基於商品內容的推薦系統

- 目的：適用於購物網站
- 介紹：推薦系統大致分三類，基于内容的推薦系統，協同過濾推薦系統和混合推薦系統（使用前兩者方式的組合）。基於協同過濾的推薦系統使用的是用戶的行為數據資料。

  基於內容的推薦系統則可以很好的規避不同用戶或環境下資料不同，皆會有結果差異的問題。基於內容的推薦系統，它使用商品的數據(如商品的名稱)來為用户進行推薦，忽略用戶行為(例如商品評價數據、購買類別、用戶基本資料....等) ，而忽略了用戶行為(不使用評分數據),所以它可以有效的規避"冷啟動"問題。
- 作法：

  ✧ **處理數據**

  - 數據資料的準備：適用於[購物網站商品資料] products，
  - Products 資料的斷詞：採用 jieba 庫(分詞經過優化適用於購物網站 jieba_split.py)，將所有的商品名稱分詞，常時維護及淨洗 CleaningData (dict.txt 字典集、stopwords.txt 停用詞)。
  - Products 資料的分類：若 Products 資料未經過分類例如[手機/智慧穿戴/智慧型手機]、[生活家電/冰箱/冷凍櫃] ...，由來源資料 spider 時就要分好類別，若商品無分類其產生的結果

分歧很多。(可基於向量演算透過機器學習 word2vec 分類或訓練 MultinomialNB() 樸素貝葉斯演算法、訓練 LinearSVC() 線性演算法來分類，常時 CleaningData 樣本資料集)，商品分類最好是較為明確的類例如:[手機/智慧穿戴/智慧型手機]

✧ 套件使用

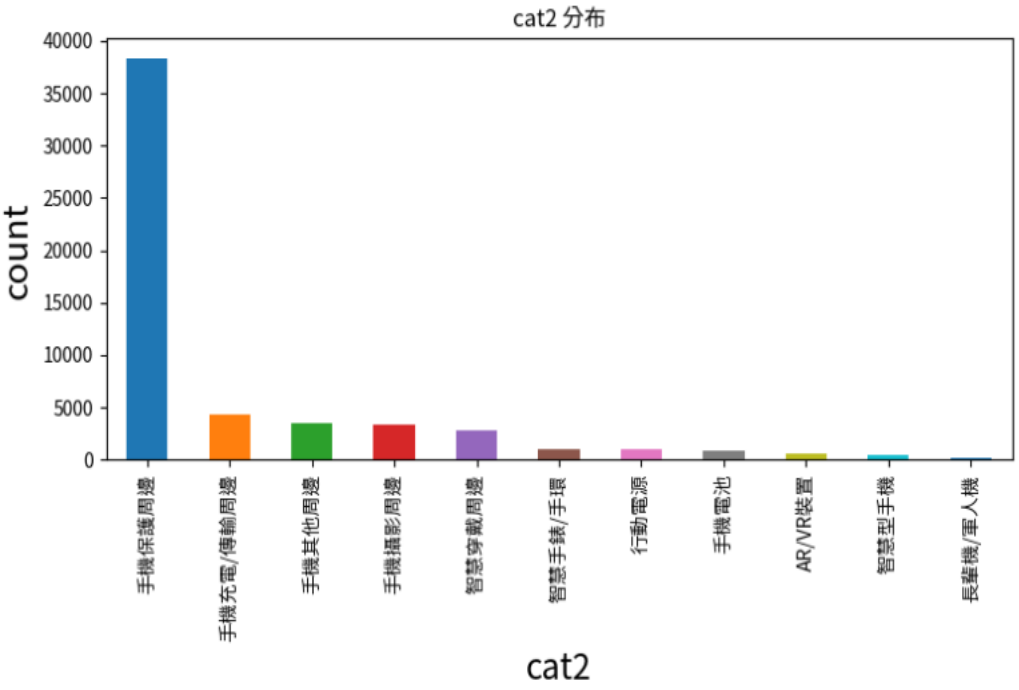
■ 套件 import

```
import pandas as pd
import matplotlib
import numpy as np
import matplotlib.pyplot as plt

import jieba as jb    #斷詞 jieba.cut 一般斷詞使用

import jieba_split as jieba_comm    #自訂斷詞優化

import re

#生成 TF-IDF 詞向量套件

from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation


import goodsProc as goodsProc_comm #分詞


plt.rcParams['axes.unicode_minus'] = False

#  將字體換成 fontlist-v300.json.name

plt.rcParams['font.sans-serif'] = ['Noto Sans TC']

#  修復負號顯示問題

 plt.rcParams['axes.unicode_minus']=False
```

✧ 數據分析

■ 檢視數據分佈：(商品分類數、商品分詞分佈數)，來決定是否 CleaningData 做出來的預測模型才會精準。

例如以下案例：



商品數目：56,442 (有分類的品項)

g_name 數量: 2,738 (智慧型手表\手環)的品項數量

商品名稱分詞數據：

name 平均詞語數 7.147187728268809

name 最少詞語數 2

name 最多詞語數 23

- 代碼程式

```
    # 數據分析
  def plt2cat2lab(self):
        d = {'cat2':
self.products_new['cat2'].value_counts().index, 'count':
self.products_new['cat2'].value_counts()}
        df_cat1 =
pd.DataFrame(data=d).reset_index(drop=True)
        df_cat1.plot(x='cat2', y='count', kind='bar',
legend=False, figsize=(8, 5))
        plt.title("cat2 分布")
        plt.ylabel('count', fontsize=18)
        plt.xlabel('cat2', fontsize=18)
        plt.show()
    # 數據分析
    def data2cl(self):
        self.smart_wear['word_count'] =
self.smart_wear['cut_name'].apply(lambda x:
len(str(x).split()))
```

```
name_lengths = list(self.smart_wear['word_count'])
print("g_name 分詞數量:", len(name_lengths),
                        "\nname 平均詞語數",
np.average(name_lengths),"\nname 最少詞語數",
min(name_lengths),
                        "\nname 最多詞語數",
max(name_lengths))
```

✧ 推薦模型

- 對已經過分詞(斷詞)的 cut_name 字段進行向量化處理,使用 sklearn 的 TfidfVectorizer 來對欲推薦品項名稱進行向量化處裡。要注意過濾掉一些無意義的詞
- 代碼程式：

```
def recommendations(self,text, num):
        #ss = processText(text)
        ss_str = goodsProc_comm.set2jieba(text.strip(),
stopwords)
        ss=[]
        for list_k in ss_str.split(','):
            ss.append(list_k)
```

```
            ss_vec = self.tfidf.transform(ss)
            cos_sim = cosine_similarity(ss_vec, self.tfidf_vec)
            arr = cos_sim[0]
            idxs = list(np.argsort(-arr)[:num])
            print("g_name: [ %s ] ,斷詞: [ %s ]" %
(str(text),str(ss)))
            for idx in idxs:
                row = self.smart_wear[self.smart_wear.index ==
idx]
                product_su = row.su.values[0]
                g_name = row.g_name.values[0]
                print("su:", product_su, ",", g_name)
        #推薦模型
        def recgoods(self):
            self.tfidf = TfidfVectorizer(analyzer='word',
ngram_range=(1, 2),
min_df=0).fit(self.smart_wear['cut_name'])
            self.tfidf_vec =
self.tfidf.transform(self.smart_wear['cut_name'])
```

self.recommendations("小米手環 3 專用 充電線(副廠)",10)

結果：su:商品 pk 或 id;產生 10 組推薦商品

g_name: [小米手環 3 專用 充電線(副廠)] ,斷詞: [['手環', '小米', '充電線']]

su: gud6an9 , MR 小米手環 3/4 通用運動矽膠替換錶帶(迷彩淺綠)

su: a4a2yr3 , MR 小米手環 3/4 通用不鏽鋼三珠摺疊扣錶帶(鏡面黑)

su: 9khq5ew , 【MR】小米手環 5 單色運動防水矽膠替換錶帶

su: ak324kd , 【MR】小米手環 3/4 通用金屬編織卡扣式錶帶

su: 84kr3i7 , 兩款任選 小米手環替換錶帶 金屬 皮革任選

su: endsxdn , 小米手環 4 威尼斯精鋼 3 珠錶帶

su: qemfniq , 米布斯 小米手環 3/4/5/6 代矽膠撞色 X 錶帶 小米智能手環通用替換錶帶 免工具安裝

su: 62i9aqh , 小米手環 5 單色 高質感運動替換錶帶 優質的矽膠材質(小米手環 5 錶帶)

su: kvhktdq , 小米手環 3 小米手環 4 金屬腕帶 不銹鋼材質 小米手環 3 小米手環 4 腕帶 小米手環三代四代金屬腕帶

su: 7e8eekf , 【MR】小米手環 3/4 通用運動矽膠替換錶帶(迷彩淺綠)

- 訓練模型並保存

```
#保存訓練模型
def model_tfmark(self):
    # max_df / min_df: [0.0, 1.0]
    # 內浮點數或正整數，默認值 = 1.0
    # 當設置為浮點數時，過濾出現在超過 max_df / 低於 min_df 比例的句子中的詞語；正整數時，則是超過 max_df 句句子。
    # 這樣就可以幫助我們過濾掉出現太多的無意義詞語

    #ngram_range=(1, 2):允許詞表使用 1 個詞語，或者 2 個詞語的組合
    #max_features=10 制最多使用 10 個詞語
    #max_feature: int 限制最多使用多少個詞語，模型會優先選取詞頻高的詞語留下
    #stop_words=["是", "的"] 停用詞;這裡不用因為在匯出商品品項時就已經處裡了
    #生成訓練的 tfidf 矩陣，矩陣每行代表一個樣本的 tfidf 向量。
    #ex: , max_df=0.8,min_df=2
    self.tfidf = TfidfVectorizer(token_pattern=r"(?u)\b\w+\b",analyzer='word',ngram_range=(1, 2),min_df=3).fit(self.smart_wear['cut_name'])
    self.tfidf_vec = self.tfidf.transform(self.smart_wear['cut_name'])


    sparse.save_npz('C:/MyProject/python/webspider/webspider/model/train_tfidf.npz', self.tfidf_vec)  # 保存
    # 保存 tfidf 模型
    joblib.dump(self.tfidf, 'C:/MyProject/python/webspider/webspider/model/model_tfidf.pkl')
    # 保存 item_id 列表
    joblib.dump(self.smart_wear['su'],
```

```
    'C:/MyProject/python/webspider/webspider/model/train_item
    _id.pkl')
```

- 載入模型使用

```
# 載入訓練檔==(使用已訓練好的 model)
self.modpatch ='C:/MyProject/python/webspider/webspider/'
self.index = joblib.load(self.modpatch
+'model/train_item_id.pkl')
self.tfidf = joblib.load(self.modpatch
+'model/model_tfidf.pkl')
self.tfidf_vec = sparse.load_npz(self.modpatch
+'model/train_tfidf.npz')
```

- 計算 Python:余弦相似度 m * n 矩陣

```
#model
ss_vec = self.tfidf.transform(ss)
cos_sim = cosine_similarity(ss_vec, self.tfidf_vec)
```