Project: Knowledge Based Recommendation System

Data Collection

DSC 630

Taniya Adhikari 15/24/2021

In [77]:
```
from bs4 import BeautifulSoup as BS
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt; plt.rcdefaults()
import matplotlib.pyplot as plt
import seaborn as sns
import requests

import warnings; warnings.simplefilter('ignore')

import re
from re import sub
import multiprocessing
from unidecode import unidecode

from gensim.models.phrases import Phrases, Phraser
from gensim.models import Word2Vec
from gensim.test.utils import get_tmpfile
from gensim.models import KeyedVectors


from time import time
from collections import defaultdict

import logging  # Setting up the loggings to monitor gensim
logging.basicConfig(format="%(levelname)s - %(asctime)s: %(message)s", datefmt= '%H:%M:
import textblob

import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from textblob import TextBlob
from sklearn.cluster import KMeans
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\bibek\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Data Collection

In [2]:
```
baseurl = "https://sokoglam.com"
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
```

In [3]:
```
k = requests.get('https://sokoglam.com/collections/skincare').text
soup=BS(k,'html.parser')
productlist = soup.find_all("div",{"class":"product-grid-item"})
```

In [4]:
```
productlinks = []
for product in productlist:
    link = product.find('a').get('href')
    productlinks.append(baseurl + link)
```

Out[4]: 1

In [6]:
```
len(productlinks)
```

Out[6]: 299

In [8]:
```
data=[]
for link in productlinks:
    f = requests.get(link,headers=headers).text
    hun=BS(f,'html.parser')
    try:
        i=hun.find("div",{"class":"review-stars"})
        product_id = i.find("div").get("data-product-id")
    except:
        product_id = None

    try:
        name=hun.find("h1",{"class":"pdp__product-title"}).text.replace('\n',"")
    except:
        name = None

    try:
        brand=hun.find("h3",{"class":"pdp__product-vendor"}).text.replace('\n',"")
    except:
        brand = None

    try:
        price=hun.find("span",{"class":"pdp-product__price--sale ProductPrice"}).text.r
    except:
        price = None

    try:
        content1=hun.find("section",{"id":"content1"})
        descp=content1.find("div",{"class":"pdp-tab-content"}).text.replace('\n',"")
    except:
        descp = None

    product = {"product_ID":product_id,"product_name":name, "product_brand":brand, "pri

    data.append(product)
```

In [9]:
```
meta_df = pd.DataFrame(data)
```

In [10]:
```
meta_df.shape
```

Out[10]: (299, 5)

Removing products that are either body or hair product, makeup products that is not typical skincare, books from the website, clothes

In [11]:
```
remove = ['Hair', 'Shampoo', 'Body', 'Foot', 'Hand']
for item in remove:
```

```
    for index, row in meta_df.iterrows():
        x = str(row['product_name'])
        if item in x:
            meta_df.drop(index, inplace=True)
        else:
            None
```

In [12]:
```
remove = ['Hair', 'Shampoo', 'Body', 'Foot', 'Hand']
for item in remove:
    for index, row in meta_df.iterrows():
        x = str(row['product_description'])
        if item in x:
            meta_df.drop(index, inplace=True)
        else:
            None
```

In [13]:
```
remove = ['Set', 'Kit', 'Routine', 'Collection', 'Duo Trial', 'Treatment Duo', 'Balanci
for item in remove:
    for index, row in meta_df.iterrows():
        x = str(row['product_name'])
        if item in x:
            meta_df.drop(index, inplace=True)
        else:
            None
```

In [14]:
```
remove = ['Liner', 'Blush', 'Primer', 'Lip Luminizer', 'Lip Lacquer', 'Eyeliner']
for item in remove:
    for index, row in meta_df.iterrows():
        x = str(row['product_name'])
        if item in x:
            meta_df.drop(index, inplace=True)
        else:
            None
```

In [15]:
```
remove = ['Cotton', 'Paper', 'Book', 'Baseball Cap', 'Sweatshirt']
for item in remove:
    for index, row in meta_df.iterrows():
        x = str(row['product_name'])
        if item in x:
            meta_df.drop(index, inplace=True)
        else:
            None
```

In [16]:
```
meta_df.shape
```

Out[16]: (242, 5)

Adding a product type to the dataframe

In [17]:
```
meta_df['product_type'] = None
```

In [18]:
```
def product_type(items, word, df):
    for item in items:
        for index, row in df.iterrows():
            if row['product_type'] == None:
                x = str(row['product_name'])
                if item in x:
                    row['product_type'] = word
```

```
                else:
                    None
```

In [19]:
```
items = ['Eye']
product_type(items, 'Eye Treatment', meta_df)
```

In [20]:
```
items = ['Lip']
product_type(items, 'Lip Treatment', meta_df)
```

In [21]:
```
add = ['Sun Cream', 'Sun Essence', 'Sunscreen', 'UV', 'SPF', "I'm Safe For Sensitive Sk
for item in add:
    for index, row in meta_df.iterrows():
        if row['product_type'] == None:
            x = str(row['product_name'])
            if item in x:
                row['product_type'] = 'Sun Protection'
            else:
                None
```

In [22]:
```
items = ['Sheet Mask']
product_type(items, 'Sheet Mask', meta_df)
```

In [23]:
```
items = ['Mask']
product_type(items, 'Mask', meta_df)
```

In [24]:
```
items = ['Serum', 'Ampoule']
product_type(items, 'Serum', meta_df)
```

In [25]:
```
items = ['Essence', 'Mist']
product_type(items, 'Essence', meta_df)
```

In [26]:
```
items = ['Cleansing', 'Cleanser', 'Foam']
product_type(items, 'Cleanser', meta_df)
```

In [27]:
```
items = ['Toner', 'Toning', 'Water']
product_type(items, 'Toner', meta_df)
```

In [28]:
```
items = ['Lotion', 'Cream', 'Moisturizer', 'Emulsion']
product_type(items, 'Moisturizer', meta_df)
```

In [29]:
```
items = ['Peel', 'Peeling', 'Scrub', 'Exfoliat', 'Pad']
product_type(items, 'Exfoliator', meta_df)
```

In [30]:
```
for index, row in meta_df.iterrows():
    if row['product_type'] == None:
        x = str(row['product_name'])
        row['product_type'] = 'Other/Spot Treatments'
```

In [31]:
```
meta_df.to_csv('productlist.csv')
```

In [34]:
```
session = requests.Session()
#productToReviews = [] #empty dictionary
productReviews = []
for index, row in meta_df.iterrows():
    requestUrl = 'https://staticw2.yotpo.com/batch/app_key/kILjLgKH3AFJKWu0W8HoD8nuvs72
```

```
    recordExists = True
    i = 1
    while(recordExists):
        methodBody = '[{"method":"reviews","params":{"pid":' + row.product_ID + ',"orde
        #print(methodBody)
        x = session.post(url, data = {'methods': methodBody})
        if('be the first to write a review' in x.text):
            #print("no record")
            recordExists = False
        else:
            y = json.loads(x.text)
            soup=BS(y[0]["result"],'html.parser')
            reviewList = soup.find_all("div",{"class":"content-review"})
            for review in reviewList:
                s = BS(str(review),'html.parser')
                r = s.find("div").text.strip()
                if(len(r)):
                    productReviews.append({"product_id": row.product_ID, "review": r})
            i += 1
reviewDataFrame = pd.DataFrame.from_records(productReviews)
```

Out[34]: 1

In [436…

```
reviewDataFrame.to_csv('productReviews.csv')
```