实验报告

学号：201814849　　姓名：张延超　　班级：2018 级学硕

1、 实验任务

本次实验的任务有两个：（1）将 20news-18828 数据集中文档表示为向量空间模型（VSM)；（2）实现 KNN 算法，并将测试集文档分类，计算其准确率。

2、 实验步骤

1. 数据预处理
   首先从网站将文档下载到本地，并进行预处理操作。预处理的步骤主要有：
   （1）分词：本次实验中使用的分词工具是 TextBlob 分词工具，并利用 TextBlob 中的一些函数对文档进行预处理操作。
   （2）拼写检查：检查文档的句子中是否有拼写错误，该过程处理缓慢，谨慎使用。
   （3）大小写转化：在文档中，有大写字母、小写字母、数字以及特殊符号等字符，在实验中，将所有的大写字母转化为小写字母，并将特殊符号删除，如@,$,%等字符
   （4）复数转化为单数：名词单复数的存在会影响词表的建立，因此将名词的复数形式转化为单数形式，例如 dogs 变为 dog。
   （5）词形还原或者词干提取。考虑到时态、人称等问题的存在，在建立词表时，需要对单词进行词形还原或者词干提取，将词干提取出来作为最后的词表项。例如 went 变为 go，goes 变为 go 等
   （6）去除停用词。从网上下载常用的停用词表，将文档中的单词经过（1）-（5）的预处理之后，去除里面的停用词，这样的单词作为最终的词表项。
   （7）经过上述步骤，可以得到 3 个文件，分别为 word_frequency.npy,word_df.npy,word_tf.npy 这 3 个文件，这三个文件存储在 record.npz 中。
   （8）由于词表中存在一些低频词，而这些低频词对于文档分类任务起到的作用不大，甚至还有可能成为噪音，因此需要将低频词去除，在实验中设置的阈值是 15。过滤低频词之后，可以得到 3 个文件，分别为 word_frequency.npy,word_df.npy,word_tf.npy 这 3 个文件，这 3 个文件存储在 record_filter.npz 中。
   该步骤对应代码文件 preprocessing.py
2. 经过步骤 1，可以得到词表，单词对应的词频，文档频率以及在某个文档中的词项频率，根据这些数据，可以计算某一个单词 t 在某一个文档 d 中的 tf-idf。Tf-idf 的计算公式为

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

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

$$\mathrm{tf-idf} = \mathrm{tf(t,d)} * \mathrm{idf}$$

其中 c(t,d)为词项 t 在文档 d 中的词频，N 为总文档数。

根据上述计算公式，可以得到每一个单词在每一个文档中的 tf-idf，从而得到文档的向量空间模型（VSM）。每一个文档的向量空间模型存储为一个 npy 文件。

该步骤对应代码文件 document_vector.py。

3. 根据实验要求，将数据集划分为训练集和测试集 8:2，同时保证训练集和测试集独立同分布，并实现 kNN 算法。kNN（k-NearestNeighbor）分类算法是数据挖掘中比较简单的方法。kNN 算法的核心思想是如果一个样本在特征空间中的 k 个最相邻的样本中的大多数属于某一个类别，则该样本也属于这个类别，并具有这个类别上样本的特性。计算测试集中的样本点和训练集中的样本点之间的距离有两种方案：欧几里得距离和余弦相似度。公式分别为：

   欧几里得距离：$\mathrm{d(p,q)} = \sqrt{\sum_i (p_i - q_i}$

   余弦相似度：$cosine(d_i, d_j) = \frac{V_{d_i}^T V_{d_j}}{|V_{d_i}|_2 \times |V_{d_j}|_2}$

   该步骤对应代码文件 knn.py，data_input.py。

3、 实验结果

实验参数设置：

词表阈值：15

kNN：k=5

距离计算公式：欧几里得距离、余弦相似度

	欧几里得距离	余弦相似度
准确率	**49.02%**	**87.57%**