#!/usr/bin/env python2.7

#-\*- coding: utf-8 -\*-

import ast

# import json

import pandas as pd

import re

from wordcloud import WordCloud

from nltk.corpus import stopwords

from textblob import TextBlob

import numpy as np

import matplotlib.pyplot as plt

metadata\_filename = 'C:\Users\Shovan\Desktop\Dissertation\_New\Datasets\\metadata\_musical\_parent.json'

review\_filename = 'C:\Users\Shovan\Desktop\Dissertation\_New\Datasets\\reviews\_musical\_parent.json'

musical\_instruments = ['Guitar', 'Electric Guitar', 'Piano', 'Microphone', 'Drum',

'Violin', 'Mixer', 'Ukulele', 'Saxophone', 'Trumpet',

'Acoustic Guitar', 'Accordion', 'Cello', 'Clarinet',

'Flute', 'Bass Guitar', 'Classical Guitar', 'Xylophone',

'Harmonica', 'Trombone', 'Oboe', 'Cajon', 'Banjo', 'Mandolin',

'Bassoon', 'Viola', 'Double Bass', 'Synthesizer', 'Bagpipes',

'Harp', 'Tuba', 'Ocarina', 'Bell', 'Theremin', 'Bongo',

'Tambourine', 'Sitar', 'Lute', 'Marimba', 'Melodica',

'Lyre', 'Oud', 'Hang', 'Keyboard']

def read\_metadata(filename=""):

f = open(filename)

metadata = []

amazon\_md\_df = pd.DataFrame()

for row in f.readlines():

metadata.append(ast.literal\_eval(row))

amazon\_md\_df['product\_id'] = map(lambda data: data.get('asin', ''), metadata)

amazon\_md\_df['Description'] = map(lambda data: data.get('title', ''), metadata)

amazon\_md\_df['Price'] = map(lambda data: data.get('price', ''), metadata)

amazon\_md\_df['Manufacturer'] = map(lambda data: data.get('brand', ''), metadata)

return amazon\_md\_df

def preprocess\_metadata(amazon\_md\_df):

for i in amazon\_md\_df.columns:

amazon\_md\_df[i][amazon\_md\_df[i].apply(lambda i: True if re.search('^\s\*$', str(i)) else False)]=float('nan')

amazon\_md\_df\_updated = pd.DataFrame()

amazon\_md\_df\_updated = amazon\_md\_df.dropna(how='any')

# print amazon\_md\_df\_updated

music\_instruments\_set = set(musical\_instruments + [''])

music\_instruments\_set = map(str.lower, music\_instruments\_set)

empty\_list = []

for index, rows in amazon\_md\_df\_updated.iterrows():

empty\_set = set([])

newline = rows['Description'].lower().split()

for word in newline:

if word.endswith('s'):

# Replace with Snowball Stemmer

new\_word = re.sub('s$|es$', '', word)

empty\_list.append(new\_word)

else:

empty\_list.append(word)

# print empty\_list

new\_string = ' '.join(word for word in empty\_list)

amazon\_md\_df\_updated.set\_value(index, 'Description', new\_string)

updated\_newline = new\_string.split()

abc = set(music\_instruments\_set).intersection(updated\_newline)

empty\_set = abc.intersection(music\_instruments\_set)

empty\_list = list(empty\_set)

if len(empty\_list) == 0:

amazon\_md\_df\_updated.set\_value(index, 'Description', None)

else:

amazon\_md\_df\_updated.set\_value(index, 'Description', empty\_list[0])

return amazon\_md\_df\_updated

def read\_preprocess\_review(filename=""):

g = open(filename)

amazon\_review\_df = pd.DataFrame()

review\_data = []

for line in g.readlines():

review\_data.append(ast.literal\_eval(line))

amazon\_review\_df['product\_id'] = map(lambda data: data.get('asin', ''), review\_data)

amazon\_review\_df['Reveiwer ID'] = map(lambda data: data.get('reviewerID', ''), review\_data)

amazon\_review\_df['Reviewer Comments'] = map(lambda data: data.get('reviewText', ''), review\_data)

amazon\_review\_df['Overall Rating'] = map(lambda data: data.get('overall', ''), review\_data)

amazon\_review\_df['Overall Summary'] = map(lambda data: data.get('summary', ''), review\_data)

amazon\_review\_df['Reviewer Name'] = map(lambda data: data.get('reviewerName', ''), review\_data)

sentiments\_list = []

for d, rows in amazon\_review\_df.iterrows():

row = rows['Reviewer Comments']

testimonial = TextBlob(row)

sentiment\_polarity = testimonial.sentiment.polarity

sentiments\_list.append(sentiment\_polarity)

# print sentiments\_list

amazon\_review\_df['Sentiments Polarity'] = pd.DataFrame({'Sentiments Polarity': sentiments\_list})

# print amazon\_review\_df

return amazon\_review\_df

def filter\_by\_brand(preprocessed\_metadata, brandname):

if len(brandname):

filtered\_by\_brand = preprocessed\_metadata[preprocessed\_metadata['Manufacturer'].str.contains(brandname, case=False)]

return filtered\_by\_brand

else:

return preprocessed\_metadata

def filter\_by\_price(preprocessed\_metadata, minprice, maxprice):

if minprice != maxprice:

filtered\_by\_price = preprocessed\_metadata[preprocessed\_metadata['Price'].between(minprice, maxprice, inclusive=True)]

return filtered\_by\_price

else:

return preprocessed\_metadata

def filter\_by\_productname(preprocessed\_metadata, productname):

if len(productname):

filtered\_by\_productname = preprocessed\_metadata[preprocessed\_metadata['Description'].str.contains(productname, na=False, case=False)]

return filtered\_by\_productname

else:

return preprocessed\_metadata

def filter\_by\_keyword(metadata, review\_data, keyword):

if len(keyword):

review\_ids = review\_data[review\_data['Reviewer Comments'].str.contains(keyword)]['product\_id']

filtered = metadata[metadata['product\_id'].isin(review\_ids)]

return filtered

else:

return metadata

def remove\_stopwords(text):

text = re.sub('\W+', ' ', text)

text = re.sub('\s+\S\s+', ' ', text)

text = re.sub(' +', ' ', text)

words = text.split(" ")

processed\_words = ' '.join([word for word in words if word not in (stopwords.words('english'))])

# print processed\_words

return processed\_words

def get\_filtered\_reviews(df, id\_list):

filtered\_reviews = df[df['product\_id'].isin(id\_list)]

# filter\_count = filtered\_reviews.shape[0]

reviews = []

rating = 0

for index, rows in filtered\_reviews.iterrows():

reviews.append(rows['Reviewer Comments'].lower())

rating += rows['Overall Rating']

# print "Total number of reviews of the instrument",len(reviews)

# print "Average of all the overall ratings",rating/filter\_count

total\_reviews = ' '.join(reviews)

music\_instruments\_list = musical\_instruments + ['']

music\_instruments\_list = map(str.lower, music\_instruments\_list)

for word in music\_instruments\_list:

# print word

if word in total\_reviews:

total\_reviews = re.sub(word, '', total\_reviews)

# print total\_reviews,"total\_reviews"

total\_reviews = remove\_stopwords(total\_reviews)

# print total\_reviews, "stopwords"

return total\_reviews

def wordcloud(dataframe, dataframereview):

id\_list = []

for index, rows in dataframe.iterrows():

id\_list.append(rows["product\_id"])

# print(id\_list)

text = get\_filtered\_reviews(dataframereview, id\_list)

# print(text)

# Generate a word cloud image

wordcloud = WordCloud(background\_color="white").generate(text)

# Display the generated image:

# the matplotlib way:

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis("off")

plt.savefig('image2.png')

# lower max\_font\_size

wordcloud = WordCloud(max\_font\_size=40, background\_color="white").generate(text)

plt.figure()

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis("off")

plt.savefig('image1.png')

plt.show()

def main():

brandname = ""

minprice = 0

maxprice = 0

productname = "guitar"

metadata\_df = read\_metadata(metadata\_filename)

review\_df = read\_preprocess\_review(review\_filename)

preprocessed\_metadata = preprocess\_metadata(metadata\_df)

brand\_filter = filter\_by\_brand(preprocessed\_metadata, brandname)

price\_filter = filter\_by\_price(brand\_filter, minprice, maxprice)

productname\_filter = filter\_by\_productname(price\_filter, productname)

# wordcloud(dataframe=brand\_filter, dataframereview=review\_df)

# wordcloud(dataframe=price\_filter, dataframereview=review\_df)

wordcloud(dataframe=productname\_filter, dataframereview=review\_df)

if \_\_name\_\_ == '\_\_main\_\_':

main()