04ClusteringPracticeWord2Vec

November 4, 2018

```
In [1]: import pandas as pd
        import numpy as np
        import nltk
        from pprint import pprint

        from nltk.corpus import stopwords
        # if this is first time you use nltk, please open terminal and type codes bellow
        #   >>> import nltk
        #   >>> nltk.download('stopwords')
        #   >>> nltk.download('punkt')

        stops = set(stopwords.words('english'))
        import string
        puns = string.punctuation
```

1 Word2Vec

1. 簡介

   Word2Vec 其實是 Word to Vector 的簡稱，意在將每一個字轉換成一條向量，並讓這字的語意透過這條向量描繪出來。早期做自然語言處理時，很難對讓電腦對詞背後的意思有更深一層的理解，因此詞與詞之間的關係很難被挖掘出來，像是相似詞、相反詞、對應詞等，因此 Word2Vec 在這樣的背景下產生就顯得極其珍貴。

2. 作用 & 賣點

   1. 它可以找到相似的字。
   2. 它可以加減，像是 Taiwan-Taipei=Germany-Berlin。

3. 訓練方法: 參照這篇網誌

4. 已經訓練好的 model:

   1. Various Models
   2. 本課程使用之GloVe Word2Vec(時間考量已經壓縮過，只留下這個文件中會用到的字)。

5. 專案: 將 e-commerce 商品標籤分群

2 Load Data

```
In [2]: with open('all_categories.list', 'r', encoding='utf8') as f:
            all_categories = np.array(eval(f.read()))
        print("商品標籤個數:", len(all_categories))
        print("前 10 個商品標籤:", all_categories[:10])
```

```
商品標籤個數: 910
前 10 個商品標籤: ['Small Animal' 'Kitchen' 'Fragrance' 'Track & Sweat Suits' 'Wallet'
 'Favors' 'Quilts' 'Sticker' 'Pets' 'Skirt']
```

```
In [3]: word_vec_mapping = {}
        path = "glove.twitter.27B.50d.txt"

        # 打開上述檔案，並將每一行中的第一個詞作為 key，後面的數字做為向量，加入到 word_vec_mappi
        with open(path, 'r', encoding='utf8') as f:  ## 這個文檔的格式是一行一個字並配上他的向量
            for line in f:
                #=============your works starts===============#
                tokens =
                token =
                vec =
                word_vec_mapping[token] =
                #==============your works ends================#

        vec_dimensions = len(word_vec_mapping.get('men'))
        print("vec_dimensions:", vec_dimensions)
        print("word_vec_mapping length:", len(list(word_vec_mapping.items())))
        pprint(list(word_vec_mapping.items())[:5])
        # vec_dimensions: 50
        # word_vec_mapping length: 947
        # [('shoes',
        #   array([-0.75313002, -1.78719997,  0.14522   , -0.29681   ,  0.12436   ,
        #        -0.40922999,  1.22679996,  0.50806999,  0.27913001,  0.34277001,
        #        -0.013902  ,  1.52499998, -3.44880009,  1.05630004, -0.49985   ,
```

```
vec_dimensions: 50
word_vec_mapping length: 947
[('protection',
  array([-2.6041e-01,  9.3470e-03, -1.2779e+00,  9.3997e-01,  1.3464e-01,
         2.8652e-01,  4.7567e-01, -5.1847e-01,  6.8337e-01, -6.8621e-01,
         2.5913e-01,  2.9725e-01, -2.7242e+00,  3.8473e-01,  1.2560e+00,
         9.2542e-01, -1.0193e-01,  1.5966e-01,  2.2935e-03, -3.8759e-01,
        -1.0683e+00, -5.3661e-01,  6.1156e-03,  1.5376e-01,  3.5490e-01,
         5.9846e-01,  8.5329e-02,  8.7829e-01,  2.1870e-01,  1.0114e+00,
        -6.9214e-03, -5.1215e-01, -2.7296e-01, -8.3198e-01,  8.5664e-01,
        -5.2144e-01, -4.6561e-01,  9.8429e-01, -6.7122e-01, -9.6129e-01,
         7.8881e-01, -6.1323e-01,  2.6551e-01, -3.9457e-01,  5.8291e-01,
        -5.6443e-01,  3.0565e-01,  4.1577e-02,  8.3677e-01, -4.5295e-01],
     dtype=float32)),
 ('headsets',
  array([-0.026823 , -0.092859 , -1.0312   ,  0.90884  , -0.49068  ,
         0.3701   ,  0.63185  , -0.61496  ,  0.57759  , -1.0434   ,
        -0.21304  ,  1.0892   , -0.67756  ,  0.39059  ,  0.12729  ,
        -0.39507  ,  0.070779 , -0.53489  ,  0.62134  ,  0.35858  ,
        -0.3978   , -0.053658 ,  0.59362  , -0.15411  , -0.50521  ,
         0.64195  , -0.34041  ,  0.33592  , -0.085267 ,  0.035649 ,
         0.53375  ,  0.034903 ,  0.54944  , -0.80055  ,  0.72882  ,
        -1.1041   ,  0.41168  , -0.53155  , -0.46107  ,  1.2108   ,
         1.3971   ,  1.1987   , -0.52795  , -0.0043981, -0.30671  ,
        -0.059243 , -0.079573 , -0.33629  ,  0.24877  ,  0.44079  ],
     dtype=float32)),
 ('jackets',
  array([ 8.5055e-02, -1.7499e+00, -1.7031e-01, -2.6836e-01, -4.0375e-01,
        -3.1689e-01,  9.3073e-01,  1.2195e-01,  5.3707e-01, -8.3221e-01,
         9.2084e-01,  1.1719e+00, -2.0878e+00,  6.3773e-01, -2.4071e-01,
         1.3865e-03, -1.3375e+00, -8.2863e-01, -6.2409e-02,  1.2643e-01,
         3.3143e-01, -4.6428e-02,  1.2935e+00, -8.8655e-01, -4.2175e-01,
         1.3437e+00,  3.2258e-01,  5.6446e-01, -3.7294e-01, -9.2323e-01,
         2.8631e-01,  1.0212e+00,  6.5575e-01, -1.7160e-01,  1.1947e+00,
        -1.6754e+00, -1.0792e+00,  2.9502e-01,  1.3830e-01,  9.4862e-01,
         5.9838e-01, -9.7969e-02,  2.2767e-01, -1.6997e-01, -6.8851e-02,
         1.1240e-01,  5.0171e-01, -1.7130e-01, -2.4996e-01,  2.0415e-01],
     dtype=float32)),
 ('cart',
  array([-9.5941e-01, -1.3244e-01, -4.0781e-01, -1.6105e-01,  9.0079e-01,
        -6.2495e-01,  1.9808e-02, -3.3392e-01,  6.9751e-01, -3.2706e-01,
         7.6056e-01,  9.7973e-01, -2.0602e+00,  2.8303e-02,  4.0289e-01,
         3.9693e-01,  3.7966e-01, -6.2229e-01, -2.0016e-03, -6.7711e-02,
        -2.3231e-01, -2.5877e-01,  6.0567e-01, -5.0645e-03, -1.0308e-01,
         8.1790e-01, -8.9303e-01,  2.6405e-01,  1.9210e-01, -4.2307e-01,
         2.5479e-01, -6.8431e-02, -1.4063e-02, -1.3592e+00,  1.5808e-01,
         2.1494e-01,  6.8527e-01,  5.2884e-01,  3.3263e-01,  2.5476e-02,
        -2.1072e-01,  1.3368e+00, -8.4683e-01, -7.6833e-01,  8.5318e-01,
         4.0324e-01, -3.4235e-02,  2.7536e-01,  1.2719e-01,  4.8708e-01],
     dtype=float32)),
 ('seats',
  array([ 0.26285  , -0.36878  ,  0.95363  ,  0.25466  , -0.21188  ,
        -0.50289  ,  0.92677  ,  0.46699  ,  0.26938  , -0.73464  ,
         0.68061  , -0.46538  , -2.9934   ,  0.15943  ,  0.92958  ,
         0.96445  , -0.19603  , -0.84996  , -0.57998  ,  0.22855  ,
        -0.80445  , -1.4504   ,  0.77642  , -0.049717 , -0.37453  ,
         0.99299  , -0.78962  , -0.33776  , -0.43973  , -0.35432  ,
         0.66495  ,  0.72955  ,  0.43549  ,  0.49489  ,  1.4032   ,
        -0.42153  ,  0.0055356,  0.2083   ,  0.24984  ,  0.24588  ,
         0.60279  ,  1.9941   , -0.25869  , -0.47908  ,  0.1719   ,
        -0.30145  ,  0.67101  , -0.20577  ,  0.54319  ,  0.74157  ],
      dtype=float32))]
```

3 Tokenize & Doc2Vec

```
In [4]: # 將每一個句子 (商品類別) 的詞彙
        # 1. 切割開來
        # 2. 去掉停用字 (set(stopwords.words('english')))
        # 3. 去掉標點符號 (string.punctuation)
        # 4. 轉小寫

        def tokenize(Doc):
            if pd.notnull(Doc):
                # 使用 nltk.wordpunct_tokenize 將 Doc 切開
                # 去掉停用字與標點符號，並轉小寫
                #=============your works starts===============#
                tokens =
                words =
                #==============your works ends================#
                return words
            else:
                return None

        print("before tokenize:", all_categories[0])
        print("after tokenize:", tokenize(all_categories[0]))
        print("before tokenize:", all_categories[3])
        print("after tokenize:", tokenize(all_categories[3]))
        # before tokenize: Small Animal
        # after tokenize: ['small', 'animal']
        # before tokenize: Track & Sweat Suits
        # after tokenize: ['track', 'sweat', 'suits']
```

```
before tokenize: Small Animal
after tokenize: ['small', 'animal']
before tokenize: Track & Sweat Suits
after tokenize: ['track', 'sweat', 'suits']
```

```
In [5]: test_arr = np.array([
            [1,2,3,4],
            [4,5,6,7],
            [7,8,9,10]
        ])
        # 請將 test_arr 中的三個 array 做 element-wise 的平均
        #=============your works starts===============#
        test_out =
        #==============your works ends================#
        print(test_out)
        # [4. 5. 6. 7.]

[4. 5. 6. 7.]


In [6]: def doc2vec(doc, word2vec=word_vec_mapping):
            if pd.notnull(doc):
                # 使用剛剛定義好的 tokenize 函式 tokenize doc，並指派到 trems
                # 找出每一個詞彙的代表向量 (word_vec_mapping)
                # 並平均 (element-wise) 所有出現的詞彙向量 (注意 axis=0)，作為 doc 的代表向量
                #=============your works starts===============#
                terms =
                termvecs =
                docvec =
                #===============your works ends=================#

            if np.sum(np.isnan(docvec)) > 0:
                ## 若找不到對應的詞向量，則給一條全部為零的向量，長度為原詞彙代表向量的長度 (vec_d
                #=============your works starts===============#
                docvec=
                #===============your works ends=================#
            return docvec

        print("before tokenize:", all_categories[3])
        print("output shape", doc2vec(all_categories[3]).shape)
        print("after tokenize:", doc2vec(all_categories[3])[:5])
        print("before tokenize:", all_categories[70])
        print("output shape", doc2vec(all_categories[70]).shape)
        print("after tokenize:", doc2vec(all_categories[70])[:5])
        # before tokenize: Track & Sweat Suits
        # output shape (50,)
        # after tokenize: [-0.76383996 -0.49650002  0.23154134 -0.16717     0.42855397]
        # before tokenize: Teethers
        # output shape (50,)
        # after tokenize: [0. 0. 0. 0. 0.]

before tokenize: Track & Sweat Suits
output shape (50,)
after tokenize: [-0.76383996 -0.49650002  0.23154134 -0.16717     0.42855397]
before tokenize: Teethers
output shape (50,)
after tokenize: [0. 0. 0. 0. 0.]


  avg = a.mean(axis)
  ret = ret.dtype.type(ret / rcount)
```

```
In [7]: # 將 doc2vec 應用到 all_categories 中的每一個元素上
        #=============your works starts===============#
        cat_vecs =
        #==============your works ends================#

        print("cat_vecs length:", len(cat_vecs))
        print(cat_vecs[3][:5])
        print(cat_vecs.shape)
        # cat_vecs length: 910
        # [-0.76383996 -0.49650002  0.23154134 -0.16717     0.42855397]
        # (910, 50)
```

```
cat_vecs length: 910
[-0.76383996 -0.49650002  0.23154134 -0.16717     0.42855397]
(910, 50)
```

4 Clustering

```
In [8]: from sklearn.cluster import KMeans
        from sklearn.cluster import AgglomerativeClustering
        from sklearn.cluster import DBSCAN
        from collections import Counter
        X = cat_vecs
        n_clusters= 20
```

4.1 K means

```
In [9]: # 請使用 kmeans 將商品類別分成 20 類
        #=============your works starts===============#
        kmeans =
        all_categories_labels_kmeans =
        #==============your works ends================#

        for i in range(3):
            print(";;".join(all_categories[all_categories_labels_kmeans==i]))
            print("=============")
```

```
Athletic Training;;Golf;;Golf Apparel;;Track & Field;;Boxing & MMA;;Lacrosse;;Hockey;;Badminto
=============
Small Animal;;Pets;;Animals;;Toddler;;Animal;;Art Doll;;Kids;;Human Figure Doll;;Afghan;;Doll
=============
Girls;;Face;;Fan Shop;;Dress Up & Pretend Play;;Sleep Positioners;;Watch;;Boys;;50 To 75 Years
=============
```

4.2 Hireachy(single link)

```
In [10]: # 請使用 hierachical(single link) 將商品類別分成 20 類
         #=============your works starts===============#
         hierachy =
         all_categories_labels_single =
         #==============your works ends================#


         for i in range(3):
             print(";;".join(all_categories[all_categories_labels_single==i]))
             print("=============")
```

```
Athletic Training;;Motorcycle;;Golf;;Golf Apparel;;Hiking & Camping;;Water Sports;;Bike & Skat
=============
Historical, Military;;Exercise;;Arts & Crafts;;Education & Teaching;;Writing;;Potty Training;;
=============
Quilts;;Calendars;;Diapering;;Plush;;Knitting Supplies;;Baskets & Bins;;Yarn;;Baguette;;Painti
=============
```

4.3 Hireachy(average link)

```
In [11]: # 請使用 hierachical(average link) 將商品類別分成 20 類
         #=============your works starts===============#
         hierachy =
         all_categories_labels_average =
         #==============your works ends================#


         for i in range(3):
             print(";;".join(all_categories[all_categories_labels_average==i]))
             print("=============")
```

```
Athletic Training;;Boxing & MMA;;Lacrosse;;Hockey;;Badminton;;Bowl;;Varsity;;Team Sports;;Athl
=============
Dusting;;Waxing;;Thermometers;;Oils & Fluids;;Toothbrushes;;Sponges;;Brushes & Applicators;;Br
=============
Eyes;;Feet;;Sleep Positioners;;Lips;;Relaxed
=============
```

4.4 Hireachy(complete link)

```
In [12]: # 請使用 hierachical(complete link) 將商品類別分成 20 類
         #=============your works starts===============#
         hierachy =
         all_categories_labels_complete =
         #==============your works ends================#
```

```
for i in range(3):
    print(";;".join(all_categories[all_categories_labels_complete==i]))
    print("=============")
```

```
Kitchen;;Photo Albums & Frames;;Plush;;Storage & Organization;;Knitting Supplies;;Puzzles;;Bas
=============
Pets;;Athletic Training;;Animals;;Golf;;Water Sports;;Exercise;;Track & Field;;Education & Tea
=============
Charm;;Beads;;Felted;;Bracelet;;Handmade;;Vintage & Collectibles;;Geekery;;Bead;;Glassware;;Wr
=============
```

4.5 DBSCAN

```
In [13]: # 請使用 hierachical(complete link) 將商品類別分成 20 類 (eps=0.3)
         #=============your works starts===============#
         dbscan =
         all_categories_labels_dbscan =
         #==============your works ends================#

         for i in [-1, 0]:
             print(";;".join(all_categories[all_categories_labels_dbscan==i]))
             print("=============")
```

```
=============
Teethers;;Playards;;Epilators;;Sweatercoat;;Rainwear;;Needlecraft;;Bedspreads & Coverlets;;Deh
=============
```

```
In [14]: all_categories_labels_dbscam = DBSCAN().fit_predict(X)
         Counter(all_categories_labels_dbscam)
         # Counter({-1: 898, 0: 12})
```

```
Out[14]: Counter({-1: 898, 0: 12})
```

```
In [15]: df_cat = pd.DataFrame(all_categories_labels_dbscam, index=all_categories, columns=['la
         print(list(df_cat[df_cat['label'] == 0].index))
         # ['Teethers', 'Playards', 'Epilators', 'Sweatercoat', 'Rainwear', 'Needlecraft', 'Be
```

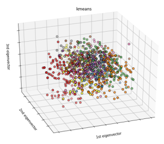
```
['Teethers', 'Playards', 'Epilators', 'Sweatercoat', 'Rainwear', 'Needlecraft', 'Bedspreads & 
```

5 PCA

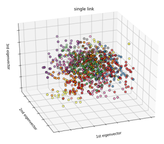
```
In [16]: import matplotlib.pyplot as plt
         from mpl_toolkits.mplot3d import Axes3D
```

```
        from sklearn.decomposition import PCA

        def draw_PCA(X, Y, title):
            fig = plt.figure(1, figsize=(8, 6))
            ax = Axes3D(fig, elev=-150, azim=110)

            X_reduced = PCA(n_components=3).fit_transform(X)
            ax.scatter(X_reduced[:, 0], X_reduced[:, 1], X_reduced[:, 2], c=Y,
                       cmap=plt.cm.Set1, edgecolor='k', s=40)
            ax.set_title(title)
            ax.set_xlabel("1st eigenvector")
            ax.w_xaxis.set_ticklabels([])
            ax.set_ylabel("2nd eigenvector")
            ax.w_yaxis.set_ticklabels([])
            ax.set_zlabel("3rd eigenvector")
            ax.w_zaxis.set_ticklabels([])

            plt.show()
```

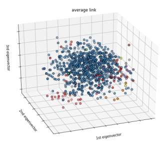
```
In [17]: draw_PCA(X, all_categories_labels_kmeans, 'kmeans')
```



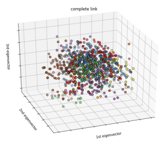
```
In [18]: draw_PCA(X, all_categories_labels_single, 'single link')
```



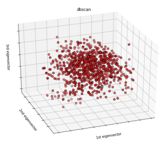
```
In [19]: draw_PCA(X, all_categories_labels_average, 'average link')
```



```
In [20]: draw_PCA(X, all_categories_labels_complete, 'complete link')
```



```
In [21]: draw_PCA(X, all_categories_labels_dbscan, 'dbscan')
```



6 Evaluation

```
In [22]: df_cat = pd.DataFrame(all_categories_labels_kmeans, index=all_categories, columns=['la
         for i in range(len(set(all_categories_labels_kmeans))):
             cats = list(df_cat[df_cat['label'] == i].index)
             print("cluster " + str(i) + ": ")
             print(list(cats))
             print("==============================================")
             print("==============================================")

cluster 0:
['Athletic Training', 'Golf', 'Golf Apparel', 'Track & Field', 'Boxing & MMA', 'Lacrosse', 'Ho
==============================================
==============================================
cluster 1:
['Small Animal', 'Pets', 'Animals', 'Toddler', 'Animal', 'Art Doll', 'Kids', 'Human Figure Dol
==============================================
==============================================
cluster 2:
['Girls', 'Face', 'Fan Shop', 'Dress Up & Pretend Play', 'Sleep Positioners', 'Watch', 'Boys',
==============================================
==============================================
cluster 3:
['Teethers', 'Playards', 'Humor', 'Epilators', 'Cargo', 'Sweatercoat', 'Rainwear', 'Needlecraf
==============================================
==============================================
cluster 4:
['Crewneck', 'Turtleneck', 'Vest', 'Gloves', 'Sweatshirt, Pullover', 'Blazers & Sport Coats',
=============================================
=============================================
cluster 5:
['Fragrance', 'Plate', 'Coffee & Tea Accessories', 'Pot Holder', 'Bubble Bath', 'Baguette', 'Di
=============================================
=============================================
cluster 6:
['Dusting', 'Monitors', 'Thermometers', 'Oils & Fluids', 'Garbage Disposals', 'Amplifiers & Ef
=============================================
=============================================
cluster 7:
=============================================
=============================================
cluster 8:
['Motorcycle', 'Storage & Organization', 'Medical Supplies & Equipment', 'Car Electronics & Ac
=============================================
=============================================
cluster 9:
['Books', 'Puzzles', 'Arts & Crafts', 'Paintings', 'Crafting', 'Christian Books & Bibles', 'No
=============================================
=============================================
cluster 10:
['Quilts', 'Diapering', 'Plush', 'Baskets & Bins', 'Yarn', 'Teapot', 'Bakeware', 'Grooming', '
=============================================
=============================================
cluster 11:
['Historical, Military', 'Trading Cards', 'Exercise', 'Education & Teaching', 'Health', 'Busin
=============================================
=============================================
cluster 12:
['Skirt', 'Dresses', 'Leggings', 'Jeans', 'Fashion Sneakers', 'Jerseys', 'Sweater', 'Dress Sui
=============================================
=============================================
cluster 13:
['Luggage', 'Pouch', 'Felted', 'Blouse', 'Lightweight', 'Tiered', 'Double Breasted', 'Frame',
=============================================
=============================================
cluster 14:
['Track & Sweat Suits', 'Wallet', 'Full-Length', 'Shams, Bed Skirts & Bed Frame Draperies', 'S
=============================================
=============================================
cluster 15:
['Charm', 'Beads', 'Bracelet', 'Handmade', 'Vintage & Collectibles', 'Bead', 'Earrings', 'Cabo
=============================================
=============================================
cluster 16:
['TV, Audio & Surveillance', 'Laptop', 'DVD & Blu-ray Players', 'Radio', 'Patch', 'Pin', 'Magi
=============================================
==============================================
cluster 17:
['Eyes', 'Waxing', 'Hair Relaxers', 'Nail Care', 'Pregnancy & Maternity', 'Scrubs & Body Treatr
==============================================
==============================================
cluster 18:
['Sticker', 'Photo Albums & Frames', "Men's Accessories", 'Changing Kits', 'GPS Accessories & I
==============================================
==============================================
cluster 19:
['Kitchen', 'Hiking & Camping', 'Water Sports', 'Home Decor', 'Kitchen Safety', 'Kitchen Storag
==============================================
==============================================
```

```
In [23]: df_cat = pd.DataFrame(all_categories_labels_single, index=all_categories, columns=['la
         for i in range(len(set(all_categories_labels_kmeans))):
             cats = list(df_cat[df_cat['label'] == i].index)
             print("cluster " + str(i) + ": ")
             print(sorted(list(cats)))
             print("==============================================")
             print("==============================================")
```

```
cluster 0:
['Athletic', 'Athletic Apparel', 'Athletic Training', 'Badminton', 'Ballet', 'Band & Orchestra
==============================================
==============================================
cluster 1:
['Afghan', 'Arts & Crafts', 'Bomber', 'Education & Teaching', 'Educational', 'Exercise', 'Fitne
==============================================
==============================================
cluster 2:
['Aceo', 'Baguette', 'Bakeware', 'Baskets', 'Baskets & Bins', 'Bedding', 'Bookmark', 'Bouquets
==============================================
==============================================
cluster 3:
['Backpacks & Carriers', 'Batteries', 'Binoculars & Telescopes', 'Brushes', 'Brushes & Applicat
==============================================
==============================================
cluster 4:
['Apron', 'Asymmetrical', 'Asymmetrical Hem', 'Beading', 'Capri, Cropped', 'Capris, Cropped',
==============================================
==============================================
cluster 5:
['Apparel', 'Belt', 'Blazer', 'Blazers & Sport Coats', 'Blouse', 'Boots', 'Bottoms', 'Buckle',
==============================================
==============================================
cluster 6:
```

```
['Accessory', 'Action Figures & Statues', 'Activity Centers & Entertainers', 'Art', 'Arts & Pho
==============================================
==============================================
cluster 7:
['100 Years or Older', '50 To 75 Years', '75 To 100 Years', 'All Other Sports', 'Baby', 'Baby &
==============================================
==============================================
cluster 8:
==============================================
==============================================
cluster 9:
['Accessories', 'Bathing Accessories', 'Beach Accessories', 'Custom', 'Dolls & Accessories', '
==============================================
==============================================
cluster 10:
['A-Line', 'Action Figure', 'Action, Adventure', 'Baby Gyms & Playmats', 'Baby Seats', 'Block'
==============================================
==============================================
cluster 11:
['Baseball', 'Baseball & Softball', 'Basketball', 'Bowl', 'Bowls', 'Football', 'Game', 'Games'
==============================================
==============================================
cluster 12:
['Antique', 'Bead', 'Beads', 'Bracelet', 'Bracelets', 'Brooch', 'Cabochon', 'Charm', 'Collecti
==============================================
==============================================
cluster 13:
['Air Conditioners', 'Air Fresheners', 'Air Purifiers', 'Amplifiers & Effects', 'Breastfeeding
==============================================
==============================================
cluster 14:
['Basic Supplies', 'Bathroom', 'Bathroom Accessories', 'Bathroom Furniture Sets', 'Bathroom Sa
==============================================
==============================================
cluster 15:
['Above Knee, Mini', 'Backpack', 'Backpack Style', 'Baggy, Loose', 'Boot Cut', 'Box', 'Button'
==============================================
==============================================
cluster 16:
['Advertisement', 'Animation', 'Artwork', 'Biographies & Memoirs', 'Biography', 'Book', 'Books
==============================================
==============================================
cluster 17:
['Bedspreads & Coverlets', 'Dehumidifiers', 'Epilators', 'Humidifiers', 'Needlecraft', 'Paperma
==============================================
==============================================
cluster 18:
['Animal', 'Animals', 'Art Doll', 'Baby & Toddler Toys', 'Building Toys', 'Child Friendly', 'C
```

```
==============================================
==============================================
cluster 19:
['Area Rugs & Pads', 'Backpacks, Bags & Briefcases', 'Bags & Cases', 'Bags and Purses', 'Bath 
==============================================
==============================================
```