Project: Knowledge Based Recommendation System

Data Preparation

DSC 630

Taniya Adhikari 15/24/2021

In [1]:
```
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!
```

Functions

1. Remove WhiteSpace

In [2]:
```
def no_whitespace(string):
    return string.strip()
```

In [3]:
```
# stripping white space from all column type as object
def clean_string(df):

    for col in df.columns:
        if df[col].dtype == np.object:
            new = '{}_new'.format(col)
            df[new] = df[col].apply(no_whitespace)
            df[col] = df[new]
            df = df.drop(columns=[new])
    return df
```

1. Remove Punctuation and Convert string to Lower Case

In [4]:
```
import string


def text_w_punc(text):
    pattern = r'[^A-Za-z ]'
    regex = re.compile(pattern)
    text = regex.sub(' ', text)
    return text


def pre_processing1(df, col):
    # converting all text to lowercase
    df[col] = df[col].str.lower()

    # removing punctuation using string.punctuations and join()
    df[col] = df[col].apply(lambda x: "".join([i for i in x if i not in string.punctuat
    df[col] = df[col].apply(lambda x: text_w_punc(x))

    return df
```

1. Remove Stopwords

In [5]:
```
stop = stopwords.words('english')

def remove_stopwords(df, col, stop):
# remove stop words for bag of word model and td-idf

    df['stopwords'] = df[col].apply(lambda x: len([i for i in x.split() if i in stop]))
    df['clean']= df[col].apply(lambda x: " ".join(i for i in x.split() if i not in stop
    df['stopwords'] = df[col].apply(lambda x: len([i for i in x.split() if i in stop]))
    del df['stopwords']
    return df
```

1. Lemmatization

In [6]:
```
from nltk import pos_tag
from nltk.corpus import wordnet
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer


def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
```

```
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

def listToString(s):
    # initialize an empty string
    str1 = " "

    # return string
    return (str1.join(s))


def lemmatized_text(df, col):

    text_list = df[col].tolist()

    tagged_texts = []

    # tag each word
    for text in text_list:
        text_tag = pos_tag(word_tokenize(text))
        tagged_texts.append(text_tag)

    df['pos_tags'] = tagged_texts
    df['wordnet_pos'] = df['pos_tags'].apply(lambda x: [(word, get_wordnet_pos(pos_tag)

    wnl = WordNetLemmatizer()
    df['lemmatized'] = df['wordnet_pos'].apply(lambda x: [wnl.lemmatize(word, tag) for
    df[col]= df["lemmatized"].apply(lambda x: " ".join(i for i in x))
    return df
```

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

1. Convert currency to float

In [7]:
```
def clean_currency(x):

    #If the value is a string, then remove currency symbol and delimiters
    #otherwise, the value is numeric and can be converted

    if isinstance(x, str):
        return(x.replace('$', '').replace(',', ''))
    return(x)
```

Data Preparation

Data Frame 1 Product List

In [8]:
```
productList_df = pd.read_csv("productlist.csv", index_col=0)
```

In [9]:
```
productList_df.head(3)
```

Out[9]:

	product_ID	product_name	product_brand	price	product_description	product_type
0	6562638659653	VITALIFT-A	Dr. Different	$42	This night-time skin treatment is ideal for th...	Other/Spot Treatments
1	6562639675461	VITALIFT-A Forte	Dr. Different	$52	Those that need an extra boost to smooth out f...	Other/Spot Treatments
2	6562640429125	VITALIFT-A Eye & Neck	Dr. Different	$40	For those looking to target fine lines and wri...	Eye Treatment

In [10]:
```
productList_df = clean_string(productList_df)
productList_df = pre_processing1(productList_df, "product_description")
productList_df = remove_stopwords(productList_df, "product_description", stop)
productList_df = lemmatized_text(productList_df, "clean")
productList_df = productList_df.rename(columns={"clean": "description_clean"})
productList_df.head(3)
```

Out[10]:

	product_ID	product_name	product_brand	price	product_description	product_type	description_c
0	6562638659653	VITALIFT-A	Dr. Different	$42	this nighttime skin treatment is ideal for tho...	Other/Spot Treatments	nighttime treatment look improve
1	6562639675461	VITALIFT-A Forte	Dr. Different	$52	those that need an extra boost to smooth out f...	Other/Spot Treatments	need extra b smooth fine wrinkle
2	6562640429125	VITALIFT-A Eye & Neck	Dr. Different	$40	for those looking to target fine lines and wri...	Eye Treatment	look target line wr specifically

In [11]:
```
productList_df['price'] = productList_df['price'].apply(clean_currency).astype('float')
productList_df = productList_df.rename(columns={"product_ID": "product_id"})
```

In [12]:
```
productList_df.head(3)
```

Out[12]:

	product_id	product_name	product_brand	price	product_description	product_type	description_c
0	6562638659653	VITALIFT-A	Dr. Different	42.0	this nighttime skin treatment is ideal for tho...	Other/Spot Treatments	nighttime treatment look improve
1	6562639675461	VITALIFT-A Forte	Dr. Different	52.0	those that need an extra boost to smooth out f...	Other/Spot Treatments	need extra b smooth fine wrinkle

	product_id	product_name	product_brand	price	product_description	product_type	description_c
2	6562640429125	VITALIFT-A Eye & Neck	Dr. Different	40.0	for those looking to target fine lines and wri...	Eye Treatment	look target line wr specifically

In [13]:
```
X = productList_df[['product_id','product_name','product_brand', 'price','product_type'
```

In [14]:
```
X.to_csv('prod_descR.csv')
```

Data Frame 2 Product Reviews

In [15]:
```
productReviews_df = pd.read_csv("productReviews.csv", index_col=0)
```

In [16]:
```
productReviews_df[['product_id','review']].head(3)
```

Out[16]:

	product_id	review
0	4669755719749	This makes my skin smooth and soft and is ligh...
1	4669755719749	Love the silky texture. It's very lightweight ...
2	4669755719749	I've been trying to find a moisturizer that wo...

In [17]:
```
productReviews_df.iloc[2,1]
```

Out[17]:
```
'I've been trying to find a moisturizer that would not dry out my skin for months. Espec
ially while wearing a mask. This product does much better than lots of others I have tri
ed. Thank you'
```

In [18]:
```
productReviews_df = clean_string(productReviews_df)
productReviews_df.head(2)
```

Out[18]:

	product_id	review
0	4669755719749	This makes my skin smooth and soft and is ligh...
1	4669755719749	Love the silky texture. It's very lightweight ...

In [19]:
```
productReviews_df = productReviews_df.rename(columns={"review": "review_sentiment"})
```

In [20]:
```
productReviews_df = pre_processing1(productReviews_df, 'review_sentiment')
```

Deleting any reviews that includes hi or sorry words, because those are replies from the seller

In [21]:
```
remove = ['Hi ', 'hi ', 'sorry', 'Sorry']
for item in remove:
    for index, row in productReviews_df.iterrows():
        x = str(row['review_sentiment'])
        if item in x:
            productReviews_df.drop(index, inplace=True)
        else:
            None
```

In [22]:
```
productReviews_df = remove_stopwords(productReviews_df, "review_sentiment", stop)
```

In [23]:
```
productReviews_df = lemmatized_text(productReviews_df, "clean")
productReviews_df = productReviews_df.rename(columns={"clean": "review_similarity"})
```

In [25]:
```
productReviews_df= productReviews_df.reset_index(drop = True)
```

In [26]:
```
productReviews_df['review_similarity'].isnull().values.any()
```

Out[26]:
```
False
```

In [27]:
```
productReviews_df.shape
```

Out[27]:
```
(14562, 6)
```

In [28]:
```
productReviews_df.head(3)
```

Out[28]:

	product_id	review_sentiment	review_similarity	pos_tags	wordnet_pos	lemmatized
0	4669755719749	this makes my skin smooth and soft and is ligh...	make skin smooth soft lightweight absorbs quickly	[(makes, VBZ), (skin, JJ), (smooth, JJ), (soft...	[(makes, v), (skin, a), (smooth, a), (soft, a)...	[make, skin, smooth, soft, lightweight, absorb...
1	4669755719749	love the silky texture its very lightweight bu...	love silky texture lightweight hydrate leaf sk...	[(love, VB), (silky, JJ), (texture, NN), (ligh...	[(love, v), (silky, a), (texture, n), (lightwe...	[love, silky, texture, lightweight, hydrate, l...
2	4669755719749	i ve been trying to find a moisturizer that wo...	try find moisturizer would dry skin month espe...	[(trying, VBG), (find, NN), (moisturizer, NN),...	[(trying, v), (find, n), (moisturizer, n), (wo...	[try, find, moisturizer, would, dry, skin, mon...

In [30]:
```
productReviews_df.to_csv('clean_reviews.csv')
```