

# Project: Knowledge Based Recommendation System

## Exploratory Data Analysis

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.rcParams()
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:%S')
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!
```

### Dataset 1: Product List and Description

```
In [2]: productList_df = pd.read_csv("prod_descR.csv", index_col=0)
```

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

```
Out[3]: product_id product_name product_brand price product_type product_description description_c
```

	product_id	product_name	product_brand	price	product_type	product_description	description_c
0	6562638659653	VITALIFT-A	Dr. Different	42.0	Other/Spot Treatments	this nighttime skin treatment is ideal for those that need an extra boost to smooth out fine lines and wrinkles.	nighttime treatment look improve
1	6562639675461	VITALIFT-A Forte	Dr. Different	52.0	Other/Spot Treatments	those that need an extra boost to smooth out fine lines and wrinkles.	need extra boost to smooth fine wrinkles
2	6562640429125	VITALIFT-A Eye & Neck	Dr. Different	40.0	Eye Treatment	for those looking to target fine lines and wrinkles specifically.	look target line wrinkles specifically

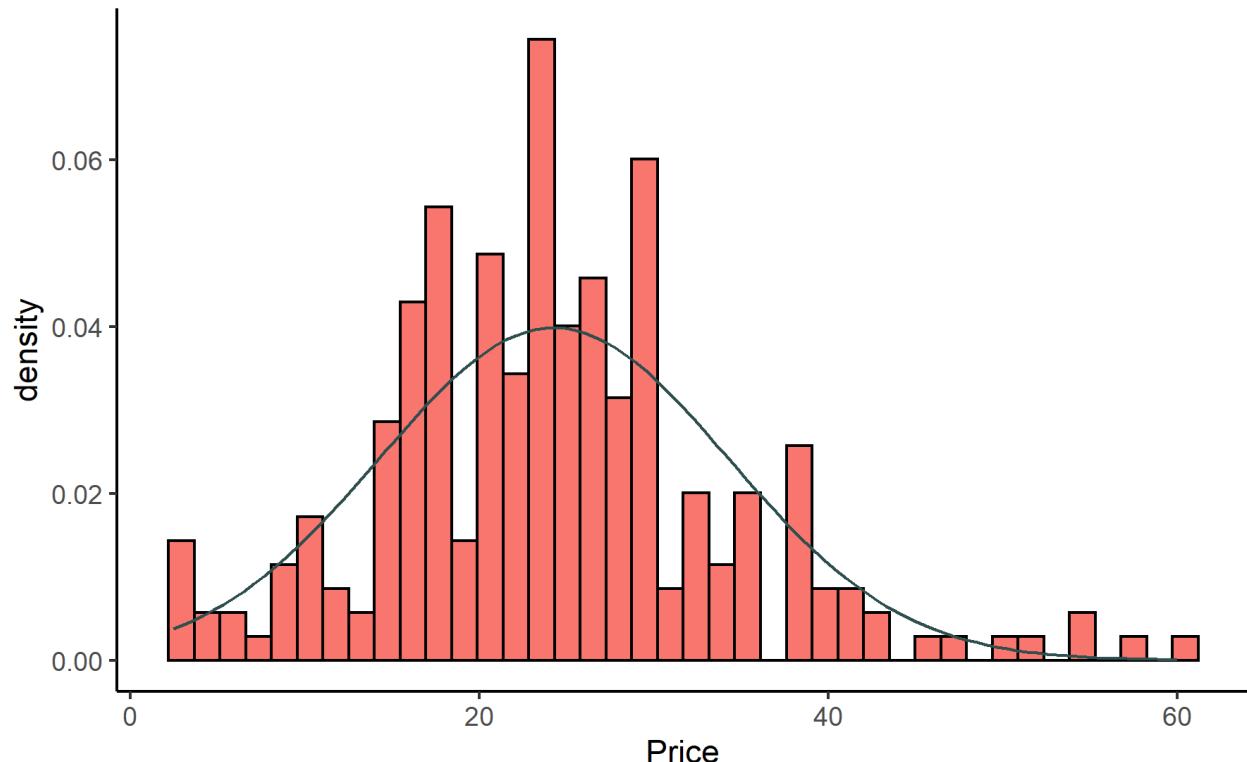


In [4]:

```
from IPython.display import Image
Image(filename='histogram_price.png')
```

Out[4]:

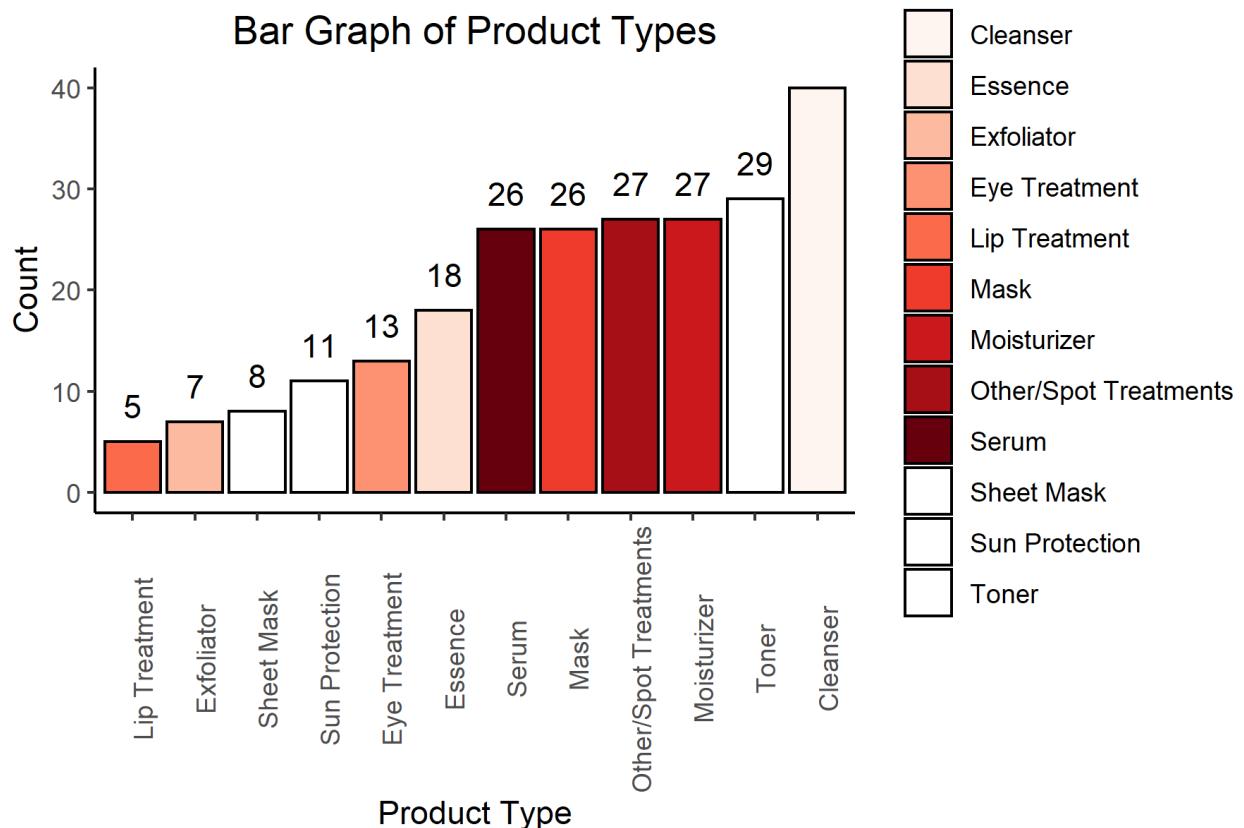
### Product Price Distribution



In [5]:

```
from IPython.display import Image
Image(filename='Bar_productType.png')
```

Out[5]:



## Dataset 2: Product Reviews

```
In [6]: productReviews_df = pd.read_csv('clean_reviews.csv', index_col=0).dropna()
productReviews_df.head(3)
```

INFO - 05:29:11: NumExpr defaulting to 8 threads.

	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'), ('lightweight', 'JJ')]	[('makes', 'v'), ('skin', 'a'), ('smooth', 'a'), ('lightweight', 'a')] ...	['make', 'skin', 'smooth', 'soft', 'lightweigh...']
1	4669755719749	love the silky texture its very lightweight bu...	love silky texture lightweight hydrate leaf sk...	[('love', 'VB'), ('silky', 'JJ'), ('texture', 'NN')]	[('love', 'v'), ('silky', 'a'), ('texture', 'n')] ...	['love', 'silky', 'texture', 'lightweight', 'h...']
2	4669755719749	i ve been trying to find a moisturizer that wo...	try find moisturizer would dry skin month espe...	[('try', 'VBG'), ('find', 'VBD'), ('moisturi...', 'NN')]	[('try', 'v'), ('find', 'n'), ('moisturizer...', 'n')] ...	['try', 'find', 'moisturizer', 'would', 'dry', ...']

```
In [7]: productReviews_df.shape
```

Out[7]: (14548, 6)

```
In [8]: productReviews_df = productReviews_df.reset_index(drop = True)
```

## Fake review detection

Most common words

```
In [9]: from collections import Counter

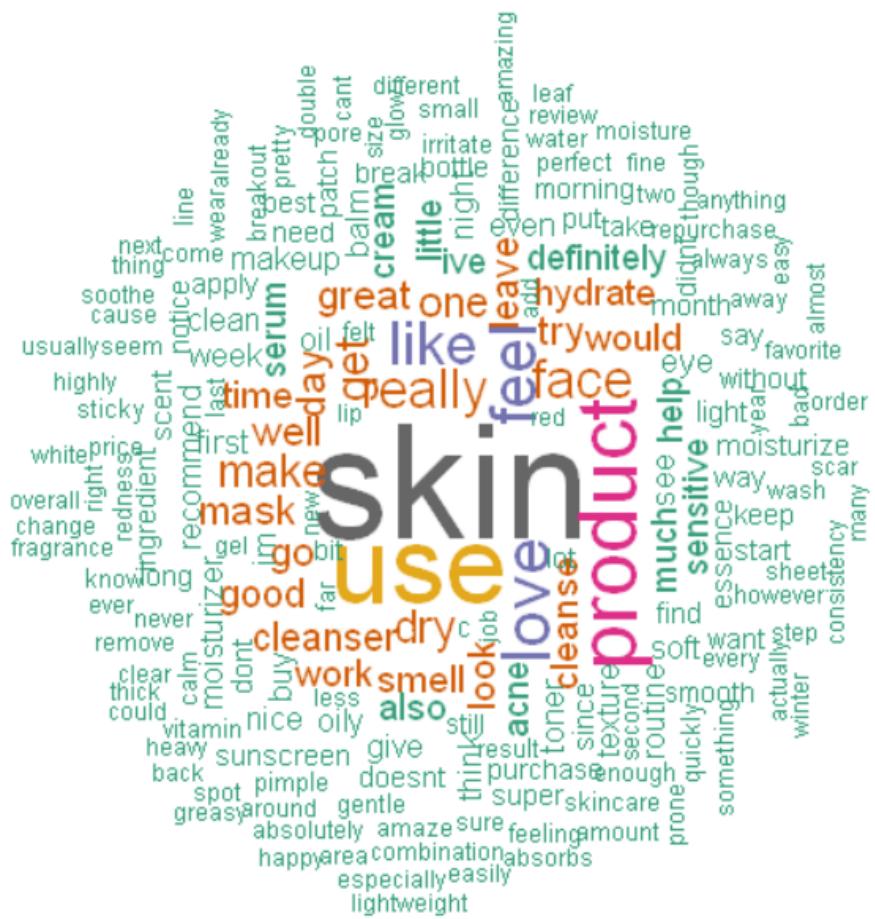
aggregate_counter = Counter()
for row_index, row in productReviews_df.iterrows():
    c = Counter(row['review_similarity'].split())
    aggregate_counter += c

common_words = [word[0] for word in aggregate_counter.most_common()]
common_words_counts = [word[1] for word in aggregate_counter.most_common()]
```

```
In [10]: common_word = pd.DataFrame(list(zip(common_words, common_words_counts)),columns =['Word', 'Count'])  
common_word.to_csv('common_word.csv')
```

```
In [11]: Image(filename='Most_Frequent_word.png')
```

Out[11]:



No word Embeddings because we are not trying to find the contextual meaning

```
In [12]: from sklearn.feature_extraction.text import TfidfVectorizer  
# Create array with each text  
data_arr = np.array(productReviews_df['review_similarity'].values.astype("U"))  
  
# Create the tf-idf feature matrix  
tfidf = TfidfVectorizer()  
feature = tfidf.fit_transform(data_arr)
```

```
# Show feature matrix
feature
```

Out[12]: <14548x10744 sparse matrix of type '<class 'numpy.float64'>'  
with 322688 stored elements in Compressed Sparse Row format>

In [13]: #create dataframe  
X=pd.DataFrame(feature.toarray())  
tfidf2 = X.values.tolist()

In [14]: from sklearn.metrics.pairwise import cosine\_similarity  
cosine\_sim = cosine\_similarity(feature)  
Y=pd.DataFrame(cosine\_sim)

## Checking if there is any similar reviews

In [15]: for col in Y.columns:  
 if any(Y[col] > 80):  
 print("True")

Fuzzy Matching for Feature Extraction from reviews and User preference

In [16]: from sklearn.feature\_extraction.text import CountVectorizer  
  
features = productReviews\_df["review\_similarity"].tolist()  
CountVec = CountVectorizer()  
#transform  
Count\_data = CountVec.fit\_transform(features)  
Count\_data

Out[16]: <14548x10744 sparse matrix of type '<class 'numpy.int64'>'  
with 322688 stored elements in Compressed Sparse Row format>

In [17]: #create dataframe  
cv\_dataframe=pd.DataFrame(Count\_data.toarray(),columns=CountVec.get\_feature\_names())

In [18]: words = pd.DataFrame(cv\_dataframe.idxmax(1),columns=['word'])

In [19]: from fuzzywuzzy import process, fuzz

In [20]: def token\_fuzzy(df, percent, scorer):  
  
 # unique words lists  
 unique\_word = df['word'].unique().tolist()  
 sorted(unique\_word)  
  
 '''Token Ratio'''  
 #Create tuples of words, matched words, and the score of token sort  
 score = [(x,) + i  
 for x in unique\_word  
 for i in process.extract(x, unique\_word, scorer=scorer)]  
  
 #Create a dataframe from the tuples  
 similarity = pd.DataFrame(score, columns=['word','match','score'])  
  
 # because this will have repetitive pairs, we will remove duplicated pairs from this  
 similarity['sorted\_word'] = np.minimum(similarity['word'], similarity['match'])

```
# Token high score
high_score = token_high(similarity, percent)

return similarity, high_score
```

In [24]:

```
def token_high(df, percent):

    # create a subset dataframe with higher percent match values
    high_score = df[(df['score'] >= percent) &
                    (df['word'] != df['match']) &
                    (df['sorted_word'] != df['match'])]

    # Drop the representative value column
    high_score = high_score.drop('sorted_word', axis=1).copy()

    return high_score
```

In [26]:

```
similarity, high_score = token_fuzzy(words, 60, fuzz.token_sort_ratio)
high_score.groupby(['word', 'score']).agg({'match': ' '.join}).sort_values(['score'], ascending=False)

# create dictionary
match_dict = dict(zip(high_score.match, high_score.word))
match_dict
```

Out[26]:

```
{'absorbtion': 'absorbs',
 'balms': 'balm',
 'buyyy': 'buy',
 'buying': 'burning',
 'could': 'cold',
 'cool': 'control',
 'couldnt': 'account',
 'would': 'couldnt',
 'away': 'aware',
 'way': 'say',
 'atleast': 'always',
 'burn': 'broken',
 'hand': 'andor',
 'husband': 'hand',
 'never': 'need',
 'everyday': 'every',
 'boost': 'bonus',
 'least': 'eas',
 'chest': 'best',
 'aloe': 'allinone',
 'blue': 'able',
 'blend': 'beyond',
 'absolute': 'able',
 'amazed': 'amaze',
 'make': 'amazed',
 'amazing': 'amaaaaazing',
 'name': 'inflame',
 'lovely': 'likely',
 'loveee': 'loooovvvveee',
 'moisturizers': 'moisturiser',
 'moisturizes': 'moisture',
 'creamy': 'crazy',
 'crem': 'creamy',
 'creme': 'crem',
 'dreamy': 'dewy',
 'efficiently': 'efficient',
 'gently': 'general',
```

```
'create': 'accurate',
'fr': 'far',
'favor': 'fav',
'flare': 'far',
'jar': 'apr',
'favorites': 'faves',
'since': 'include',
'fence': 'confidence',
'essential': 'especially',
'see': 'asleep',
'face': 'ache',
'maskcne': 'acne',
'live': 'deliver',
'allergy': 'aftergym',
'eric': 'allergic',
'alli': 'allergic',
'happy': 'chap',
'applying': 'apply',
'asiatica': 'aesthetician',
'changer': 'angry',
'charge': 'careful',
'feels': 'else',
'felt': 'el',
'free': 'fr',
'little': 'climate',
'ive': 'five',
'breakout': 'bergamot',
'breaking': 'backing',
'breakouts': 'breakout',
'basically': 'basic',
'time': 'sometimes',
'item': 'bite',
'carolina': 'aroma',
'bathroom': 'aroma',
'thing': 'something',
'noticing': 'antiaging',
'something': 'nothing',
'yet': 'etc',
'theyre': 'others',
'liked': 'lid',
'likely': 'briefly',
'line': 'baseline',
'try': 'tr',
'goto': 'go',
'month': 'dont',
'around': 'amount',
'nothing': 'annoying',
'put': 'cut',
'calms': 'balms',
'camu': 'calm',
'great': 'gear',
'black': 'backup',
'backup': 'back',
'pack': 'back',
'buck': 'bc',
'different': 'adherent',
'fermented': 'different',
'experience': 'expensive',
'absorbs': 'absorbent',
'absorbed': 'absorb',
'absorbent': 'absorbed',
'absored': 'absorbent',
'day': 'daily',
'easily': 'certainly',
'oily': 'acneoily',
```

```
'ever': 'elmer',
'evening': 'depending',
'evident': 'emollient',
'every': 'dewey',
'addicted': 'added',
'fade': 'face',
'dead': 'bead',
'bastante': 'agitates',
'chap': 'cap',
'cheap': 'chap',
'absolutley': 'absolute',
'completely': 'complaint',
'fabulously': 'fabulous',
'scar': 'carry',
'case': 'becausee',
'dont': 'do',
'dirt': 'admit',
'isnt': 'instantly',
'purchasing': 'purchase',
'prone': 'broke',
'alot': 'adult',
'amount': 'akon',
'others': 'bother',
'bother': 'another',
'water': 'aftergym',
'full': 'fall',
'careful': 'awful',
'fact': 'act',
'fave': 'fade',
'effectively': 'effect',
'efficient': 'effective',
'please': 'plane',
'love': 'alove',
'lighten': 'gotten',
'right': 'alright',
'night': 'high',
'shave': 'havent',
'heavenly': 'eventually',
'sure': 'cure',
'effective': 'affect',
'perfect': 'effect',
'enjoyed': 'enjoy',
'caste': 'case',
'last': 'asthma',
'cost': 'chest',
'bead': 'beach',
'brand': 'bad',
'badly': 'baby',
'dozen': 'doesnt',
'dot': 'do',
'burning': 'bring',
'applicator': 'applicable',
'expectation': 'exceptional',
'anyone': 'allinone',
'approve': 'acneprone',
'beautifully': 'beautefulthanks',
'beauty': 'beat',
'good': 'go',
'dry': 'dr',
'say': 'sad',
'cica': 'chemical',
/heal': 'deal',
'physical': 'chemical',
'alittle': 'agitates',
'irritate': 'irradiate',
```

```
'form': 'effort',
'added': 'add',
'bad': 'amd',
'amd': 'amazed',
'pad': 'amd',
'bring': 'breaking',
'bit': 'birth',
'bright': 'alright',
'makeup': 'keep',
'market': 'mark',
'maskne': 'maskcne',
'take': 'cakey',
'house': 'hope',
'though': 'enough',
'improvement': 'impressed',
'remove': 'improve',
'well': 'acnewell',
'blind': 'blend',
'wish': 'wash',
'friend': 'fermented',
'wind': 'win',
'fine': 'definelty',
'found': 'around',
'favorite': 'favor',
'melt': 'mell',
'flat': 'fast',
'fit': 'benefit',
'mist': 'isnt',
'list': 'isnt',
'awe': 'acwell',
'care': 'adore',
'wear': 'gear',
'sizzle': 'size',
'curious': 'citrus',
'leaf': 'la',
'lady': 'glad',
'become': 'awesome',
'end': 'beyond',
'ready': 'read',
'area': 'agree',
'read': 'dead',
'clesances': 'cleanse',
'clearer': 'carry',
'zone': 'none',
'tone': 'money',
'winter': 'intend',
'ok': 'chok',
'cant': 'can',
'anytime': 'anymore',
'anything': 'annoying',
'quantity': 'anti',
'rinse': 'nose',
'intend': 'ginseng',
'glowy': 'gloss',
'glowing': 'blowing',
'go': 'ago',
'fragranced': 'barefaced',
'fragrant': 'fragranced',
'fragrantless': 'fragrant',
'extract': 'etc',
'texture': 'future',
'next': 'extra',
'balmy': 'baby',
'bay': 'balmy',
'barely': 'badly',
```

```
'describe': 'decide',
'excite': 'excess',
'act': 'ac',
'acne': 'ache',
'back': 'ac',
'mask': 'mark',
'eas': 'ask',
'hanskin': 'asian',
'feel': 'el',
'week': 'cheek',
'write': 'rise',
>wonder': 'powder',
'fun': 'fan',
'refund': 'around',
'nightly': 'night',
'even': 'en',
'must': 'mist',
'freshen': 'free',
'boot': 'boost',
'boston': 'benton',
'condition': 'additional',
'combinationdehydrated': 'combination',
'foundation': 'conjunction',
'contain': 'certainly',
'consistent': 'congested',
'content': 'concentrate',
'fragrance': 'appearance',
'disappear': 'appear',
>bite': 'birth',
'hit': 'fit',
'fail': 'facial',
'daily': 'acneoily',
'avoid': 'aid',
'grail': 'general',
'powder': 'order',
'pore': 'korean',
'cream': 'areasmof',
'enough': 'bought',
'cleanser': 'changer',
'hyper': 'hope',
'calm': 'call',
'palm': 'balm',
'cleanse': 'cleaning',
'cleaner': 'changer',
'clear': 'clarify',
'cast': 'asthma',
'encanto': 'cant',
'timy': 'time',
'absolutely': 'absolute',
'wrk': 'work',
'cleansingbefore': 'cleaning',
'notice': 'antiacne',
'price': 'eric',
'confuse': 'bonus',
'fabulous': 'bonus',
'compliment': 'complain',
'correctly': 'comfortably',
'couple': 'collage',
'oit': 'oil',
'smooth': 'month',
'news': 'new',
'one': 'none',
'review': 'remove',
'recently': 'eventually',
'get': 'budget',
```

'amazingly': 'amaaaaazing',  
'damaging': 'amaaaaazing',  
'calming': 'applying',  
'rise': 'advertise',  
'seem': 'eczema',  
'sheen': 'seem',  
'sheet': 'seem',  
'use': 'suppose',  
'give': 'five',  
'cover': 'concern',  
'bumpy': 'bumpwill',  
'bumps': 'bum',  
'pump': 'bumps',  
'afterwards': 'afterward',  
'behind': 'begin',  
'funky': 'chunky',  
'sun': 'son',  
'agree': 'aggressive',  
'change': 'can',  
'ago': 'age',  
'sunscreen': 'screen',  
'triple': 'pimple',  
'waste': 'caste',  
'patch': 'batch',  
'additional': 'addition',  
'transition': 'additional',  
'appear': 'apart',  
'apply': 'applied',  
'suppose': 'super',  
'emulsion': 'emulsifies',  
'like': 'le',  
'perfectly': 'correctly',  
'grainy': 'certainly',  
'scent': 'current',  
'excelent': 'decent',  
'wowed': 'wow',  
'gain': 'asian',  
'akin': 'agin',  
'begin': 'agin',  
'improve': 'impressed',  
'cough': 'clog',  
'spray': 'say',  
'routine': 'outside',  
'stick': 'asiatica',  
'sticky': 'stick',  
'sticker': 'stick',  
'cause': 'case',  
'hydrated': 'dehydrate',  
'hydrating': 'dehydration',  
'spot': 'point',  
'soothe': 'smooth',  
'stop': 'soap',  
'strip': 'drip',  
'bag': 'ago',  
'const': 'congested',  
'cotton': 'bottom',  
'year': 'gear',  
'travel': 'rave',  
'wonderful': 'wonder',  
'probably': 'baby',  
'drop': 'drip',  
'really': 'ready',  
'cap': 'apr',  
'appeal': 'apparu',  
'believer': 'believe',

```
'deliver': 'believer',
'leave': 'lavender',
'aesthetician': 'aesthetic',
'arsenal': 'area',
'white': 'item',
'vet': 'fit',
'hot': 'dot',
'briefly': 'barely',
'early': 'barely',
'along': 'akon',
'above': 'aloe',
'althea': 'alternate',
'blackhead': 'black',
'already': 'afraid',
'sample': 'cumple',
'accurate': 'accumulate',
'accutane': 'accurate',
'ampule': 'ampoule',
'cumple': 'couple',
'brightens': 'brightening',
'brighter': 'bitter',
'brightening': 'brighten',
'airy': 'air',
'far': 'fabric',
'fast': 'fact',
'packaging': 'backing',
'however': 'ever',
'whenever': 'never',
'come': 'combine',
'recommend': 'reccomrnded',
'bee': 'become',
'comedo': 'combo',
'terrible': 'describe',
'actually': 'accidentally',
'actual': 'act',
'bummer': 'blur',
'smell': 'mell',
'compare': 'come',
'quite': 'adequate',
'long': 'blowing',
'also': 'aloe',
'easy': 'eas',
'screen': 'creep',
'exfoliant': 'emollient',
'cute': 'cure',
'top': 'hop',
'trop': 'tr',
'feek': 'cheek',
'comfortable': 'comfort',
'floral': 'affordable',
'brightness': 'brightens',
'boy': 'body',
'hype': 'hope',
'keep': 'deep',
'dewey': 'deep',
'thick': 'stick',
'shiny': 'chin',
'exellent': 'excelent',
'experiment': 'expensive',
'angry': 'airy',
'carry': 'airy',
'definitely': 'definely',
'didnt': 'definite',
'whole': 'awhile',
'look': 'hook',
```

```
'hook': 'chok',
'brighten': 'bright',
'concern': 'concentrate',
'concentrate': 'concealer',
'container': 'consider',
'cooler': 'alcohol',
'greasyshiny': 'grainy',
'none': 'non',
'beat': 'bead',
'needle': 'need',
'follow': 'allow',
'side': 'downside',
'size': 'since',
'base': 'abrasive',
'heals': 'eas',
'chunky': 'chunk',
'healthy': 'chalky',
'complain': 'carolina',
'especially': 'basically',
'aggressive': 'abrasive',
'hydration': 'dehydration',
'facial': 'aczima',
'cystic': 'cyst',
'circle': 'cica',
'dang': 'bang',
'many': 'annoy',
'another': 'ahewr',
'anymore': 'andor',
'alive': 'active',
'bottle': 'boot',
'super': 'spray',
'collection': 'collagen',
'coverage': 'average',
'pimple': 'cumple',
'appearance': 'appear',
'buy': 'bum',
'continue': 'confidence',
'elsenot': 'else',
'le': 'else',
'arrive': 'abrasive',
'refresh': 'freshen',
'freshly': 'freshen',
'lightwieght': 'lightweight',
'lightyet': 'lighten',
'pretty': 'cruelty',
'collagen': 'collage',
'firm': 'film',
'consider': 'confidence',
'future': 'figure',
'finger': 'fine',
'picture': 'future',
'hydrate': 'dehydrate',
'repurchase': 'purchase',
'climate': 'activate',
'result': 'amazingresult',
'feeling': 'eyelid',
'everything': 'agedefying',
'glow': 'gloss',
'active': 'activate',
'cheek': 'achieve',
'delicious': 'curious',
'harsh': 'brush',
'dropper': 'drop',
'firming': 'firm',
'five': 'fine',
```

```
'office': 'notice',
'bitter': 'barrier',
'buttery': 'bitter',
'either': 'bother',
'irradiate': 'immediately',
'banila': 'animal',
'bang': 'bag',
'big': 'bag',
'applies': 'applicable',
'aid': 'afraid',
'aside': 'aid',
'greasy': 'grainy',
'heavy': 'heavenly',
'loooovvveee': 'looooove',
'cook': 'chok',
'bombee': 'bomb',
'combo': 'combine',
'hard': 'hand',
'body': 'anybody',
'def': 'de',
'due': 'dilute',
'deep': 'de',
'dewy': 'dewey',
'convert': 'convenient',
'amazingresult': 'amazingly',
'work': 'coworker',
'okay': 'ok',
'faves': 'facemask',
'downside': 'consider',
'pha': 'pad',
'half': 'ha',
'skin': 'asian',
'sink': 'shiny',
'drip': 'drink',
'hope': 'hop',
'havent': 'arent',
'loveeeee': 'chileeeee',
'applied': 'applicable',
'application': 'action',
'fav': 'fan',
'less': 'excess',
'think': 'chunk',
'chunk': 'chin',
'aroma': 'arm',
'charm': 'charge',
'probiotic': 'apricot',
'aspect': 'affect',
'disappointed': 'disappoint',
'formula': 'form',
'overall': 'floral',
'foam': 'floral',
'prescription': 'exceptional',
'pouch': 'much',
'mucin': 'emulsion',
'quench': 'much',
'include': 'dilute',
'freckle': 'circle',
'nice': 'circle',
'kp': 'keep',
'haha': 'ahabhapha',
'ha': 'bha',
'ahead': 'aha',
'advertise': 'adverse',
'average': 'adverse',
'delivers': 'believer',
```

```
'exfoliators': 'exfoliant',
'moisturizer': 'moisturiser',
'barrier': 'arrive',
'serum': 'scrub',
'light': 'alright',
'course': 'cornhusker',
'wash': 'harsh',
'laisse': 'assume',
'overnight': 'convenient',
'appreciate': 'activate',
'con': 'com',
'help': 'el',
'comfortably': 'comfortable',
'lotion': 'boston',
'divine': 'advice',
'start': 'part',
'sugar': 'star',
'highly': 'high',
'dear': 'dead',
'mark': 'dark',
'drink': 'dark',
'darkness': 'dark',
'summer': 'assume',
'friendly': 'briefly',
'skincare': 'since',
'pigmentation': 'expectation',
'zit': 'vit',
'mix': 'fix',
'famous': 'fabulous',
'convenient': 'consistent',
'emollient': 'elsenot',
'cyst': 'crystal',
'claim': 'aim',
'benton': 'benefit',
'star': 'scar',
'cosrx': 'corsx',
'cure': 'cu',
'color': 'clog',
'brilliantly': 'ability',
'dryness': 'darkness',
'redness': 'excess',
'effect': 'affect',
'exfoliator': 'exfoliant',
'undereye': 'dewey',
'certainly': 'certain',
'clean': 'can',
'cleaning': 'certain',
'yes': 'eys',
'lot': 'hot',
'coconut': 'account',
'irritation': 'irradiate',
'first': 'dirt',
'dr': 'dirt',
'plane': 'laneige',
'exfoliate': 'exfoliant',
'expect': 'aspect',
'eye': 'enjoyed',
'dissapointed': 'disappoint',
'micellar': 'centella',
'part': 'earth',
'apr': 'apart',
'pat': 'past',
'curated': 'accurate',
'creep': 'create',
'need': 'congested',
```

```
'certain': 'ceramides',
'complexion': 'complain',
'conjunction': 'collection',
'instantly': 'brilliantly',
'saw': 'sad',
'addition': 'addicted',
'red': 'ready',
'lock': 'block',
'sunblock': 'lock',
'absorption': 'absorbtion',
'eventually': 'accidentally',
'original': 'hormonal',
'strong': 'astringent',
'treatment': 'improvement',
'astringent': 'adjustment',
'arent': 'adherent',
'decent': 'adherent',
'ingredient': 'incredibly',
'adverse': 'adhere',
'pick': 'pack',
'adherent': 'adhere',
'adore': 'adorable',
'ahewr': 'adhere',
'disappointing': 'disappoint',
'neogen': 'edge',
'affordable': 'adorable',
'mom': 'mm',
'beyond': 'benton',
'non': 'neogen',
'return': 'refund',
'cut': 'cu',
'two': 'top',
'past': 'part',
'crystal': 'christmas',
'combine': 'bombee',
'best': 'beat',
'break': 'beat',
'chalky': 'cakey',
'foot': 'boot',
'decide': 'besides',
'birth': 'birch',
'birthday': 'birth',
'product': 'pouch',
'arm': 'amd',
'pilling': 'ill',
'find': 'blind',
'problem': 'probably',
'maybe': 'many',
'money': 'honestly',
'point': 'oit',
'advice': 'addicted',
'sensitive': 'expensive',
'asian': 'african',
'fabric': 'eric',
'la': 'glad',
'crazy': 'clay',
'hyperpigmentation': 'expectation',
'alone': 'allinone',
'son': 'season',
'soft': 'smooth',
'instant': 'distinct',
'citrusy': 'citrus',
'aha': 'aaa',
'mascara': 'aaa',
'broken': 'broke',
```

```
'blur': 'blue',
'cleansing': 'cleaning',
'blowing': 'blessing',
'admit': 'addicted',
'currently': 'current',
'sting': 'astringent',
'sometimes': 'something',
'im': 'aim',
'annoying': 'annoy',
'anybody': 'annoy',
'balance': 'alain',
'antiaging': 'antiacne',
'adjustment': 'adjustable',
'adjustable': 'adjust',
'adult': 'adequate',
'jus': 'adjust',
'sad': 'husband',
'distinct': 'dissapointed',
'air': 'aim',
'aim': 'aczima',
'anti': 'ai',
'still': 'ill',
'oil': 'ill',
'pilled': 'ill',
'andor': 'adore',
'acnewell': 'acewell',
'acwell': 'acnewell',
'centella': 'acewell',
'empty': 'attempt',
'sokoglam': 'glam',
'cornhusker': 'confuse',
'snail': 'arsenal',
'fall': 'fail',
'lid': 'eyelid',
'deal': 'dalba',
'loveeee': 'loooovvveee',
'awful': 'adult',
'impressed': 'impresindible',
'incredibly': 'impresindible',
'alcoholfree': 'alcohol',
'dehydrate': 'combinationdehydrated',
'bed': 'bead',
'annoy': 'acneoily',
'step': 'asleep',
'asthma': 'althea',
'issue': 'assume',
'complaint': 'complain',
'coworker': 'cornhusker',
'dissolve': 'desolve',
'excellent': 'excelent',
'glam': 'glad',
'alternative': 'alternately',
'alternately': 'alternate',
'soap': 'soak',
'definite': 'definelty',
'korean': 'joan',
'morning': 'burning',
'animal': 'aczima',
'eczema': 'aczima',
'daytime': 'anytime',
'dehydration': 'dehydrate',
'hidrated': 'dehydrate',
'gentle': 'general',
'toner': 'tone',
'rave': 'abrasive',
```

```
'clay': 'clarify',
'awesome': 'assume',
'afterward': 'afraid',
'call': 'alli',
'moisturize': 'moisturiser',
'moisturiser': 'moisture',
'produc': 'pouch',
'purchase': 'pouch',
'akon': 'akin',
'bulk': 'bubbly',
'literally': 'alternately',
'eyeliner': 'eyelid',
'block': 'black',
'gel': 'el',
'budget': 'bouquet',
'drown': 'dozen',
'beautiful': 'beautefulthanks',
'eyelid': 'enjoyed',
'actuactually': 'accidentally',
'finally': 'accidentally',
'wow': 'two',
'adorable': 'adjustable',
'breathable': 'adjustable',
'daughter': 'ahewr',
'evaluate': 'adequate',
'lip': 'lid',
'film': 'fail',
'bearing': 'bang',
'chemical': 'charcoal',
'hormonal': 'charcoal',
'bought': 'boot',
'basic': 'asiatica',
'catch': 'batch',
'beach': 'batch',
'itchy': 'birthday',
'fresh': 'brush',
'achieve': 'ache',
'bump': 'bum',
'brown': 'broken',
'else': 'elmer',
'consistency': 'confidence',
'believe': 'baseline',
'outside': 'downside'}
```

In [50]:

```
words["word"].replace({'moisturizers': 'moisturize', 'moisturizes': 'moisturize', 'moisturizer': 'moisturize', 'clean': 'clear', 'cleaner': 'clear', 'bright': 'brightening', 'glowing': 'glow', 'dewey': 'soft', 'dewy': 'brightening', 'collage': 'collagen', 'pimples': 'acne', 'brightens': 'brightening', 'pimple': 'acne', 'breakout': 'acne', 'breakouts': 'acne', 'dehydration': 'hydration', 'clog': 'clogged', 'pore': 'pores', 'skin': 'dryness': 'dry', 'oil': 'oily', 'unclog': 'unclogged', 'line': 'line', 'blackhead': 'blackheads', 'reduces': 'reduce', 'improved': 'improved', 'cleanser': 'cleanse', 'cleanses': 'cleanse', 'hydrated': 'hydrated'})
```

In [42]:

```
words["word"].replace(match_dict, inplace=True)
words["word"].replace({'dehydration': 'dehydrate', 'cleanser': 'cleanse'}, inplace=True)
```

In [43]:

```
similarity, high_score = token_fuzzy(words, 85, fuzz.token_sort_ratio)
high_score.groupby(['word', 'score']).agg({'match': ' ', '.join'}).sort_values(['score'], ascending=False)
# create dictionary
match_dict = dict(zip(high_score.match, high_score.word))
```

```
In [44]: words["word"].replace({'absorbed': 'absorb', 'consistent': 'consistency', 'wrk': 'work',
                            'clear', 'clean': 'clear', 'cleaner': 'clear', 'brighten': 'bright',
                            'applies': 'apply', 'actually': 'actual', 'fave': 'fav', 'friendly':
                            'lot': 'alot', 'glowing': 'glow', 'moisturiser': 'moisturize'}, inplace=True)
```

```
In [45]: similarity, high_score = token_fuzzy(words, 83, fuzz.token_sort_ratio)
high_score.groupby(['word', 'score']).agg({'match': ' ', '.join}).sort_values(['score'], ascending=False)
# create dictionary
match_dict = dict(zip(high_score.match, high_score.word))
```

```
In [46]: match_dict
```

```
Out[46]: {'charge': 'change',
          'thing': 'nothing',
          'different': 'difference',
          'lighten': 'light',
          'enjoyed': 'enjoy',
          'extract': 'extra',
          'cover': 'convert',
          'appear': 'apparau',
          'couldnt': 'could',
          'sticker': 'stick',
          'refresh': 'fresh',
          'freshly': 'fresh',
          'freshen': 'fresh',
          'bitter': 'better',
          'gently': 'gentle',
          'cumple': 'couple',
          'summer': 'bummer',
          'pigmentation': 'hyperpigmentation',
          'nightly': 'night',
          'effect': 'affect',
          'exfoliator': 'exfoliate',
          'others': 'bother',
          'bring': 'bearing',
          'soothe': 'smooth',
          'burning': 'bring',
          'least': 'atleast',
          'light': 'alright',
          'right': 'alright'}
```

```
In [47]: words["word"].replace({'lighten': 'light', 'enjoyed': 'enjoy', 'sticker': 'stick', 'refres':
                            'freshly': 'fresh', 'freshen': 'fresh', 'gently': 'gentle', 'nightl':
                            'exfoliator': 'exfoliate', 'moisturiser': 'moisturize'}, inplace=True)
```

```
In [48]: similarity, high_score = token_fuzzy(words, 81, fuzz.token_sort_ratio)
high_score.groupby(['word', 'score']).agg({'match': ' ', '.join}).sort_values(['score'], ascending=False)
# create dictionary
match_dict = dict(zip(high_score.match, high_score.word))
```

```
In [49]: match_dict
```

```
Out[49]: {'moisturizing': 'moisturize',
          'fragrant': 'fragrance',
          'continue': 'container',
          'irritate': 'irradiate'}
```

```
In [50]: words["word"].replace({'moisturizing': 'moisturize', 'fragrant': 'fragrance'}, inplace=True)
```

```
In [51]: similarity, high_score = token_fuzzy(words, 79, fuzz.token_sort_ratio)
high_score.groupby(['word', 'score']).agg({'match': ' ', '.join}).sort_values(['score'], ascending=False)
```

```
# create dictionary
match_dict = dict(zip(high_score.match, high_score.word))
```

```
In [52]: words["word"].replace({'absorbent': 'absorb', 'lovely': 'love', 'loveee': 'love',
                           'screen': 'sunscreen', 'disappointing': 'disappointed'}, inplace=True)
```

```
In [53]: similarity, high_score = token_fuzzy(words, 77, fuzz.token_sort_ratio)
high_score.groupby(['word', 'score']).agg({'match': ' '.join}).sort_values(['score'], ascending=False)
# create dictionary
match_dict = dict(zip(high_score.match, high_score.word))
```

```
In [54]: words["word"].replace({'applying': 'apply', 'breaking': 'break', 'lightyet': 'light', 'impulsive': 'impulsive',
                           'purchasing': 'purchase', 'comfortable': 'comfort', 'becausee': 'because',
                           'fav': 'fave', 'annoying': 'annoy', 'amaaaaazing': 'amazing'}, inplace=True)
```

```
In [55]: similarity, high_score = token_fuzzy(words, 75, fuzz.token_sort_ratio)
high_score.groupby(['word', 'score']).agg({'match': ' '.join}).sort_values(['score'], ascending=False)
# create dictionary
match_dict = dict(zip(high_score.match, high_score.word))
```

```
In [56]: words["word"].replace({'buyyy': 'buy', 'felt': 'feel', 'badly': 'bad', 'added': 'add', 'melting': 'melting',
                           'timy': 'tiny', 'wowed': 'wow', 'agin': 'aging', 'brightness': 'brightness'})
```

```
In [57]: BOW_df = pd.DataFrame(words)
```

```
In [60]: "aging" in words
```

```
Out[60]: False
```

```
In [62]: X = pd.get_dummies(BOW_df)
X.columns = X.columns.str.replace(r"[word_]", "")
```

```
In [75]: "pore" in X
```

```
Out[75]: False
```

```
In [53]: df_merged = productReviews_df.merge(X, how='inner', left_index=True, right_index=True)
```

```
In [54]: df_merged.to_csv('reviews_preprocess.csv')
```