朴素贝叶斯分类器

本次实验是创建一个朴素贝叶斯分类器，并测试其在数据集上的效果,数据集来自 http://qwone.com/~jason/20Newsgroups/ 上的20个分类的文章

```
hw2/classification.py    分类器
hw2/gen_tarin_data.py    生成训练数据、测试数据、标答
hw2/valuation.py         对分类器输出的结果与标答作比较输出准确率
hw2/train_data.json      训练数据
hw2/test_data.json       测试数据
hw2/correct_test_data2.json 标准答案
hw2/answer.json          分类器输出的答案
```

1 准备

用spacy分词，spacy一个python的分词库，“最快的分词器”（自己说的），官网链接：https://spacy.io/

按照文档说明安装spacy以及英文分词的model。

系统：win10专业版 (mibook air i5 6200U)

2 实验步骤

1> 推公式

假设我们有m个文档的分类（这个数据集是20个）$(v_1,v_2,v_3,\cdots,v_m)$

同时有n个词 ($term$)，这些词的范围是我们自己确定的，可以作为分类依据的词，也可以是m个文档的全部单词。$(x_1,x_2,x_3,\cdots,x_n)$

根据我们的训练数据，能够得一个分类里的所以词的概率，即当确定分类为 v_j 时，x_i 的概率。用 $P(x_i | v_j)$ 表示

一个分类为 v_j 的概率，用 $P(v_j)$ 表示

一个词为 x_i 的概率，用 $P(x_i)$ 表示

假设测试集中的一篇文章，$<x_1,x_2,x_3,\cdots,x_k>$ 能够描述这篇文章 A。 $({x_k |x_k出现在文章A中})$ 根据朴素贝叶斯，这篇文章属于 v_j 分类的概率为 $P(v_j| {x_k |x_k出现在文章A中})$

$$ P(v_j|x_1,x_2,x_3,\cdots,x_k) = \displaystyle\frac{P(x_1,x_2,x_3,\cdots,x_k \cap v_j)}{P(x_1,x_2,x_3,\cdots,x_k)} =\displaystyle\frac{P(x_1,x_2,x_3,\cdots,x_k | v_j)*P(v_j)}{P(x_1,x_2,x_3,\cdots,x_k)} $$

取 v_j 使得 $max, P(v_j|x_1,x_2,x_3,\cdots,x_k)$，上式分母与 v_j 无关，使 $max, P(x_1,x_2,x_3,\cdots,x_k | v_j)*P(v_j)$ 即可

由条件独立性， $$ P(x_1,x_2,x_3,\cdots,x_k | v_j)*P(v_j) = P(v_j)*\displaystyle \prod^{x_1,x_2,x_3,\cdots,x_k} {x_i}P(x_i|v_j) $$ *由于连乘会增加复杂度并且降低精度。对等式取 log $$ log\Big(P(v_j)*\displaystyle*

\prod^{x_1,x_2,x_3,\cdots,x_k}
{x_i}P(x_i|v_j)\Big)=log\big(P(v_j)\big)+\displaystyle\sum^{x_1,x_2,x_3,\cdots,x_k}_{x_i}log(x_i|v_j) $$

2> spacy分词

spacy提供了简单易用的api，分词很容易，就是刚开始的时候处理的速度有点慢，google了一下，发现spacy的分词是有许多除了分词之外的 *pipiline*，可以在加载model的时候选择不加载，例如

```
spacy.load('en', disable=['parser','tagger','ner'])
```

相关文档：https://spacy.io/usage/processing-pipelines#disabling

还有,spacy可以使用 *pipe* 加速处理文本，原理大概是使用了多线程。

```
for doc in nlp.pipe(all_string_list):
    for i in doc:
        self._add_to_idf(i.lemma_)
        self._add_to_tf(i.lemma_,a)
```

相关文档：https://spacy.io/api/pipe

3> 其他

按照公式这样算就好了，检查一个词在不在词典里，用try..catch 抓异常比手动检查快。

对于文章*A*，如果不考虑单词的重复，每个词只考虑一次，重复的不考虑。分类器的正确率在 82% 左右。考虑重复，分类器的正确率可以达到 88%。以上两个数据在随机生成的训练数据占 *50%* ， 测试数据为剩余 *50%* 时计算所得。