 2.077

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B003JE7JZS	tie-dye	Kelly	SHIRT



ASIN : B003JE7JZS

Brand : tie dye

euclidean distance from input : 2.12

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B06ZZMXP8K	EP-Pro	Blue	SHIRT

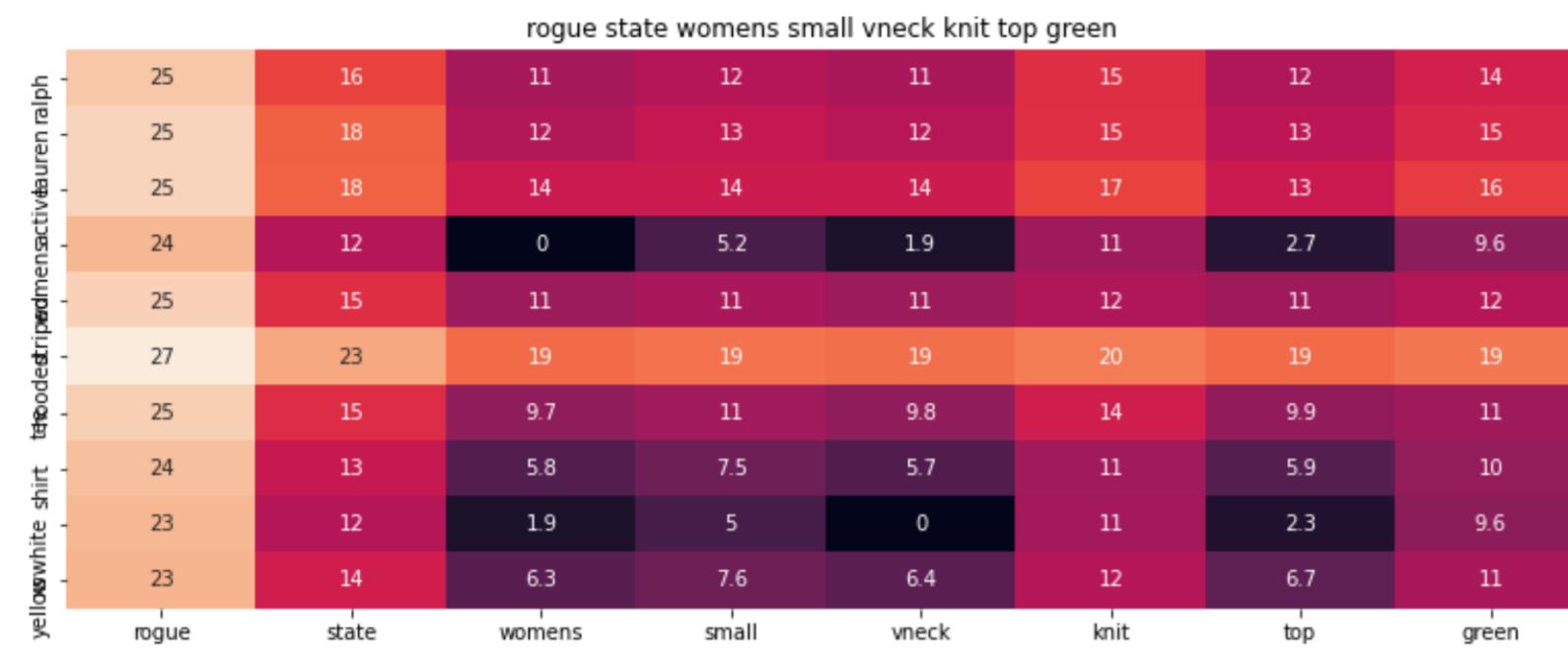


ASIN : B06ZZMXP8K

Brand : EP Pro

euclidean distance from input : 2.238

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01MV37DJE	Rogue-State	Green	SHIRT



ASIN : B01MV37DJE

Brand : Rogue State

euclidean distance from input : 18.609

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01DAGTXS0	LuckyBrand	Blue	SHIRT



ASIN : B01DAGTXS0

Brand : LuckyBrand

euclidean distance from input : 18.687

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B01MY1Z5SK	SIFINI	White	SHIRT

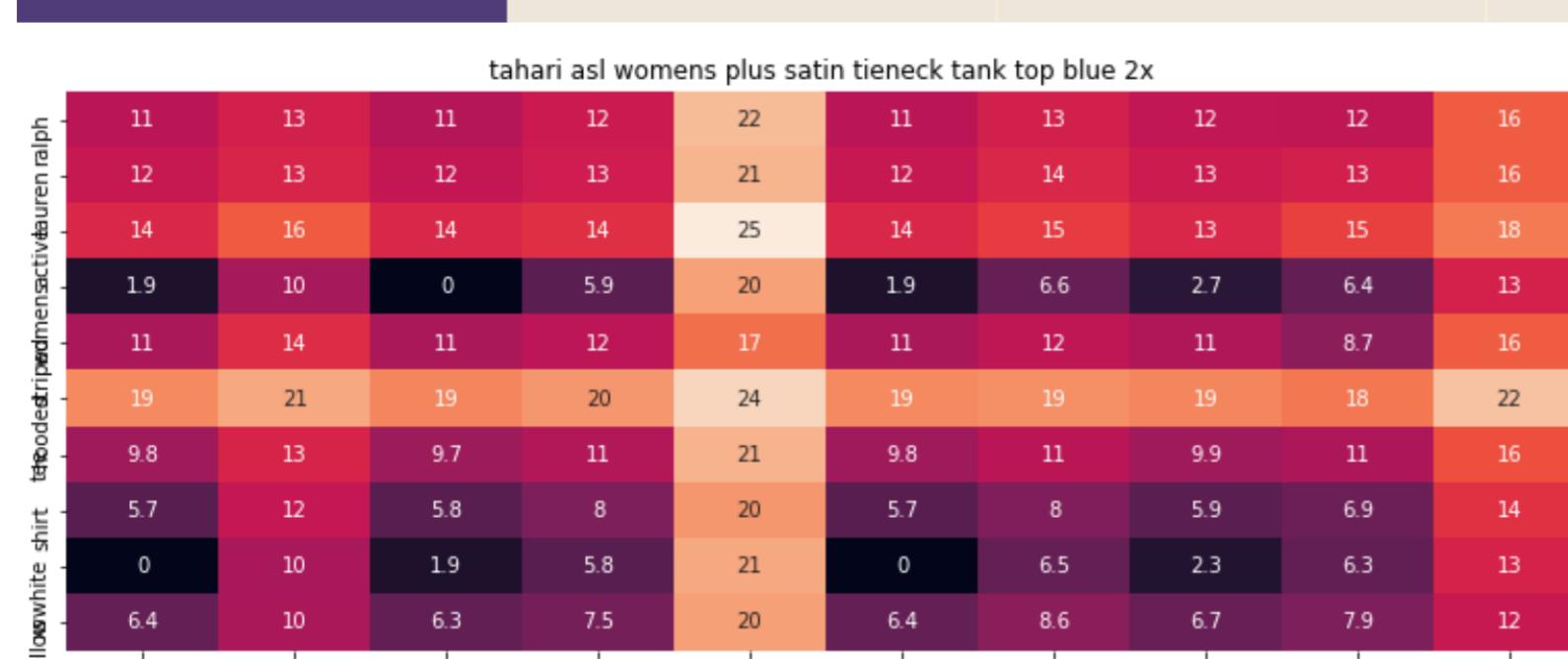


ASIN : B01MY1Z5SK

Brand : SIFINI

euclidean distance from input : 19.475

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B06WLMWCS7	Tahari-by-Arthur-S.-Levine	Blue	SHIRT

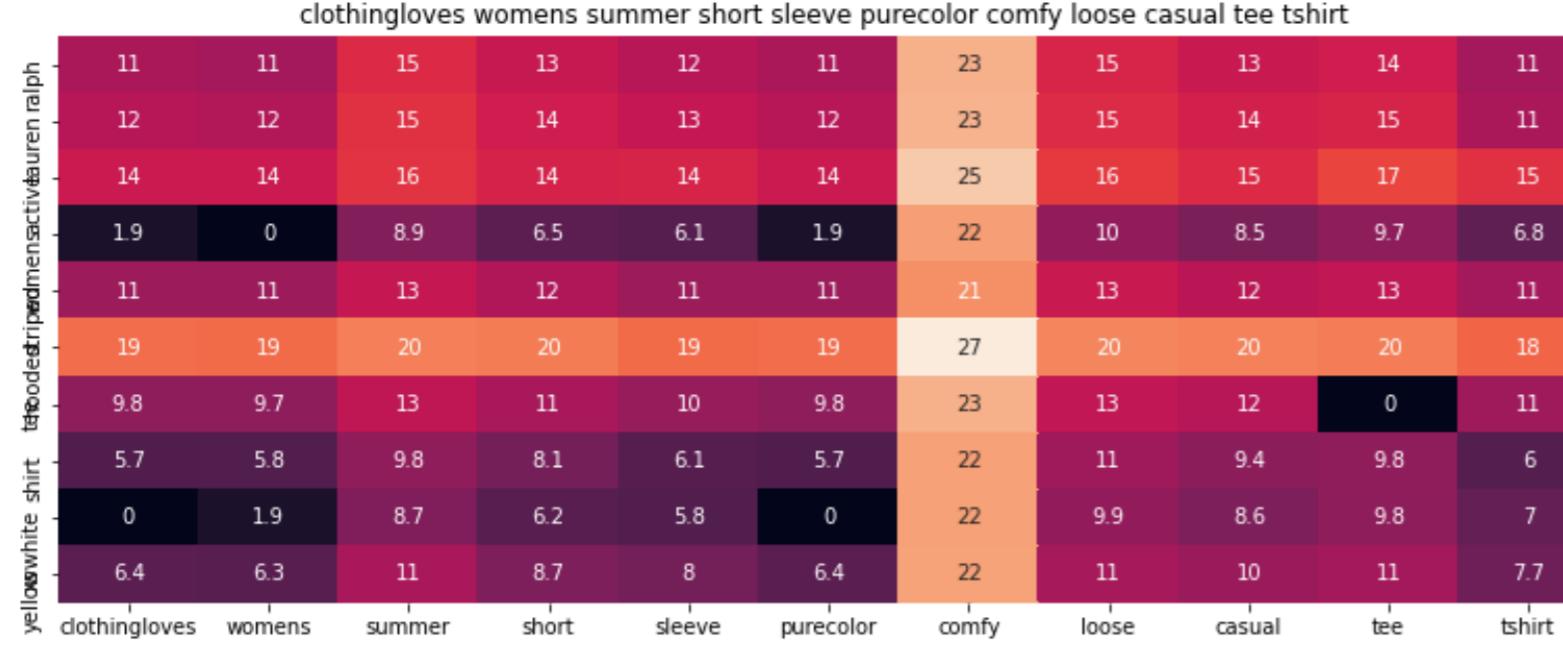


ASIN : B06WLMWCS7

Brand : Tahari by Arthur S. Levine

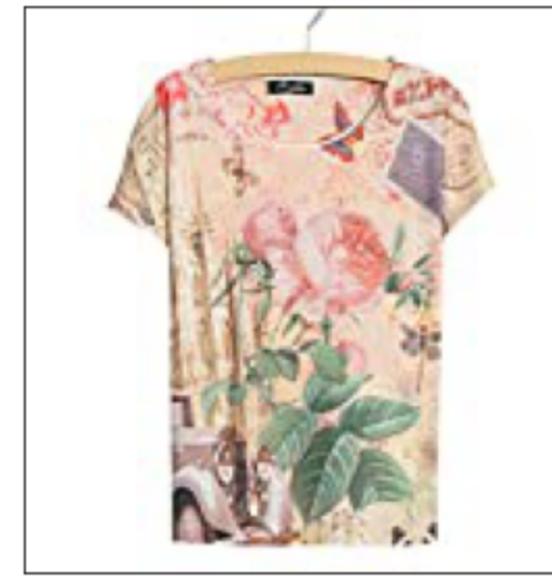
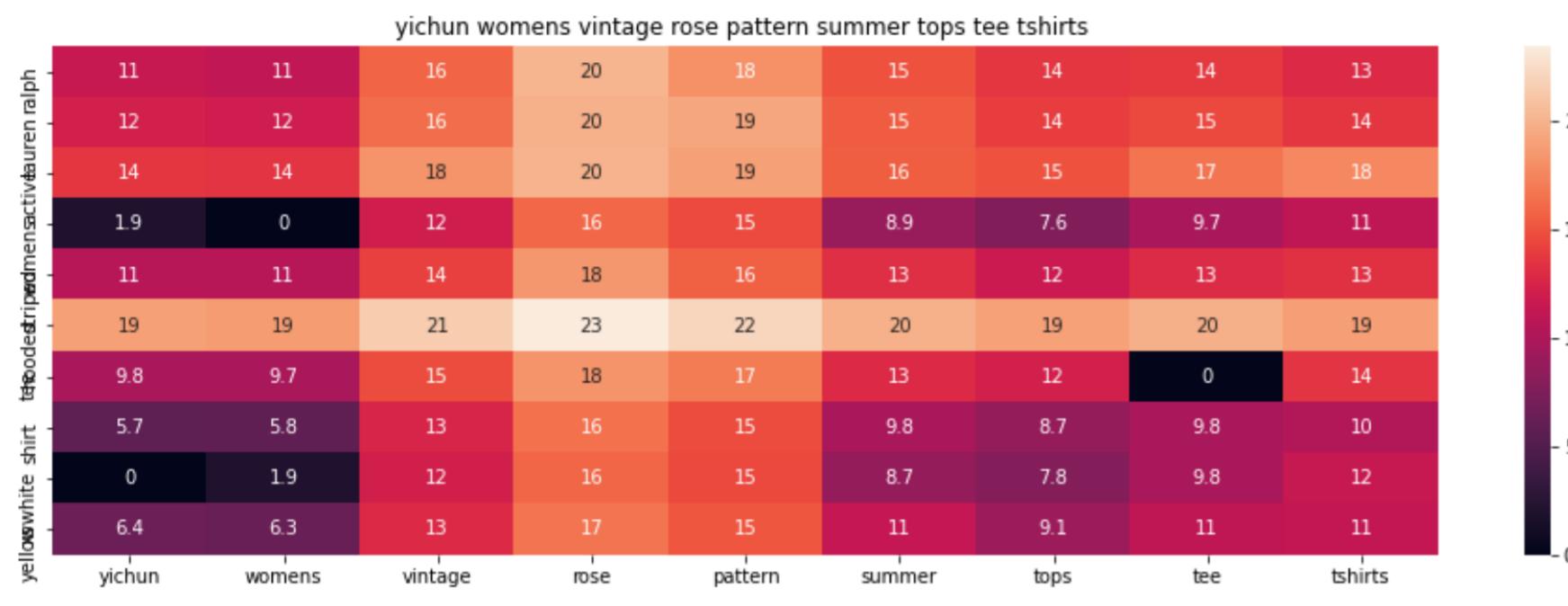
euclidean distance from input : 19.616

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B0746G8DGZ	ClothingLoves	Rose	SHIRT



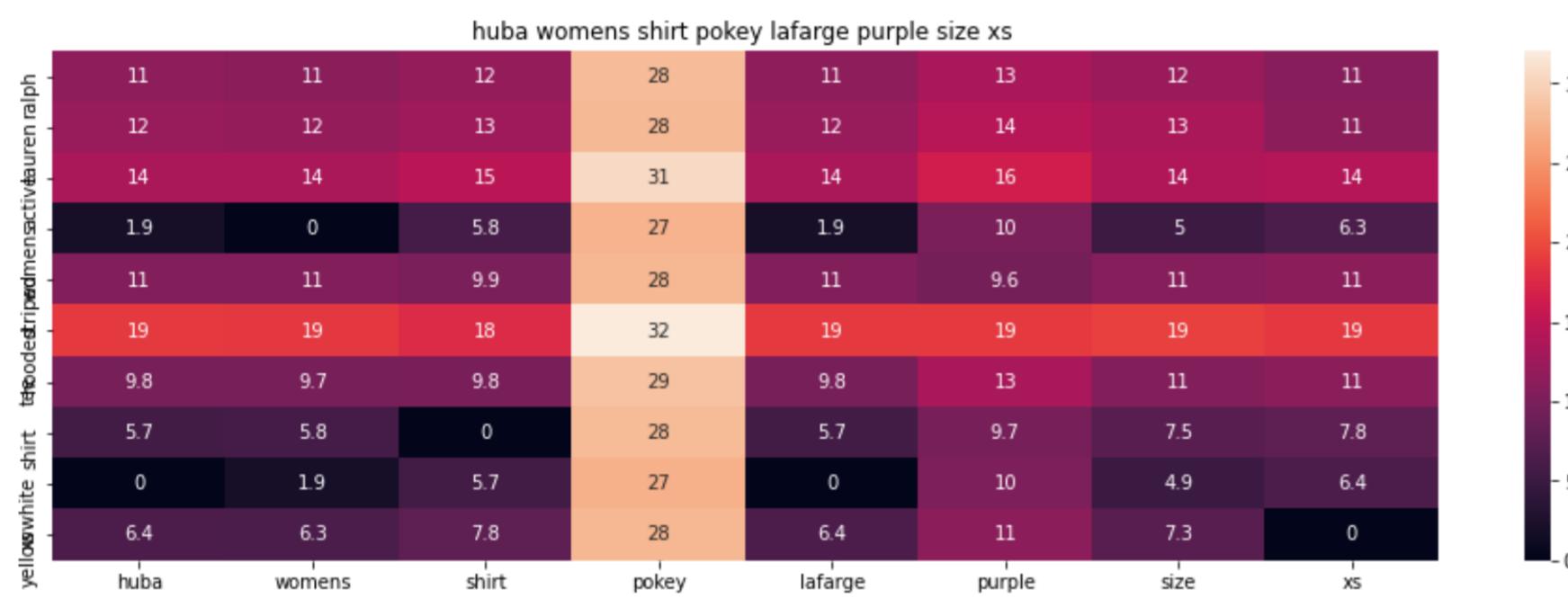
ASIN : B0746G8DGZ  
Brand : ClothingLoves  
euclidean distance from input : 19.814

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B010NN9SZG	YICHUN	Multicoloured	SHIRT



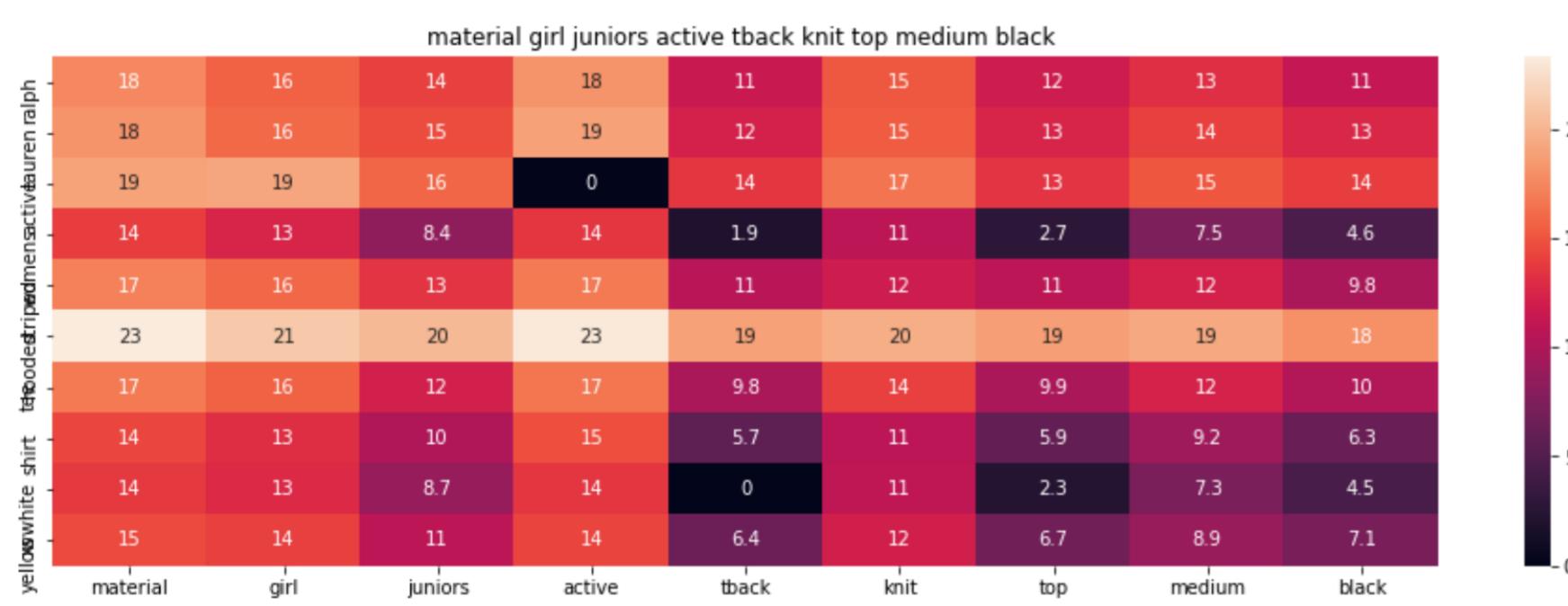
ASIN : B010NN9SZG  
Brand : YICHUN  
euclidean distance from input : 19.902

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B017X00EDW	HUBA-Design	Purple	BOOKS_1973_AND_LATER



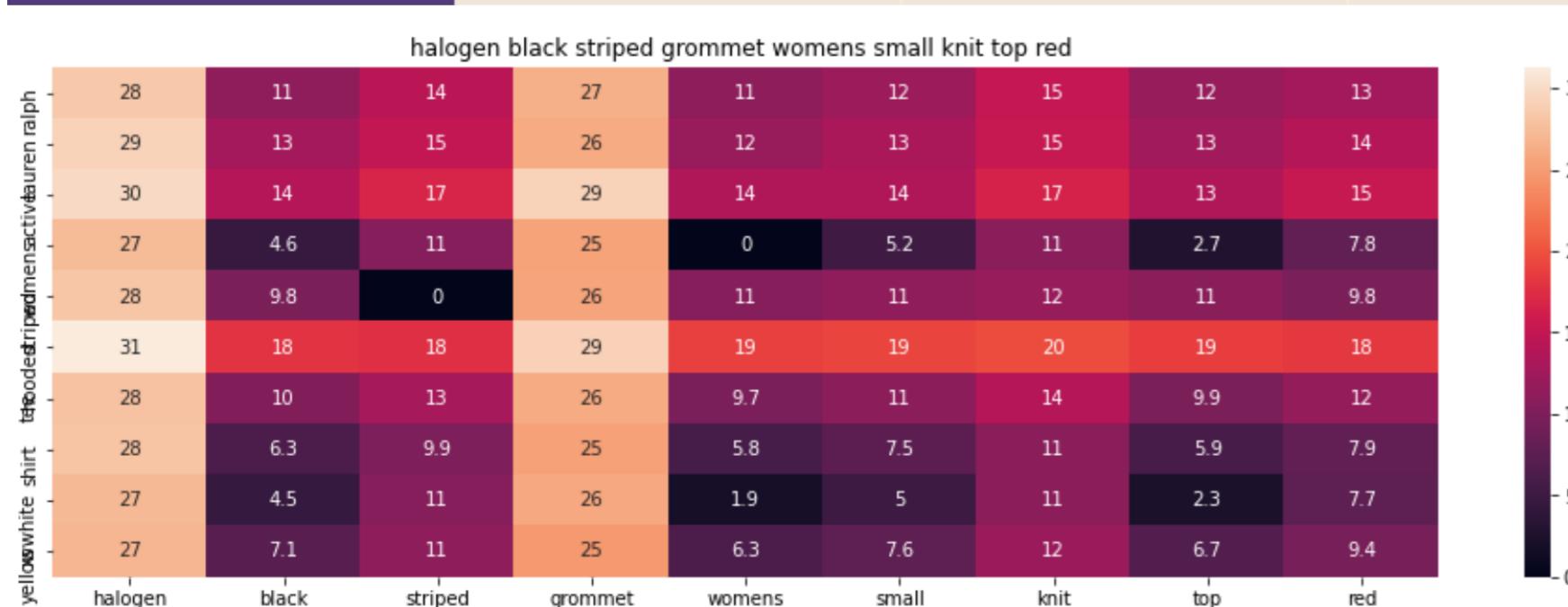
ASIN : B017X00EDW  
Brand : HUBA Design  
euclidean distance from input : 19.95

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B06XKZL1QG	Material-Girl	Black	SHIRT



ASIN : B06XKZL1QG  
Brand : Material Girl  
euclidean distance from input : 20.068

Asin	Brand	Color	Product type
B01JME4H6W	Ralph-Lauren-Active	Yellow/White	SHIRT
B0711YCRKG	Halogen	Red	SHIRT



ASIN : B0711YCRKG  
Brand : Halogen  
euclidean distance from input : 20.11

Observations :

- We have given more weight to Text feature(10) and Image(15), we can see that our model is able to predict shirts with long sleeves and plain colors well.

