关键词搜索

Kaggle竞赛题：[https://www.kaggle.com/c/home-depot-product-search-relevance (https://www.kaggle.com/c/home-depot-product-search-relevance)](https://www.kaggle.com/c/home-depot-product-search-relevance)

鉴于课件里已经完整的show了NLTK在各个NLP处理上的用法，我这里就不再重复使用了。

本篇的教程里会尽量用点不一样的库，让大家感受一下Python NLP领域各个库的优缺点。

Step1：导入所需 ¶

所有要用到的库

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestRegressor, BaggingRegressor
from nltk.stem.snowball import SnowballStemmer
```

读入训练/测试集

In [2]:

```
df_train = pd.read_csv('../input/train.csv', encoding="ISO-8859-1")
df_test = pd.read_csv('../input/test.csv', encoding="ISO-8859-1")
```

这里还有个有用的玩意儿，叫产品介绍

In [3]:

```
df_desc = pd.read_csv('../input/product_descriptions.csv')
```

看看数据们都长什么样子

In [5]:

```
df_train.head()
```

Out[5]:

	id	product_uid	product_title	search_term	relevance
0	2	100001	Simpson Strong-Tie 12-Gauge Angle	angle bracket	3.00
1	3	100001	Simpson Strong-Tie 12-Gauge Angle	l bracket	2.50
2	9	100002	BEHR Premium Textured DeckOver 1-gal. #SC-141 ...	deck over	3.00
3	16	100005	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	rain shower head	2.33
4	17	100005	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	shower only faucet	2.67

In [6]:

```
df_desc.head()
```

Out[6]:

	product_uid	product_description
0	100001	Not only do angles make joints stronger, they ...
1	100002	BEHR Premium Textured DECKOVER is an innovativ...
2	100003	Classic architecture meets contemporary design...
3	100004	The Grape Solar 265-Watt Polycrystalline PV So...
4	100005	Update your bathroom with the Delta Vero Singl...

看来不要做太多的复杂处理，我们于是直接合并测试/训练集，以便于统一做进一步的文本预处理

In [7]:

```
df_all = pd.concat((df_train, df_test), axis=0, ignore_index=True)
```

In [9]:

```
df_all.head()
```

Out[9]:

	id	product_title	product_uid	relevance	search_term
0	2	Simpson Strong-Tie 12-Gauge Angle	100001	3.00	angle bracket
1	3	Simpson Strong-Tie 12-Gauge Angle	100001	2.50	l bracket
2	9	BEHR Premium Textured DeckOver 1-gal. #SC-141 ...	100002	3.00	deck over
3	16	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	100005	2.33	rain shower head
4	17	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	100005	2.67	shower only faucet

合并之后我们得到:

In [11]:

```
df_all.shape
```

Out[11]:

```
(240760, 5)
```

产品介绍也是一个极有用的信息，我们把它拿过来：

In [13]:

```
df_all = pd.merge(df_all, df_desc, how='left', on='product_uid')
```

In [14]:

```
df_all.head()
```

Out[14]:

	id	product_title	product_uid	relevance	search_term	product_description
0	2	Simpson Strong-Tie 12-Gauge Angle	100001	3.00	angle bracket	Not only do angles make joints stronger, they ...
1	3	Simpson Strong-Tie 12-Gauge Angle	100001	2.50	l bracket	Not only do angles make joints stronger, they ...
2	9	BEHR Premium Textured DeckOver 1-gal. #SC-141 ...	100002	3.00	deck over	BEHR Premium Textured DECKOVER is an innovativ...
3	16	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	100005	2.33	rain shower head	Update your bathroom with the Delta Vero Singl...
4	17	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	100005	2.67	shower only faucet	Update your bathroom with the Delta Vero Singl...

好了，现在我们得到一个全体的数据大表格

Step 2: 文本预处理

我们这里遇到的文本预处理比较简单，因为最主要的就是看关键词是否会被包含。

所以我们统一化我们的文本内容，以达到任何term在我们的数据集中只有一种表达式的效果。

我们这里用简单的Stem做个例子：

（有兴趣的同学可以选用各种你觉得靠谱的预处理方式：去掉停止词，纠正拼写，去掉数字，去掉各种emoji，等等）

In [15]:

```
stemmer = SnowballStemmer('english')

def str_stemmer(s):
    return " ".join([stemmer.stem(word) for word in s.lower().split()])
```

为了计算『关键词』的有效性，我们可以naive地直接看『出现了多少次』

In [16]:

```
def str_common_word(str1, str2):
    return sum(int(str2.find(word)>=0) for word in str1.split())
```

接下来，把每一个column都跑一遍，以清洁所有的文本内容

In [18]:

```
df_all['search_term'] = df_all['search_term'].map(lambda x:str_stemmer(x))
```

In [19]:

```
df_all['product_title'] = df_all['product_title'].map(lambda x:str_stemmer(x))
```

In [20]:

```
df_all['product_description'] = df_all['product_description'].map(lambda x:str_stemm
er(x))
```

Step 3: 自制文本特征

一般属于一种脑洞大开的过程，想到什么可以加什么。

当然，特征也不是越丰富越好，稍微靠谱点是肯定的。

关键词的长度：

In [21]:

```
df_all['len_of_query'] = df_all['search_term'].map(lambda x:len(x.split())).astype(n
p.int64)
```

标题中有多少关键词重合

In [25]:

```
df_all['commons_in_title'] = df_all.apply(lambda
x:str_common_word(x['search_term'],x['product_title']), axis=1)
```

描述中有多少关键词重合

In [26]:

```
df_all['commons_in_desc'] = df_all.apply(lambda
x:str_common_word(x['search_term'],x['product_description']), axis=1)
```

等等等等。。变着法子想出些数字能代表的features，一股脑放进来~

搞完之后，我们把不能被『机器学习模型』处理的column给drop掉

In [27]:

```
df_all = df_all.drop(['search_term','product_title','product_description'],axis=1)
```

Step 4: 重塑训练/测试集

舒淇说得好，要把之前脱下的衣服再一件件穿回来

数据处理也是如此，搞完一圈预处理之后，我们让数据重回原本的样貌

分开训练和测试集

In [28]:

```
df_train = df_all.loc[df_train.index]
df_test = df_all.loc[df_test.index]
```

记录下测试集的**id**

留着上传的时候 能对的上号

In [29]:

```
test_ids = df_test['id']
```

分离出**y_train**

In [30]:

```
y_train = df_train['relevance'].values
```

把原集中的**label**给删去

否则就是cheating了

In [31]:

```
X_train = df_train.drop(['id','relevance'],axis=1).values
X_test = df_test.drop(['id','relevance'],axis=1).values
```

Step 5: 建立模型

我们用个最简单的模型：Ridge回归模型

In [32]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
```

用CV结果保证公正客观性；并调试不同的alpha值

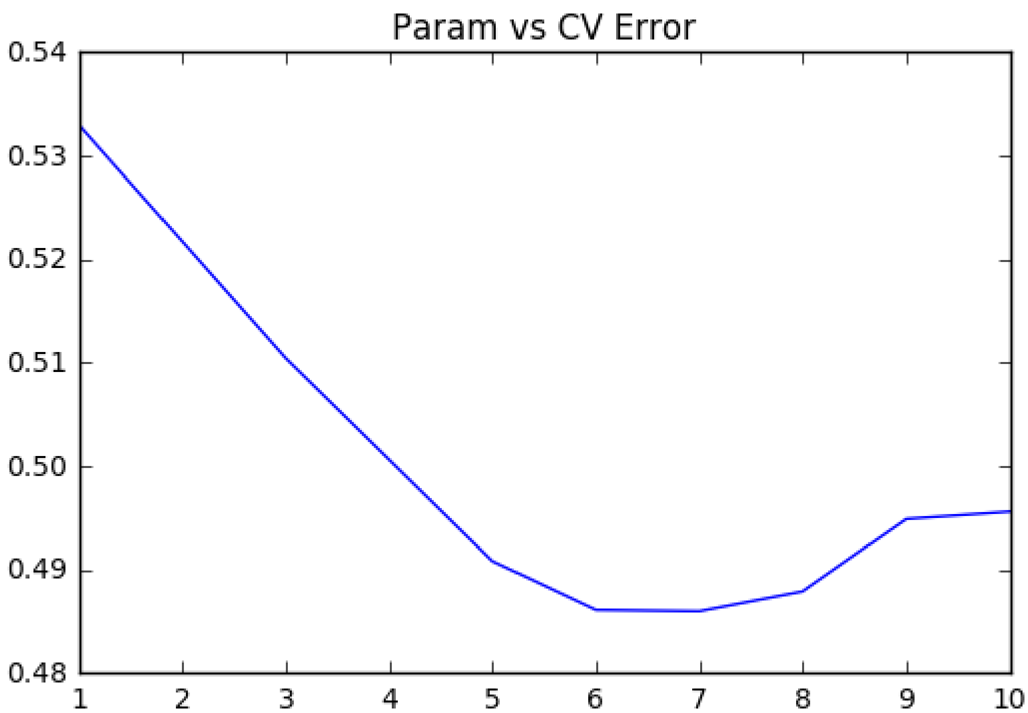
In [38]:

```
params = [1,3,5,6,7,8,9,10]
test_scores = []
for param in params:
    clf = RandomForestRegressor(n_estimators=30, max_depth=param)
    test_score = np.sqrt(-cross_val_score(clf, X_train, y_train, cv=5, scoring='neg_
mean_squared_error'))
    test_scores.append(np.mean(test_score))
```

画个图来看看：

In [39]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(params, test_scores)
plt.title("Param vs CV Error");
```



大概6~7的时候达到了最优解