prod_categorization

August 19, 2019

1 Imports

In [1]:
```
import json
import re
from pprint import pprint
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.stem import SnowballStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.cluster import KMeansClusterer
import numpy as np
from collections import Counter
from sklearn import preprocessing
import random
```

2 Read input file and sanity check

In [2]:
```
random.seed(0)


def read_file(file_name):

    with open(file_name, encoding='utf-8') as data_file:
        data = json.loads(data_file.read())

    pprint(data[0])

    return data
```

3 Preprocess Input data

In [3]:
```
''' Preprocess the text to remove html tags, special characters, stop words, lemmatiza
    Save the output to a json'''
def preprocess(data):
```

```
    stopword = set(stopwords.words('english'))
    wordnet_lemmatizer = WordNetLemmatizer()
    snowball_stemmer = SnowballStemmer('english')
    stopword = list(stopword) +  ['entire', 'collection', 'selection','select', 'free'
                                  'jcrew', 'urban', 'outfitter', 'shopify', 'long', 'lo
                                  'fit', 'loose','item', 'import', 'shop', 'jcrewcom',
                                  'price', 'yet', 'youll', 'would', 'could', 'look', 'c
                                  'tall', 'short', 'petite', 'men', 'size', 'made', 'st
                                  'all', 'woman', 'man', 'kid', 'girl', 'boy', 'length'
                                  'fit', 'dry', 'clean', 'back', 'body', 'one', 'knit',
                                  'special', 'charge', 'sale', 'dont', 'color']

    corpus = []
    for cnt, element in enumerate(data):

            cleaned_des = re.sub('<[^<]+?>','', element['description'])
            des = re.sub('[^a-z\-\s]+', '',cleaned_des.lower())
            des = re.sub('[\-]', ' ',des)
            word_tokens = nltk.word_tokenize(des)
            lemmatized_word = [wordnet_lemmatizer.lemmatize(word) for word in word_toke
            #stemmed_word = [snowball_stemmer.stem(word) for word in lemmatized_word]
            removing_stopwords = [word for word in lemmatized_word if word not in set(s
            element['description'] = [word for word in removing_stopwords if len(word)

            element['id'] = cnt
            element['Category'] = 'Other'
            if len(element['description']) > 1:
                corpus.append(' '.join(element['description']))

    with open('prepr_data.json', 'w', encoding='utf-8') as f:
        json.dump(data, f, ensure_ascii=False, indent=4)
    print('Preprocessed data output at prepr_data.json ')

    return corpus
```

4 Tf-idf vec generation

```
In [4]: def tfidf_vectorizer(corpus, max_df = 0.80, min_df = 0.002):

    vectorizer = TfidfVectorizer(max_df=max_df, min_df = min_df)
    X = vectorizer.fit_transform(corpus)
    print(X.shape)
    #print(vectorizer.get_feature_names())

    return X
```

5 Kmeans with cosine distance similarity

```
In [5]: def kmeans(X, num_centers = 8):

            kclusterer = KMeansClusterer(num_means = num_centers, distance=nltk.cluster.util.co
                                         repeats= 7, avoid_empty_clusters=True)

            assigned_clusters = kclusterer.cluster(np.asarray(X.todense()), assign_clusters=Tru

            return assigned_clusters
```

Post-process and get top words in cluster

```
In [6]: def post_process(data, assigned_clusters):

            track_corpus = 0
            corpus_set = {}

            for element in data:
                if len(element['description']) > 1:
                    element['Category'] = assigned_clusters[track_corpus]
                    corpus_set.setdefault(assigned_clusters[track_corpus],[])
                    corpus_set[assigned_clusters[track_corpus]].extend(element['description'])
                    track_corpus = track_corpus + 1

            print("Unnamed categories in initial_res_data.json\n\n")

            with open('initial_res_data.json', 'w', encoding='utf-8') as f:
                json.dump(data, f, ensure_ascii=False, indent=4)


            for key,val in corpus_set.items():
                cnt = Counter(val)
                print(key)
                print(cnt.most_common(15))
```

6 Save first stage clustering results

```
In [32]: def save_first_stage_res(data, catg):

             for element in data:
                 if len(element['description']) > 1:
                     key = int(element['Category'])
                     element['first_stage'] = catg[key]

             with open('first_stage_res.json', 'w', encoding='utf-8') as f:
                 json.dump(data, f, ensure_ascii=False, indent=4)
```

```
        print("Saved first stage clustering results in first_stage_res.json ")
```

7 Main program

```
In [28]: file_name = 'product_data.json'
         max_df = 0.8
         min_df = 0.003
         data = read_file(file_name)
         corpus = preprocess(data)
         Xvect = tfidf_vectorizer(corpus, max_df = max_df, min_df = min_df)
```

```
{'description': 'Supersoft speckled French terry makes this (tush covering!) '
                "turtleneck-sweatshirt hybrid the layering piece you'll want "
                'to wear to the gym, to lunch, to, well, everywhere this '
                'winter. Loose fit. Body length: 27 1/2. Cotton. Import.',
 'images_url': 'https://www.jcrew.com/s7-img-facade/H3588_PK6317_m?fmt=jpeg&qlt=90,0&resMode=sh
Preprocessed data output at prepr_data.json
(949, 1120)
```

```
In [29]: # num_centers were tried from 6-11 and set to 8 (max_categories in product list given
         num_centers = 8
         assigned_clusters = kmeans(Xvect,num_centers)
         #Save assigned clusters to text file
         np.savetxt("initial_kmeans.out",assigned_clusters,fmt = '%u', delimiter=',')
```

```
/Users/ramya/miniconda3/lib/python3.6/site-packages/nltk/cluster/util.py:133: RuntimeWarning: i
  sqrt(numpy.dot(u, u)) * sqrt(numpy.dot(v, v))))
```

```
In [30]: clusters = np.loadtxt("initial_kmeans.out")
         post_process(data, clusters)
```

```
Unnamed categories in initial_res_data.json
```

```
5.0
[('sweater', 22), ('sweatshirt', 15), ('terry', 14), ('french', 13), ('fleece', 11), ('crewneck
0.0
[('shirt', 207), ('button', 55), ('sleeve', 35), ('top', 30), ('perfect', 27), ('slim', 20), (
1.0
[('swimwear', 67), ('top', 64), ('bikini', 39), ('stripe', 28), ('tie', 27), ('bottom', 18), (
6.0
[('dress', 122), ('skirt', 17), ('tie', 16), ('perfect', 14), ('floral', 14), ('sheath', 14),
7.0
```

```
[('rise', 74), ('hand', 72), ('leg', 62), ('oil', 62), ('soft', 45), ('care', 44), ('detail', 4
4.0
[('pant', 74), ('jacket', 43), ('und', 36), ('pour', 35), ('avec', 33), ('contenu', 25), ('soi
2.0
[('earring', 45), ('necklace', 20), ('jewelry', 18), ('fine', 17), ('gold', 17), ('plated', 17
3.0
[('leather', 41), ('bag', 30), ('accessory', 20), ('heel', 15), ('sole', 14), ('upper', 13), (
```

```
In [34]: catg_names = {5: "Tops", 0: "Tops", 1 : "Swimwear, Intimates, Others",
                       6: "Dresses, Skirts, Others", 7: "Unnamed, Others",
                       4: "Pant, Jacket, Others", 2: "Jewellery", 3: "Bags, Shoes"}
         save_first_stage_res(data, catg_names)
```

8 Attempt second stage clustering

```
In [62]: ''' Obtain Sub-corpus of given category'''

         def get_subcorpus(data, catg):

             corpus = []
             for element in data:
                 if (len(element['description']) > 1 and int(element['Category']) == catg):
                     corpus.append(' '.join(element['description']))

             return corpus
```

```
In [63]: ''' Post process second stage clustering results'''

         def pp_sec_stage(data, catg, assigned_clusters):

             track_corpus = 0
             corpus_set = {}

             for element in data:
                 if (len(element['description']) > 1 and int(element['Category']) == catg):
                     element['Category_Sec'] = assigned_clusters[track_corpus]
                     corpus_set.setdefault(assigned_clusters[track_corpus],[])
                     corpus_set[assigned_clusters[track_corpus]].extend(element['description'])
                     track_corpus = track_corpus + 1

             for key,val in corpus_set.items():
                 cnt = Counter(val)
                 print(key)
                 print(cnt.most_common(15))
```

```
In [64]: ''' Update data based on second stage clustering results'''
```

```
        def update_data(data, catg, catg_names):

            for element in data:
                if (len(element['description']) > 1 and int(element['Category']) == catg):
                    key = int(element['Category_Sec'])
                    element['sec_stage'] = catg_names[key]
```

```
In [65]: ''' Save second stage clustering results '''

        def save_sec_stage_res(data):

            with open('final_res.json', 'w', encoding='utf-8') as f:
                json.dump(data, f, ensure_ascii=False, indent=4)

            print("Saved second stage clustering results in final_res.json ")
```

```
In [66]: with open('first_stage_res.json', encoding='utf-8') as data_file:
                data = json.loads(data_file.read())
```

9 Attempt second stage for Catgeory 1

```
In [79]: max_df = 0.95
         min_df = 0.001
         catg = 1
         num_centers = 4
         sub_corpus = get_subcorpus(data, catg )
         sub_Xvect = tfidf_vectorizer(sub_corpus, max_df = max_df, min_df = min_df)
         assigned_clusters = kmeans(sub_Xvect,num_centers)
         #Save assigned clusters to text file
         out_filename = "subkmeans_catg_" + str(catg)
         np.savetxt(out_filename,assigned_clusters,fmt = '%u', delimiter=',')
```

```
(138, 350)
```

```
In [124]: catg = 1
          out_filename = "subkmeans_catg_" + str(catg)
          clusters = np.loadtxt(out_filename)
          pp_sec_stage(data, catg, clusters)
```

```
1.0
[('top', 34), ('tie', 17), ('tank', 14), ('perfect', 7), ('front', 6), ('active', 6), ('pretty
0.0
[('swimwear', 28), ('piece', 14), ('swimsuit', 14), ('stripe', 10), ('print', 7), ('inch', 7),
2.0
[('bikini', 39), ('swimwear', 36), ('top', 23), ('bottom', 18), ('stripe', 9), ('playa', 7), (
```

```
3.0
[('pajama', 9), ('set', 8), ('sleeve', 8), ('top', 7), ('stripe', 6), ('print', 6), ('button',
```

```
In [125]: catg_names = {1: "Tops", 3: "Others", 0 : "Swimwear", 2 : "Swimwear"}
          update_data(data, catg, catg_names)
```

10 Attempt second stage for Catgeory 6

```
In [89]: max_df = 0.95
         min_df = 0.001
         catg = 6
         num_centers = 3
         sub_corpus = get_subcorpus(data, catg )
         sub_Xvect = tfidf_vectorizer(sub_corpus, max_df = max_df, min_df = min_df)
         assigned_clusters = kmeans(sub_Xvect,num_centers)
         #Save assigned clusters to text file
         out_filename = "subkmeans_catg_" + str(catg)
         np.savetxt(out_filename,assigned_clusters,fmt = '%u', delimiter=',')
```

```
(89, 342)
```

```
In [90]: clusters = np.loadtxt(out_filename)
         pp_sec_stage(data, catg, clusters)
```

```
1.0
[('dress', 101), ('floral', 12), ('sleeve', 12), ('sheath', 11), ('mercantile', 10), ('wrap', 9
2.0
[('dress', 20), ('tie', 15), ('perfect', 10), ('waist', 9), ('shoulder', 8), ('linen', 7), ('ru
0.0
[('skirt', 16), ('sandal', 6), ('ruffle', 3), ('tiered', 2), ('libertyreg', 2), ('floral', 2),
```

```
In [91]: catg_names = {0: "Skirts", 1: "Dresses", 2: "Dresses"}
         update_data(data, catg, catg_names)
```

11 Attempt second stage for Catgeory 7

```
In [105]: max_df = 0.95
          min_df = 0.005
          catg = 7
          num_centers = 3
          sub_corpus = get_subcorpus(data, catg )
          sub_Xvect = tfidf_vectorizer(sub_corpus, max_df = max_df, min_df = min_df)
          assigned_clusters = kmeans(sub_Xvect,num_centers)
          #Save assigned clusters to text file
          out_filename = "subkmeans_catg_" + str(catg)
          np.savetxt(out_filename,assigned_clusters,fmt = '%u', delimiter=',')
```

```
(229, 1121)
```

```
In [106]: clusters = np.loadtxt(out_filename)
          pp_sec_stage(data, catg, clusters)
```

```
1.0
[('rise', 71), ('leg', 60), ('front', 41), ('detail', 34), ('knee', 23), ('inseam', 22), ('slir
2.0
[('oil', 62), ('extract', 41), ('hand', 40), ('skin', 36), ('acid', 22), ('soft', 22), ('ingred
0.0
[('care', 20), ('album', 17), ('hand', 17), ('mother', 16), ('piece', 14), ('tee', 14), ('model
```

```
In [107]: catg_names = {1: "Pant", 0: "Others", 2: "Others"}
          update_data(data, catg, catg_names)
```

12 Attempt second stage for Catgeory 4

```
In [112]: max_df = 0.95
          min_df = 0.01
          catg = 4
          num_centers = 3
          sub_corpus = get_subcorpus(data, catg )
          sub_Xvect = tfidf_vectorizer(sub_corpus, max_df = max_df, min_df = min_df)
          assigned_clusters = kmeans(sub_Xvect,num_centers)
          #Save assigned clusters to text file
          out_filename = "subkmeans_catg_" + str(catg)
          np.savetxt(out_filename,assigned_clusters,fmt = '%u', delimiter=',')
```

```
(173, 350)
```

```
In [113]: clusters = np.loadtxt(out_filename)
          pp_sec_stage(data, catg, clusters)
```

```
2.0
[('pour', 35), ('avec', 33), ('contenu', 25), ('soins', 24), ('taille', 23), ('vous', 18), ('un
1.0
[('pant', 74), ('und', 36), ('mit', 20), ('stretch', 18), ('inhalt', 17), ('pflege', 17), ('foo
0.0
[('jacket', 43), ('field', 8), ('downtown', 7), ('coat', 6), ('tie', 5), ('outerwear', 4), ('wa
```

```
In [114]: catg_names = {1: "Pant/Some jeans", 0: "Others/Jacket", 2: "Others"}
          update_data(data, catg, catg_names)
```

13 Attempt second stage for Catgeory 3

```
In [120]: max_df = 0.99
          min_df = 0.0001
          catg = 3
          num_centers = 3
          sub_corpus = get_subcorpus(data, catg )
          sub_Xvect = tfidf_vectorizer(sub_corpus, max_df = max_df, min_df = min_df)
          assigned_clusters = kmeans(sub_Xvect,num_centers)
          #Save assigned clusters to text file
          out_filename = "subkmeans_catg_" + str(catg)
          np.savetxt(out_filename,assigned_clusters,fmt = '%u', delimiter=',')
```

```
(71, 282)
```

```
In [121]: clusters = np.loadtxt(out_filename)
          pp_sec_stage(data, catg, clusters)
```

```
0.0
[('accessory', 19), ('bag', 14), ('tote', 8), ('straw', 6), ('striped', 4), ('market', 3), ('c
2.0
[('leather', 32), ('heel', 15), ('sole', 14), ('upper', 13), ('lining', 13), ('synthetic', 9),
1.0
[('bag', 16), ('leather', 9), ('pouch', 6), ('italian', 6), ('case', 3), ('mini', 3), ('bucket
```

```
In [122]: catg_names = {1: "Bags", 0: "Bags", 2: "Shoes"}
          update_data(data, catg, catg_names)
```

```
In [126]: save_sec_stage_res(data)
```

```
Saved second stage clustering results in final_res.json
```