Data Preprocessing and Data Wrangling

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from collections import Counter
import math
import re
import os
import seaborn as sns
from PIL import Image
import requests
from io import BytesIO
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
import nltk
import pyprind
```

In [2]:
```
data=pd.read_pickle('pickels/180k_apparel_data')
```

In [3]:
```
data.describe()
```

Out[3]:

	asin	brand	color	product_type_name	medium_image_url	title	formatted_price
count	183138	182987	64956	183138	183138	183138	28395
unique	183138	10577	7380	72	170782	175985	3135
top	B01D5VRTJ0	Zago	Black	SHIRT	https://images-na.ssl-images-amazon.com/images...	Nakoda Cotton Self Print Straight Kurti For Women	$19.99
freq	1	223	13207	167794	23	77	945

In [4]:
```
# consider products which have price information
# data['Column_name'].isnull() => gives the information
# about the dataframe row's which have null values price == None|Null
data = data.loc[~data['color'].isnull()]
print('Number of data points After eliminating Color=NULL :', data.shape[0])
data = data.loc[~data['brand'].isnull()]
print('Number of data points After eliminating Brand=NULL :', data.shape[0])
data = data.loc[~data['product_type_name'].isnull()]
print('Number of data points After eliminating Product_Type_Name=NULL :', data.shape[0])
```

```
Number of data points After eliminating Color=NULL : 64956
Number of data points After eliminating Brand=NULL : 64843
Number of data points After eliminating Product_Type_Name=NULL : 64843
```

In [5]:
```
data.to_pickle('pickels/64k_apparel_data')
```

Removing Duplicates From Title

In [6]:
```
data_sorted = data[data['title'].apply(lambda x: len(x.split())>4)]
print("After removal of products with short description:", data_sorted.shape[0])
```

```
After removal of products with short description: 63263
```

In [7]:
```
data_sorted.sort_values('title',inplace=True, ascending=False)
data_sorted.head()
```

```
a:\projects\project reference\apparel-recommendation-system\version 2.0\arsv2\lib\site-packages\ipyke
rnel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[7]:

	asin	brand	color	product_type_name	medium_image_url	title	formatted_price
27547	B073W7P8KK	Nation LTD	Blue	DRESS	https://images-na.ssl-images-amazon.com/images...	❀Nation Women Stripe Blouse Long Sleeve Shirt ...	None
31277	B01M0PWMZ8	Anglin	White	SHIRT	https://images-na.ssl-images-amazon.com/images...	✻ANGLIN✻ Women Striped Floral Long Sleeve Roun...	None
30453	B01M02GWRG	Anglin	White	SHIRT	https://images-na.ssl-images-amazon.com/images...	✻ANGLIN✻ Women Striped Floral Long Sleeve Roun...	None
32485	B01N0ADXM0	Anglin	Red	SHIRT	https://images-na.ssl-images-amazon.com/images...	✻ANGLIN✻ Women Fashion Stripe Dress Round Coll...	None
26767	B01MTQAU86	Anglin	Black	SHIRT	https://images-na.ssl-images-amazon.com/images...	✻ANGLIN✻ Women Autumn Winter Christmas Printin...	None

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

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

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

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

In [8]:
```
indices = []
for i,row in data_sorted.iterrows():
    indices.append(i)
```

In [9]:
```
%%time
import itertools
stage1_dedupe_asins = []
i = 0
j = 0
num_data_points = data_sorted.shape[0]
while i < num_data_points and j < num_data_points:

    previous_i = i

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

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

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

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

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

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

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

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

            # start searching for similar apperals corresponds 2nd string
            i = j
            break
        else:
            j += 1
    if previous_i == i:
        break
```

```
Wall time: 10.2 s
```

In [10]:
```
data = data.loc[data['asin'].isin(stage1_dedupe_asins)]
print('Number of data points : ', data.shape[0])
```

```
Number of data points :  48722
```

We removed the dupliactes which differ only at the end.

In [11]:
```
data.to_pickle('pickels/48k_apperal_data')
```

[5.2.3] Remove duplicates : Part 2

```
In the previous cell, we sorted whole data in alphabetical order of  titles.Then, we removed tit
les 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
```

Utility Functions

In [12]:
```
Total_words=0
word_index_table={}
stop_words = set(stopwords.words('english'))
print ('list of stop words:', stop_words)
indices = []
for i,row in data.iterrows():
    indices.append(i)
print(len(indices))
```

```
list of stop words: {'having', 'had', 'being', 'don', "you'd", 'what', 'same', "weren't", 'hadn', 'th
ey', 'up', 'off', 'your', 'yourself', "it's", 'y', 'when', 'other', 'isn', 'all', 'won', "shouldn't",
'each', 'yours', 'how', 'ours', 'again', 'been', "haven't", 'own', "you've", 'mustn', 'it', 'not', "h
adn't", 'a', 'shouldn', 'wasn', 'the', 'did', 'further', 'than', 'in', 'for', 'or', 'he', "wouldn't",
'couldn', "couldn't", "you're", 'so', 'ma', 'only', 'until', 'you', 'was', "mustn't", "don't", 'is',
'my', 'hers', 'are', "needn't", 'once', 'this', 'doing', 'any', 'then', 'his', 'no', "wasn't", "does
n't", 'wouldn', 'just', 'an', 'hasn', "you'll", 'their', 'above', 'both', 'of', 'some', 'now', 'mysel
f', 'll', 'needn', 'haven', "aren't", 'doesn', 'am', 'nor', 'can', "mightn't", 'weren', 'very', 'fe
w', 'if', 'be', 'such', 'does', 'between', 'do', 'about', 'yourselves', 'these', 'out', 'after', 'whi
ch', 'as', 'down', 'and', 'aren', 'whom', 'most', 'has', 'to', 'at', 'there', 'him', 'with', "has
n't", 'me', 'themselves', 'theirs', 'our', 'its', 'should', 'o', 'we', "isn't", "that'll", 'that', 'o
urselves', 'those', 'over', 'them', 'through', 'but', 'under', 'while', 'during', 'before', "should'v
e", 'into', 'ain', 'on', 'where', 'too', 'below', 'will', 'herself', 'more', 'from', 've', "won't",
'didn', 'have', 'she', 's', 'd', 'were', 'i', "shan't", 'because', 'by', 'itself', 'against', 'her',
'm', 're', 't', 'shan', 'who', 'himself', "she's", "didn't", 'here', 'why', 'mightn'}
48722
```

In [13]:
```
def nlp_preprocessing(total_text, index, column,table):
    if type(total_text) is not int:
        string = ""
        for words in total_text.split():
            # remove the special chars in review like '"#$@!%^&*()_+-~?>< etc.
            word = ("".join(e for e in words if e.isalnum()))

            # stop-word removal
            if not word in stop_words:
                string += word + " "
                # creating word-Index Dictinary
                if word in table:
                    table[word].append(index)
                else:
                    table[word]=[index]
        data[column][index] = string
    return len(string.split())
```

In [14]:
```
def preprocess_str(total_text):
    if type(total_text) is not int:
        string=[]
        for words in total_text.split():
            # Removing punctuations
            word =("".join(e for e in words if e.isalnum()))
            # Stop-Word removal
            if not word in stop_words:
                string.append(word)
    return string
```

In [15]:
```
def genrate_index_lis(stringArray):
    lis={}
    for word in stringArray:
        for ele in word_index_table[str(word)]:
            if ele in lis:
                lis[ele]+=1
            else:
                lis[ele]=1
        #lis=lis+table[str(word)]
    return [ele for ele in lis if lis[ele]>=3]
```

In [16]:
```
%%time
for index, row in data.iterrows():
    words=nlp_preprocessing(row['title'], index, 'title',word_index_table)
    Total_words=Total_words+words
print(Total_words)
```

```
470720
Wall time: 1min 7s
```

In [17]:
```
def find(A, B):
    # count
    count = {}
    # insert A in table
    for word in A:
        count[word] = count.get(word, 0) + 1
    # insert B in table
    for word in B:
        count[word] = count.get(word, 0) + 1
    # return count of Common words
    return len([word for word in count if count[word] == 2])
```

In [18]:
```
pbar=pyprind.ProgBar(data.shape[0])
```

In [19]:
```
print(len(indices))
```

```
48722
```

In [20]:
```
%%time
# This code snippet takes significant amount of time.
# O(n^2) time.
# Takes about a 2 hour to run on a decent computer.

stage2_dedupe_asins = []
while len(indices)!=0:
    i = indices.pop()
    #print("current i ",i)
    stage2_dedupe_asins.append(data['asin'].loc[i])
    # consider the first apperal's title
    a = data['title'].loc[i]
    a = preprocess_str(a)
    index_list = genrate_index_lis(a)
    # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of', 'Diamond
s', 'Women's', 'Shirt', 'X-Large']
    for j in index_list:
        #print("Current j ",j)
        b = data['title'].loc[j]
        b = preprocess_str(b)
        # store the list of words of jth string in b, ex: b = ['tokidoki', 'The', 'Queen', 'of', 'Diamo
nds', 'Women's', 'Shirt', 'X-Large']

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

        # count is used to store the number of words that are matched in both strings
        count  = find(a,b)

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

```
0% [########################      ] 100% | ETA: 00:21:21
```

```
Wall time: 1h 47min 33s
```

In [21]:
```
print(len(indices))
```

```
0
```

In [22]:
```
# from whole previous products we will consider only
# the products that are found in previous cell
data = data.loc[data['asin'].isin(stage2_dedupe_asins)]
```

In [23]:
```
print('Number of data points after stage two of dedupe: ',data.shape[0])
# from 48k apperals we reduced to 41k apperals
```

```
Number of data points after stage two of dedupe:  41012
```

In [25]:
```
data.to_pickle('pickels/41k_apperal_data')
```

Text Preprocessing

Already removed puncuations and stop-word removal

Now Lowering the text and Word Lemmantization|Steming

In [4]:
```
data = pd.read_pickle("pickels/41k_apperal_data")
```

In [5]:
```
def nlp_preprocessing2(total_text, index, column):
    if type(total_text) is not int:
        string = ""
        for words in total_text.split():
            words = words.lower()
            string =string+" "+lmtzr.lemmatize(words)
        data[column][index] = string
```

In [6]:
```
lmtzr = WordNetLemmatizer()
```

In [7]:
```
%%time
for index, row in data.iterrows():
    nlp_preprocessing2(row['title'], index, 'title')
```

```
Wall time: 37.3 s
```

In [8]:
```
data.to_pickle("pickels/41k_apperal_data2.pkl")
```

In [9]:
```
%%time
for index, row in data.iterrows():
    url = row['medium_image_url']
    #print(url)
    response = requests.get(url)
    file = open('images/41k_images/'+row['asin']+'.jpeg','wb')
    file.write(response.content)
    file.close()
    #img = Image.open(BytesIO(response.content))
    #img.save('images/41k_images/'+row['asin']+'.jpeg')
```

```
Wall time: 4h 11min 38s
```

In [ ]: