Amazon_Apparel_Recommendation_Assignment

November 26, 2018

```
In [2]: #import all the necessary packages.

        from PIL import Image
        import requests
        from io import BytesIO
        import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        import warnings
        from bs4 import BeautifulSoup
        from nltk.corpus import stopwords
        from nltk.tokenize import word_tokenize
        import nltk
        import math
        import time
        import re
        import os
        import seaborn as sns
        from collections import Counter
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.metrics.pairwise import cosine_similarity
        from sklearn.metrics import pairwise_distances
        from matplotlib import gridspec
        from scipy.sparse import hstack
        import plotly
        import plotly.figure_factory as ff
        from plotly.graph_objs import Scatter, Layout

        plotly.offline.init_notebook_mode(connected=True)
        warnings.filterwarnings("ignore")
```

```
In [3]: data = pd.read_pickle('pickels/16k_apperal_data_preprocessed')
        data.head()
```

```
Out[3]:           asin                           brand             color  \
        4   B004GSI2OS                      FeatherLite  Onyx Black/ Stone
```

```
6   B012YX2ZPI  HX-Kingdom Fashion T-shirts             White
15  B003BSRPB0                  FeatherLite             White
27  B014ICEJ1Q                        FNC7C            Purple
46  B01NACPBG2                 Fifth Degree             Black

                                     medium_image_url product_type_name  \
4   https://images-na.ssl-images-amazon.com/images...             SHIRT
6   https://images-na.ssl-images-amazon.com/images...             SHIRT
15  https://images-na.ssl-images-amazon.com/images...             SHIRT
27  https://images-na.ssl-images-amazon.com/images...             SHIRT
46  https://images-na.ssl-images-amazon.com/images...             SHIRT

                                                title formatted_price
4   featherlite ladies long sleeve stain resistant...          $26.26
6   womens unique 100 cotton  special olympics wor...           $9.99
15  featherlite ladies moisture free mesh sport sh...          $20.54
27  supernatural chibis sam dean castiel neck tshi...           $7.39
46  fifth degree womens gold foil graphic tees jun...           $6.95
```

```
In [4]: from gensim.models import Word2Vec
        from gensim.models import KeyedVectors
        import pickle

        with open('word2vec_model', 'rb') as handle:
            model = pickle.load(handle)
```

```
In [8]: idf_title_vectorizer = CountVectorizer()
        idf_title_features = idf_title_vectorizer.fit_transform(data['title'])
```

```
In [5]: # Utility functions

        def get_word_vec(sentence, doc_id, m_name):
            # sentence : title of the apparel
            # doc_id: document id in our corpus
            # m_name: model information it will take two values
                # if  m_name == 'avg', we will append the model[i], w2v representation of word i
                # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)
            vec = []
            for i in sentence.split():
                if i in vocab:
                    if m_name == 'weighted' and i in  idf_title_vectorizer.vocabulary_:
                        vec.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[i
                    elif m_name == 'avg':
                        vec.append(model[i])
                else:
                    # if the word in our courpus is not there in the google word2vec corpus, we
                    vec.append(np.zeros(shape=(300,)))
            # we will return a numpy array of shape (#number of words in title * 300 ) 300 = len
```

```
    # each row represents the word2vec representation of each word (weighted/avg) in giv
    return  np.array(vec)

def get_distance(vec1, vec2):
    # vec1 = np.array(#number_of_words_title1 * 300), each row is a vector of length 300
    # vec2 = np.array(#number_of_words_title2 * 300), each row is a vector of length 300

    final_dist = []
    # for each vector in vec1 we caluclate the distance(euclidean) to all vectors in vec
    for i in vec1:
        dist = []
        for j in vec2:
            # np.linalg.norm(i-j) will result the euclidean distance between vectors i,
            dist.append(np.linalg.norm(i-j))
        final_dist.append(np.array(dist))
    # final_dist = np.array(#number of words in title1 * #number of words in title2)
    # final_dist[i,j] = euclidean distance between vectors i, j
    return np.array(final_dist)


def heat_map_w2v(sentence1, sentence2, url, doc_id1, doc_id2, model):
    # sentance1 : title1, input apparel
    # sentance2 : title2, recommended apparel
    # url: apparel image url
    # doc_id1: document id of input apparel
    # doc_id2: document id of recommended apparel
    # model: it can have two values, 1. avg 2. weighted

    #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/avg
    s1_vec = get_word_vec(sentence1, doc_id1, model)
    #s2_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/avg
    s2_vec = get_word_vec(sentence2, doc_id2, model)

    # s1_s2_dist = np.array(#number of words in title1 * #number of words in title2)
    # s1_s2_dist[i,j] = euclidean distance between words i, j
    s1_s2_dist = get_distance(s1_vec, s2_vec)



    # devide whole figure into 2 parts 1st part displays heatmap 2nd part displays image
    gs = gridspec.GridSpec(2, 2, width_ratios=[4,1],height_ratios=[2,1])
    fig = plt.figure(figsize=(15,15))

    ax = plt.subplot(gs[0])
    # ploting the heap map based on the pairwise distances
    ax = sns.heatmap(np.round(s1_s2_dist,4), annot=True)
    # set the x axis labels as recommended apparels title
    ax.set_xticklabels(sentence2.split())
```

```
            # set the y axis labels as input apparels title
            ax.set_yticklabels(sentence1.split())
            # set title as recommended apparels title
            ax.set_title(sentence2)

            ax = plt.subplot(gs[1])
            # we remove all grids and axis labels for image
            ax.grid(False)
            ax.set_xticks([])
            ax.set_yticks([])
            display_img(url, ax, fig)

            plt.show()

In [6]: # vocab = stores all the words that are there in google w2v model
        # vocab = model.wv.vocab.keys() # if you are using Google word2Vec

        vocab = model.keys()
        # this function will add the vectors of each word and returns the avg vector of given se
        def build_avg_vec(sentence, num_features, doc_id, m_name):
            # sentace: its title of the apparel
            # num_features: the lenght of word2vec vector, its values = 300
            # m_name: model information it will take two values
                # if  m_name == 'avg', we will append the model[i], w2v representation of word i
                # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)

            featureVec = np.zeros((num_features,), dtype="float32")
            # we will intialize a vector of size 300 with all zeros
            # we add each word2vec(wordi) to this fetureVec
            nwords = 0

            for word in sentence.split():
                nwords += 1
                if word in vocab:
                    if m_name == 'weighted' and word in  idf_title_vectorizer.vocabulary_:
                        featureVec = np.add(featureVec, idf_title_features[doc_id, idf_title_vec
                    elif m_name == 'avg':
                        featureVec = np.add(featureVec, model[word])
            if(nwords>0):
                featureVec = np.divide(featureVec, nwords)
            # returns the avg vector of given sentance, its of shape (1, 300)
            return featureVec

In [9]: doc_id = 0
        w2v_title_weight = []
        # for every title we build a weighted vector representation
        for i in data['title']:
            w2v_title_weight.append(build_avg_vec(i, 300, doc_id,'weighted'))
            doc_id += 1
        # w2v_title = np.array(# number of doc in courpus * 300), each row corresponds to a doc
        w2v_title_weight = np.array(w2v_title_weight)
```

```
In [14]: w2v_title_weight.shape
```

```
Out[14]: (16042, 300)
```

```
In [15]: # some of the brand values are empty.
         # Need to replace Null with string "NULL"
         data['brand'].fillna(value="Not given", inplace=True )

         # replace spaces with hypen
         brands = [x.replace(" ", "-") for x in data['brand'].values]
         colors = [x.replace(" ", "-") for x in data['color'].values]

         brand_vectorizer = CountVectorizer()
         brand_features = brand_vectorizer.fit_transform(brands)

         color_vectorizer = CountVectorizer()
         color_features = color_vectorizer.fit_transform(colors)
```

```
In [17]: from keras.preprocessing.image import ImageDataGenerator
         from keras.models import Sequential
         from keras.layers import Dropout, Flatten, Dense
         from keras import applications
```

```
Using TensorFlow backend.
```

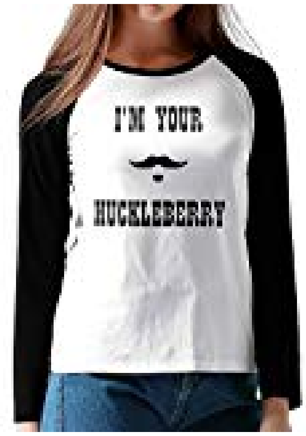
```
In [16]: bottleneck_features_train = np.load('16k_data_cnn_features.npy')
         asins = np.load('16k_data_cnn_feature_asins.npy')
```

```
In [19]: bottleneck_features_train.shape
```

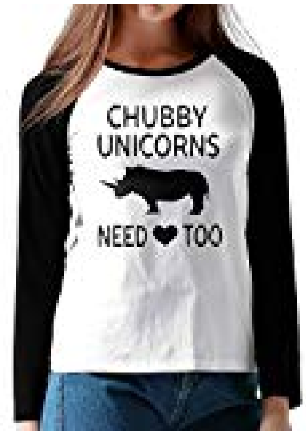
```
Out[19]: (16042, 25088)
```

```
In [26]: def heat_map_w2v_brand(sentance1, sentance2, url, doc_id1, doc_id2, df_id1, df_id2, mod

             # sentance1 : title1, input apparel
             # sentance2 : title2, recommended apparel
             # url: apparel image url
             # doc_id1: document id of input apparel
             # doc_id2: document id of recommended apparel
             # df_id1: index of document1 in the data frame
             # df_id2: index of document2 in the data frame
             # model: it can have two values, 1. avg 2. weighted

             #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/av
             s1_vec = get_word_vec(sentance1, doc_id1, model)
             #s2_vec = np.array(#number_of_words_title2 * 300), each row is a vector(weighted/av
             s2_vec = get_word_vec(sentance2, doc_id2, model)

             # s1_s2_dist = np.array(#number of words in title1 * #number of words in title2)
             # s1_s2_dist[i,j] = euclidean distance between words i, j
             s1_s2_dist = get_distance(s1_vec, s2_vec)

             data_matrix = [['Asin','Brand', 'Color', 'Product type'],
                        [data['asin'].loc[df_id1],brands[doc_id1], colors[doc_id1]], # input app
                        [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2]]] # recommond

             colorscale = [[0, '#1d004d'],[.5, '#f2e5ff'],[1, '#f2e5d1']] # to color the heading

             # we create a table with the data_matrix
             table = ff.create_table(data_matrix, index=True, colorscale=colorscale)
             # plot it with plotly
             plotly.offline.iplot(table, filename='simple_table')

             # devide whole figure space into 25 * 1:10 grids
             gs = gridspec.GridSpec(25, 15)
             fig = plt.figure(figsize=(25,5))

             # in first 25*10 grids we plot heatmap
             ax1 = plt.subplot(gs[:, :-5])
             # ploting the heap map based on the pairwise distances
             ax1 = sns.heatmap(np.round(s1_s2_dist,6), annot=True)
             # set the x axis labels as recommended apparels title
             ax1.set_xticklabels(sentance2.split())
             # set the y axis labels as input apparels title
             ax1.set_yticklabels(sentance1.split())
             # set title as recommended apparels title
             ax1.set_title(sentance2)

             # in last 25 * 10:15 grids we display image
             ax2 = plt.subplot(gs[:, 10:16])
             # we dont display grid lins and axis labels to images
             ax2.grid(False)
             ax2.set_xticks([])
             ax2.set_yticks([])

             # pass the url it display it
             display_img(url, ax2, fig)

             plt.show()


         def display_img(url,ax,fig):
```

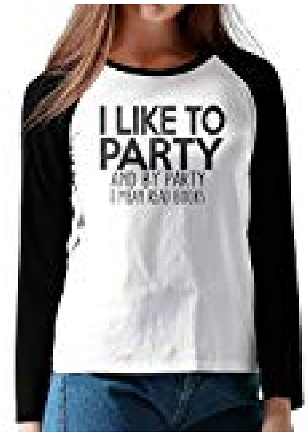
```
             # we get the url of the apparel and download it
             response = requests.get(url)
             img = Image.open(BytesIO(response.content))
             # we will display it in notebook
             plt.imshow(img)

In [50]: asins = list(asins)

         # load the original 16K dataset
         data = pd.read_pickle('pickels/16k_apperal_data_preprocessed')
         df_asins = list(data['asin'])


         from IPython.display import display, Image, SVG, Math, YouTubeVideo


         #get similar products using CNN features (VGG-16)
         def get_similar_products_cnn(doc_id, wt, wb, wc, wi, num_results):
             doc_id = asins.index(df_asins[doc_id])

             idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].resha
             brand_dist = pairwise_distances(brand_features, brand_features[doc_id])
             color_dist = pairwise_distances(color_features, color_features[doc_id])
             bottleneck_features_dist = pairwise_distances(bottleneck_features_train, bottleneck

             pairwise_dist = (wt * idf_w2v_dist +  wb * brand_dist + wc * color_dist + wi * bott

             indices = np.argsort(pairwise_dist.flatten())[0:num_results]
             pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

             for i in range(len(indices)):
                 rows = data[['medium_image_url','title']].loc[data['asin']==asins[indices[i]]]
                 for indx, row in rows.iterrows():
                     display(Image(url=row['medium_image_url'], embed=True))
                     print('Product Title: ', row['title'])
                     print('Euclidean Distance from input image:', pdists[i])
                     print('Amazon Url: www.amzon.com/dp/'+ asins[indices[i]])

         get_similar_products_cnn(12562,50,30,20,100,10)
```



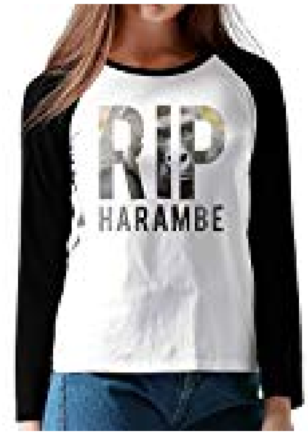
```
Product Title:  im huckleberry doc holliday 34sleeve raglan long sleeve
Euclidean Distance from input image: 0.0
Amazon Url: www.amzon.com/dp/B01MG2JKHS
```



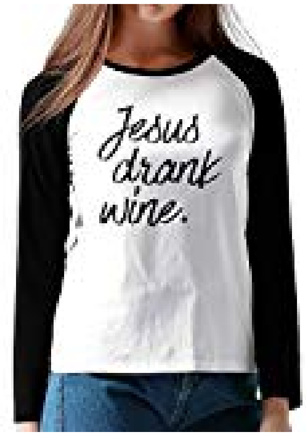
```
Product Title:  chubby unicorn need love 34sleeve raglan long sleeve
Euclidean Distance from input image: 9.20741800317262
Amazon Url: www.amzon.com/dp/B01M67SUP1
```



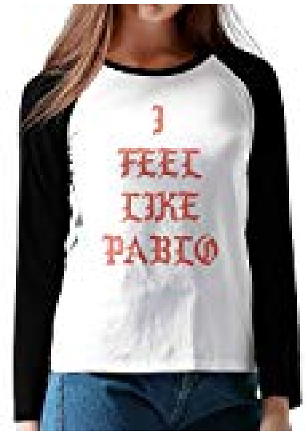
```
Product Title:  bibliophile literary like party 34sleeve raglan long sleeve
Euclidean Distance from input image: 9.67284268457086
Amazon Url: www.amzon.com/dp/B01MFAZI9M
```



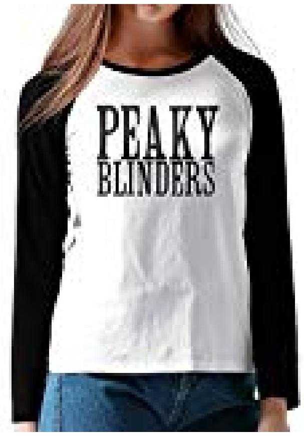
```
Product Title:  rip harambe cartoon 17th birthday killed 34sleeve raglan long sleeve
Euclidean Distance from input image: 10.001156463713391
Amazon Url: www.amzon.com/dp/B01MPZYO3I
```



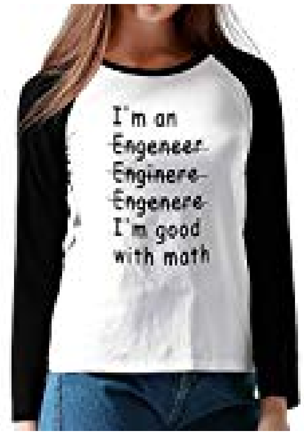
```
Product Title:  jesus drank wine 34sleeve raglan long sleeve
Euclidean Distance from input image: 10.080035934538587
Amazon Url: www.amzon.com/dp/B01MG2KH61
```



```
Product Title:  kanye west pinterest 34sleeve raglan long sleeve
Euclidean Distance from input image: 10.090882186979993
Amazon Url: www.amzon.com/dp/B01MG2KZTK
```



```
Product Title:  xjbd womens peaky drama series long sleeve blended baseball shirt size xl
Euclidean Distance from input image: 10.511070747465833
Amazon Url: www.amzon.com/dp/B01M0B32NI
```



```
Product Title:  engineer good math 34sleeve raglan long sleeve
Euclidean Distance from input image: 10.687035808653578
Amazon Url: www.amzon.com/dp/B01MG2PTEZ
```

```
Product Title:  tell time pm 34sleeve raglan long sleeve
Euclidean Distance from input image: 10.695306701750502
Amazon Url: www.amzon.com/dp/B01MFBI9GM
```

```
Product Title:  choose violence 34sleeve raglan long sleeve
Euclidean Distance from input image: 10.929325682544075
Amazon Url: www.amzon.com/dp/B01MPZY033
```

** Summary and Observation ** 1. Used Text, brand, color and image features to recommend similar products. 2. Given weights feature for all 4 features and played around with it, to see how image recommendations varry based on features.