文档分类综合报告

姓名：杨鑫　　**学号：**201814845
班级：学硕班　　**邮箱：**joeyangbuer@gmail.com

一．　算法思想：

1. KNN算法：

(1) 将文档集按照一定比例划分为训练集和测试集（保持测试集和训练集每个类的分布一致）；
(2) 对训练集和测试集进行预处理（转换大小写，分词，去除停用词，去除低频词语）；
(3) 用训练集预处理后的词语集建立词库，并统计每个文档中，这些词语出现的频率，建立文档-词频（tf）表，建立文档词频表的同时统计词库中的每个词语的文档频率（df）；
(4) 利用上诉统计的词频和文档频率按照以下三个公式计算每个词语的权重值；

$$tf(t,d) = \begin{cases} 1 + log\ c(t,d), if\ c(t,d) > 0 \\ 0, otherwise \end{cases}$$

$$idf(t) = \log(\frac{N}{df(t)})$$

$$w(t,d) = tf * idf$$

(5) 对词库中的每个单词按照上面的公式计算权重，并用这些词语的权重构成每个文档的向量表示；
(6) 按照第二步对测试集进行预处理，然后依据训练集建好的词库对测集的文档进行表示，具体单词权重的计算方法见第四步；
(7) 到此为止，训练集和测试集文档的向量表示都建立成功了，然后对测试集中每一个的文档向量，求训练集中距离最近的前k个文档，并将这k个文档中最多的一类作为这个测试文档的类别；
(8) 计算测试集的分类准确率。

2. Naïve Bayesian Classification算法

贝叶斯公式：

$$\mathrm{P}(c_i|\mathrm{d}) = \frac{P(d|c_i)P(c_i)}{P(d)}$$

c_i是文档类别，d是一篇文档，$P(d|c_i)$是似然，$P(c_i)$是先验概率，$\mathrm{P}(c_i|\mathrm{d})$是后验概率，表示文档被分为$c_i$的概率。

条件独立假设：文档中的每个词汇都是独立出现的，互不影响，即公式也可写成：

$$\mathrm{P}(c_i|\mathrm{d}) = \frac{P(x_1, x_2, \dots, x_m|c_i)P(c_i)}{P(x_1, x_2, \dots, x_m)}$$

$$= \frac{P(x_1|c_i)P(x_2|c_i)\dots P(x_m|c_i)P(c_i)}{P(x_1, x_2, \dots, x_m)}$$

x_m表示文档d的第m个词。

计算： 用贝叶斯公式计算的时候分母都一样，因此只用计算分子，统计相关类别单词出现的频数除以总的单词数就可以得到$P(x_m|c_i)$，每一个类别含有的文档数除以文档总数就可以得到$P(c_i)$，因此就可以算出文档d分到每一个类别的概率，最后选取概率最大的类别作为文档的类别。

3. 聚类算法

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers

评价指标为NMI

二． 项目结构：

代码一共由8个文件组成

1. dataSeg.py: 对每个类别进行均匀采样，将数据集按照8:2分为训练集和测试集，并将对应的文件路径存储到两个文件中；
2. docParse.py: 定义DocParser类，主要是对一个文档进行预处理

（1）全部转换成小写，

（2）利用正则表达式替换掉非字母字符和多余的空格，

（3）利用textbolb工具包统计文档的词频；

3. dicBulid.py: 定义DicBuilder类，主要是对文档集进行建立词典，类中主要维持三个数据结构（文档词频表，文档频率表，全局词频表），这三个数据结构都是字典类型，

（1） 文档词频表（每个文档中的每个单词出现的频率）结构为
{doc1:[(word1,frequency),(word2,frequency)...], ...}，

（2） 文档频率表（单词出现的文档数）的结构为
{word1: doc_frequency, ...}，

（3） 全局词频表（单词在训练集所有文档中中出现的次数和）
{word1: word_frequency,...}，

对文档集遍历处理完以后，同时也得到以上三个数据结构；

4. docVector.py: 该文件是主要是利用dicBulid.py中定义的DicBuilder类对dataSeg.py划分好的训练集和测试集进行处理。该类中有个vectorize方法，该方法可以根据上面三个数据结构计算文档的向量表示。具体地，遍历全局词频表，对每一个词频大于某个阈值（相当于过滤掉低频词语）的单词计算tf*idf，tf由（1）中的词频可以得到，idf可以由（2）得到，对一个文档而言，当全局词频表遍历完后，每个单词的权重也计算完毕，这时候一个文档的向量表示便得到了。循环所有文档，便可以得到所有文档的向量表示；得到两个数据集的向量表示后将其存储到csv文件中；
5. knn.py： 定义KNN类，该类主要实现两个方法：

（1）利用训练集向量表示以及相应的类别预测测试集类别（包含两种向量距离计算方式，余弦相似度和欧式距离），

（2）计算分类准确率；

6. knnClassifier.py: 加载docVector.py处理得到的文档向量表示，并利用knn.py中定义的KNN类对测试集做分类预测并计算准确率；
7. naiveBayesClassifier.py： 根据第(3)步计算出来的数据信息用朴素贝叶斯算法分类预测；
8. clustering.py：利用sklearn库里面的各种聚类算法对文档进行聚类。

三． 实验分析：

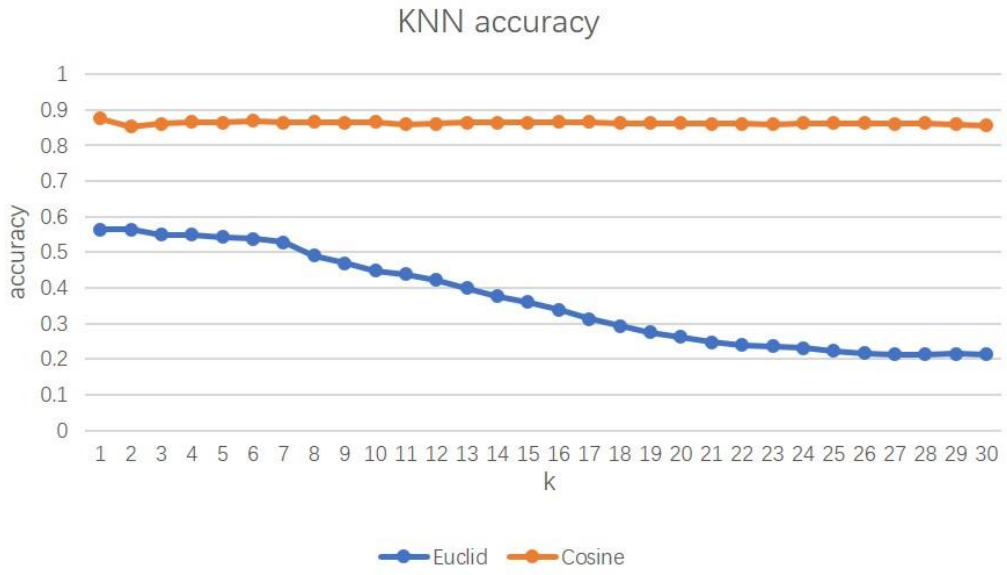
1. KNN算法

(1) 实验设置：

a. k选取1-30

b. 距离计算方式选取余弦相似度和欧式距离

(2) 实验结果：



(3) 实验总结:

a. 选取余弦相似度最为距离衡量方式的鲁棒性更强，算法性能更稳定，最终结果基本维持在86%左右，当k=1时，准确率最高，此时为87.6%，k=30时最低，此时为85.6% ；

b. 欧式距离受k值的影响更大，随着k增大，准确率逐渐下降，准确率由56.4%(k=1) 下降到 21.3%(k=30)。

2. Naïve Bayesian Classification算法思想

```
--start reading train/test data_preprocess path
  end up reading train/test data_preprocess path and the time cost is 0.055851s
--start building dic and word frequency statistics
  end up building dic and word frequency statistics and the time cost is 131.225167s
--start counting category scores
  end up counting category scores and the time cost is 6.663200s
accuracy = 0.873542
```

最后分类的准确率是87.35%

3. 聚类算法

Algorithm	Type	NMI	Algorithm	Type	NMI
K-means	~	0.8139	spectral_clustering	~	0.7713
AffinityPropagation	~	0.6313	GaussianMixture	spherical	0.7924
AgglomerativeClustering	ward	0.8065		diag	0.8031
	average	0.8907		tied	0.7402
	complete	0.7191		full	0.8044