《人工智能实验》

实验报告

Lab 1

数据处理与 KNN

学 院 名 称 ：	数据科学与计算机学院
专业（班级）：	17 级计算机科学与技术
学 生 姓 名 ：	薛伟豪
学 号 ：	17341178
联 系 方 式 ：	15013041671

Lab 1：数据处理与KNN

1. Task：文本数据集简单处理

1.1. 算法原理

根据要求，我们需要对实验提供的文本数据集进行处理。遍历整个文件，读取每一行中“有效的”句子部分，生成一个包含与文件行数相同的单词列表的列表（为叙述方便，这里将其称作词汇表），同时得到一个包含所有单词的不重复的单词向量(这里称作总单词向量)。

我们遍历词汇表中的每一个单词向量，通过判断总单词向量中的每一个单词是否在单词向量中出现，若出现则标记为出现的个数，若不出现则标记为0，从而得到One-Hot矩阵。

而后，再将One-Hot矩阵中每一个训练文本所对应的向量中单词出现的个数作归一化处理，得到TF矩阵。具体公式如下：

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

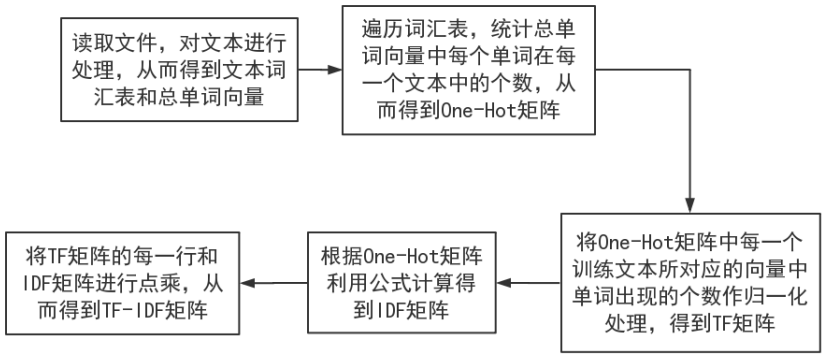
我们再对每个单词的普遍重要性进行度量，利用下列公式得到IDF矩阵。

$$idf_i = log\frac{|D|}{|\{j: t_i \in d_j\}|}$$

最后再根据以下公式得到TF-IDF矩阵

$$tfidf_{i,j} = tf_{i,j} \times idf_i$$

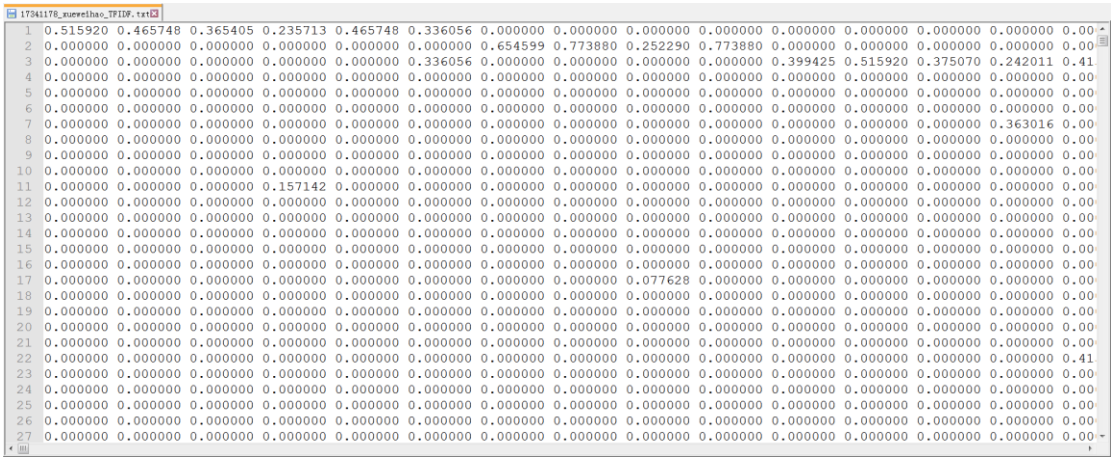
1.2. 流程图



1.3. 代码展示

```
import numpy as np

#读取文件，对文本进行处理，从而得到文本词汇表和总单词向量
file = open('D:/lab1_data/semeval.txt', 'r')
word_matrix = []
word_list = []
line_count = 0
for line in file:
    line_count += 1
    line = line.replace('\n', '')
    line = line.replace('\t', ' ')
    line = line.split(' ')
    word_matrix.append(line[8:])
    for item in line[8:]:
        if item not in word_list:
            word_list.append(item)

#统计总单词向量中每个单词在每一个文本中的个数，从而得到One-Hot 矩阵
one_hot_matrix = np.zeros((line_count, len(word_list)))
for row in range(len(word_matrix)):
    for col in range(len(word_list)):
        one_hot_matrix[row][col] = word_matrix[row].count(word_list[col])

#将One-Hot 矩阵中每一个训练文本所对应的向量作归一化处理，得到TF 矩阵
tf_matrix = one_hot_matrix.copy()
for row in range(len(tf_matrix)):
    tf_matrix[row] /= sum(one_hot_matrix[row])

#根据One-Hot 矩阵利用公式计算得到IDF 矩阵
idf_matrix = one_hot_matrix.sum(axis=0)
idf_matrix = np.log10(len(one_hot_matrix) / idf_matrix)

#将TF 矩阵的每一行和IDF 矩阵进行点乘，从而得到TF-IDF 矩阵
tfidf_matrix = tf_matrix * idf_matrix

#文件输出
np.savetxt("17341178_xueweihao_TFIDF.txt", tfidf_matrix)
file.close()
```

1.4. 实验结果及分析



可以看到，输出的txt文件里得到了我们预期的TF-IDF矩阵。

2. Task：使用KNN最近邻算法处理分类问题

2.1. 算法原理

KNN是有监督的机器学习模型。在处理分类问题上，它通过测量不同特征向量之间的距离按照多数投票原则进行分类。如果一个待预测样本的特征向量，与训练样本的特征空间中的K个距离最近的样本特征向量中的大多数属于某一个类别，则将该预测样本判定为属于该类别。

具体而言，在本次任务中，我们需要根据训练集给定的文本及其对应的情绪标签，来对验证集中的文本进行标签预测。通过改变参数、调整算法等方式，逐步提高验证集的预测准确率。我们选取能让验证集标签预测准确率达到最高的参数，利用该模型对测试集文本进行预测，以期能让测试集的标签预测准确率达到相对较高的水平。

在样本间距离计算上，有多种度量方法供我们选择，我主要考虑了以下几种：

假设给定两个n维的特征向量$\vec{X} = (x_1, x_2, x_3, \dots, x_n)$和$\vec{Y} = (y_1, y_2, y_3, \dots, y_n)$

- 欧氏距离

$$dist(\vec{X}, \vec{Y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- 曼哈顿距离

$$dist(\vec{X}, \vec{Y}) = \sum_{i=1}^{n}|x_i - y_i|$$

- 闵可夫斯基距离

$$dist(\vec{X},\vec{Y}) = \left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{\frac{1}{p}}$$

当p=1时，为曼哈顿距离

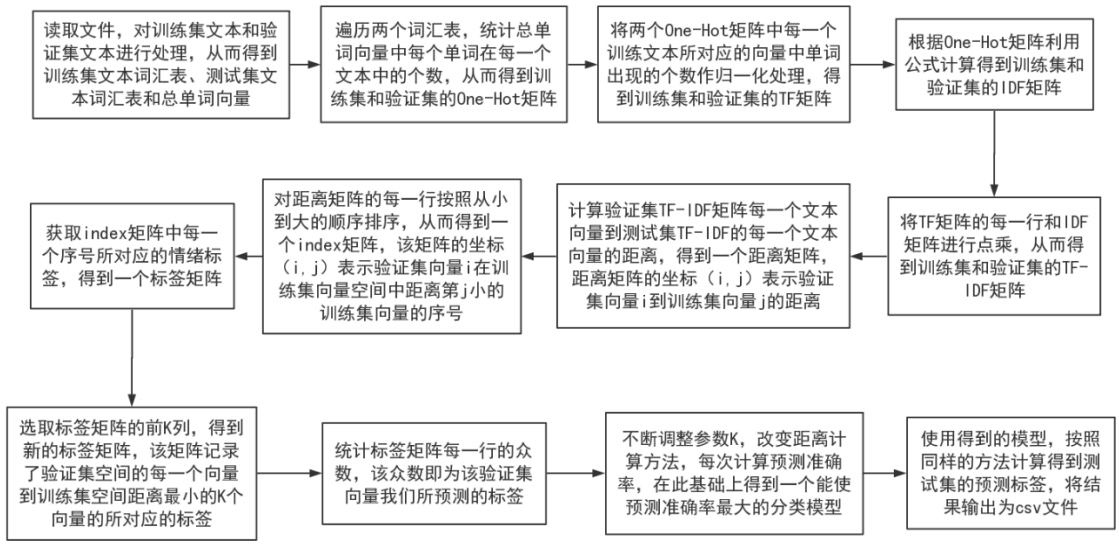
当p=2时，为欧氏距离

- 余弦距离

$$dist(\vec{X},\vec{Y}) = 1 - cos(\vec{X},\vec{Y}) = 1 - \frac{\vec{X}\cdot\vec{Y}}{|\vec{X}||\vec{Y}|}$$

在本次实验中，我选择使用余弦距离计算测试集向量和训练集向量之间的距离。

2.2. 流程图



2.3. 核心代码展示

- TF-IDF矩阵的生成（以训练集为例）

```
#生成 one-hot 矩阵
train_one_hot_matrix = np.zeros((len(word_matrix), len(word_list)))
for row in range(len(word_matrix)):
```

```
    for col in range(len(word_list)):
        train_one_hot_matrix[row][col] = word_matrix[row].count(word_list[col])

#生成 TF 矩阵
train_tf_matrix = train_one_hot_matrix.copy()
for row in range(len(train_tf_matrix)):
    train_tf_matrix[row] /= sum(train_one_hot_matrix[row])

#生成 IDF 矩阵
train_idf_matrix = train_one_hot_matrix.sum(axis=0)
train_idf_matrix = np.log10(len(train_one_hot_matrix) / (1 + train_idf_matrix))

#生成 TF-IDF 矩阵
train_tfidf_matrix = train_tf_matrix * train_idf_matrix
```

- **距离矩阵的生成**

```
dist_matrix = np.zeros((len(validation_word_matrix), len(word_matrix)))
for val_item, i in zip(val_tfidf_matrix, range(len(val_tfidf_matrix))):
    for train_item, j in zip(train_tfidf_matrix, range(len(train_tfidf_matrix))):
        #余弦距离
        dist_matrix[i][j] = 1 - np.dot(val_item, train_item) / (np.linalg.norm(val_item) *
(np.linalg.norm(train_item)))
```

- **使用KNN最近邻算法进行标签预测**

```
label_dict = {'joy': 1,
              'sad': 2,
              'fear': 3,
              'anger': 4,
              'surprise': 5,
              'disgust': 6
              }
#对距离矩阵的每一行按照从小到大的顺序排序，从而得到一个序号矩阵
#该矩阵的坐标（i,j）表示验证集向量 i 在训练集向量空间中距离第 j 小的训练集向量的序号
dist_index_matrix = np.argsort(dist_matrix, axis=1)

#选取序号矩阵的前 K 列，得到新的序号矩阵
#该矩阵记录了验证集空间的每一个向量到训练集空间距离最小的 K 个向量的所对应的序号
k_dist_index_matrix = dist_index_matrix[:, :K]
```

```
#获取序号矩阵中每一个序号所对应的情绪标签，得到一个标签矩阵
k_label_matrix = np.zeros(np.shape(k_dist_index_matrix))
for i in range(len(k_dist_index_matrix)):
    for j in range(K):
        k_label_matrix[i][j] = label_dict[train_list[k_dist_index_matrix[i][j]][1]]
k_label_matrix = k_label_matrix.astype(np.int32)

#统计标签矩阵每一行的众数，该众数即为该验证集向量我们所预测的标签
predicts = np.zeros((len(k_label_matrix), 1))
for row_index in range(len(k_label_matrix)):
    count_dict = {1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0}
    for item in k_label_matrix[row_index]:
        count_dict[item] += 1
    predicts[row_index] = max(count_dict, key=count_dict.get)
predicts = predicts.astype(np.int32)
```

2.4. 实验结果及分析

- **实验结果展示**

对测试集文本进行分类结果如图所示：

	A	B	C
1		Words (split by space)	label
2	1	senator carl krueger thinks ipods can kill you	surprise
3	2	who is prince frederic von anhalt	joy
4	3	prestige has magic touch	joy
5	4	study female seals picky about mates	surprise
6	5	no e book for harry potter vii	joy
7	6	blair apologises over friendly fire inquest	sad
8	7	vegetables may boost brain power in older adults	surprise
9	8	afghan forces retake town that was overrun by taliban	sad
10	9	skip the showers male sweat turns women on study says	surprise
11	10	made in china irks some burberry shoppers	fear
12	11	britain to restrict immigrants from new eu members	fear
13	12	canadian breakthrough offers hope on autism	surprise
14	13	russia to strengthen its military muscle	fear
15	14	alzheimer s drugs offer no help study finds	joy
16	15	uk police slammed over terror raid	fear
17	16	no rejects police state claim	joy
18	17	oprah announces new book club pick	joy
19	18	smith can t be buried until hearing	sad
20	19	african nation hopes whoopi can help	joy
21	20	tourism lags in bush s hometown	fear
22	21	air france klm profit rises	fear
23	22	au regrets sudan s expulsion of un envoy	joy

- **模型性能展示和分析**

在本次实验中，对预测准确率高低的影响因素主要有三个：第一是对文本数据集的处理方式（One-Hot矩阵/TF-IDF矩阵），第二是距离度量方式的选取（曼哈顿距离，欧氏距离，余弦距离，…），第三是KNN最近邻算法中参数K的选取。我对这几个影响因素均进行了尝试与测试，对预测准确率的影响具体如下所示：

K	曼哈顿距离		欧氏距离		闵可夫斯基距离 (p=3)		余弦距离	
	One-Hot	TF-IDF	One-Hot	TF-IDF	One-Hot	TF-IDF	One-Hot	TF-IDF
1	0.334405	0.398714	0.305466	0.321543	0.318328	0.273312	0.385852	0.440514
2	0.353698	0.437299	0.372990	0.318328	0.385852	0.299035	0.398714	0.437299
3	0.395498	0.401929	0.353698	0.260450	0.389068	0.212219	0.382637	0.424437
4	0.395498	0.385852	0.360129	0.254019	0.372990	0.192926	0.385852	0.443730
5	0.389068	0.405145	0.360129	0.231511	0.385852	0.218650	0.385852	0.421222
6	0.395498	0.385852	0.350482	0.212219	0.369775	0.228296	0.414791	0.424437
7	0.395498	0.382637	0.353698	0.199357	0.379421	0.180064	0.424437	0.443730
8	0.389068	0.379421	0.376206	0.183280	0.382637	0.170418	0.385852	0.450161
9	0.411576	0.369775	0.369775	0.189711	0.379421	0.163987	0.382637	0.424437
10	0.392283	0.382637	0.385852	0.170418	0.392283	0.163987	0.414791	0.450161
11	0.401929	0.372990	0.389068	0.192926	0.401929	0.160772	0.414791	0.427653
12	0.398714	0.376206	0.382637	0.212219	0.389068	0.176849	0.398714	0.411576
13	0.392283	0.385852	0.376206	0.221865	0.382637	0.170418	0.385852	0.424437
14	0.395498	0.392283	0.356913	0.228296	0.369775	0.173633	0.421222	0.401929
15	0.389068	0.408360	0.363344	0.218650	0.369775	0.160772	0.379421	0.392283
16	0.385852	0.392283	0.366559	0.215434	0.369775	0.154341	0.418006	0.411576
17	0.385852	0.389068	0.379421	0.225080	0.382637	0.173633	0.414791	0.398714
18	0.382637	0.398714	0.382637	0.234727	0.385852	0.170418	0.411576	0.382637
19	0.376206	0.392283	0.385852	0.244373	0.392283	0.167203	0.398714	0.385852
20	0.385852	0.395498	0.382637	0.257235	0.395498	0.173633	0.392283	0.389068
21	0.382637	0.389068	0.385852	0.250804	0.395498	0.186495	0.389068	0.389068
22	0.379421	0.385852	0.379421	0.286174	0.382637	0.196141	0.401929	0.398714
23	0.389068	0.376206	0.382637	0.282958	0.382637	0.189711	0.401929	0.395498
…	…	…	…	…	…	…	…	…
68						0.369775		
85				0.385852				

我们不难发现，采用余弦距离计算两个文本向量之间的距离要明显好于闵可夫斯基距离（包括曼哈顿距离和街区距离）。当K=10时，预测准确率达到最大值45.0161%

3. Task：使用KNN最近邻算法处理回归问题

3.1. 算法原理

KNN是有监督的机器学习模型。在处理回归问题上，通过计算一个待预测样本的特征向量与训练样本的特征空间中的K个距离最近的样本特征向量的距离，利用公式加权得到一个值来作为该预测样本的预测值。

具体而言，在本次任务中，我们需要根据训练集给定的文本及其对应的各种情绪的概率，来对验证集中的文本进行各种情绪的概率预测。我们使用相关系数作为衡量指标。通过改变参数、调整算法等方式，逐步提高验证集预测概率与实际概率相关系数。我们选取能让相关系数达到最高的参数，利用该模型对测试集文本进行概率预测，以期能让测试集的预测概率接近实际水平。

在样本间距离计算上，有多种度量方式供我们选择。和处理分类问题一样，我主要考虑了包括欧氏距离、曼哈顿距离、闵可夫斯基距离、余弦距离等距离度量方式，并最终选择了余弦距离作为距离度量方式。具体的公式在分类问题中已经给出，这里不再赘述。

在预测文本向量属于某一种情绪的概率时，我们使用了以下公式对K个最近邻向量的概率进行加权得到。具体公式如下（假定我们需要预测某个文本向量是joy的概率）：

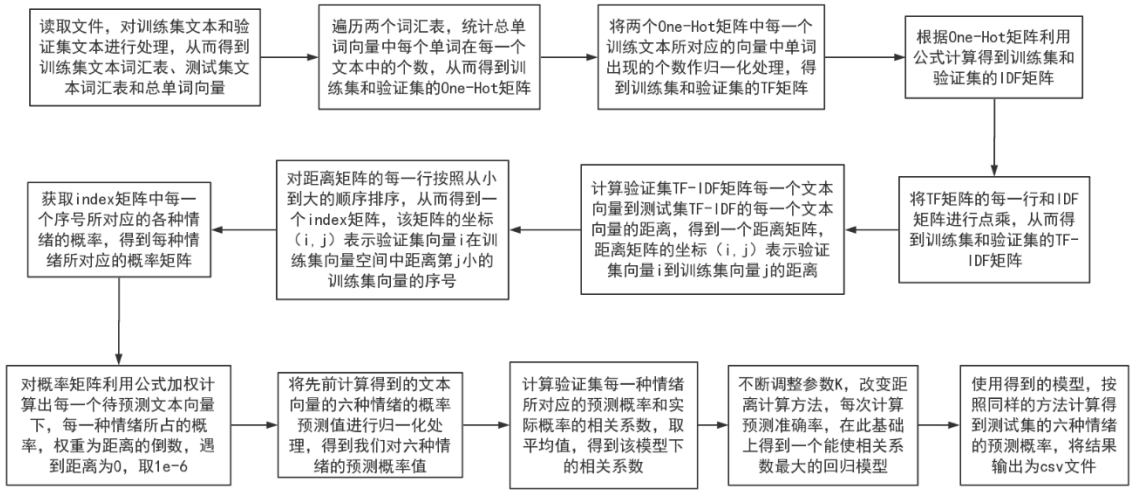
$$P(\vec{W}\ is\ joy) = \sum_{k=1}^{n} \frac{P(\overrightarrow{X_k}\ is\ joy)}{dist(\vec{W}, \overrightarrow{X_k})}$$

其中$\overrightarrow{X_k}$为训练集文本向量，$dist(\vec{W}, \overrightarrow{X_k})$表示该文本向量到训练集文本向量的距离。

而对于相关系数的计算，则需要用到以下公式：

$$r(\vec{X}, \vec{Y}) = \frac{Cov(\vec{X}, \vec{Y})}{\sqrt{Var[\vec{X}] \cdot Var[\vec{Y}]}}$$

3.2. 流程图



3.3. 核心代码展示

- **TF-IDF矩阵的生成（以训练集为例）**

```
#生成 one-hot 矩阵
train_one_hot_matrix = np.zeros((len(word_matrix), len(word_list)))
for row in range(len(word_matrix)):
    for col in range(len(word_list)):
        train_one_hot_matrix[row][col] = word_matrix[row].count(word_list[col])
#生成 TF 矩阵
train_tf_matrix = train_one_hot_matrix.copy()
for row in range(len(train_tf_matrix)):
    train_tf_matrix[row] /= sum(train_one_hot_matrix[row])
#生成 IDF 矩阵
train_idf_matrix = train_one_hot_matrix.sum(axis=0)
train_idf_matrix = np.log10(len(train_one_hot_matrix) / (1 + train_idf_matrix))
#生成 TF-IDF 矩阵
train_tfidf_matrix = train_tf_matrix * train_idf_matrix
```

- **距离矩阵的生成**

```
dist_matrix = np.zeros((len(validation_word_matrix), len(word_matrix)))
for val_item, i in zip(val_tfidf_matrix, range(len(val_tfidf_matrix))):
    for train_item, j in zip(train_tfidf_matrix, range(len(train_tfidf_matrix))):
        #余弦距离
        dist_matrix[i][j] = 1 - np.dot(val_item, train_item) / (np.linalg.norm(val_item) *
(np.linalg.norm(train_item)))
```

- **使用KNN最近邻算法进行概率预测**

```
#对距离矩阵的每一行按照从小到大的顺序排序，从而得到一个序号矩阵
#该矩阵的坐标（i,j）表示验证集向量 i 在训练集向量空间中距离第 j 小的训练集向量的序号
dist_index_matrix = np.argsort(dist_matrix, axis=1)

#选取序号矩阵的前 K 列，得到新的序号矩阵
#该矩阵记录了验证集空间的每一个向量到训练集空间距离最小的 K 个向量的所对应的序号
k_index_matrix = dist_index_matrix[:, :K]
#获取序号矩阵中每一个序号所对应的情绪的概率，得到该情绪所对应的概率矩阵
#由于有六种情绪，所以可以得到六个这样的矩阵
predicts = np.zeros((len(k_index_matrix), 6))
```

```
#根据公式进行计算
for mood in range(1, 7):
    k_prob_matrix = np.zeros(np.shape(k_index_matrix))
    for i in range(len(k_index_matrix)):
        for j in range(K):
            #如果两个向量的距离为0，则将该距离视为1e-6
            if dist_matrix[i][k_index_matrix[i][j]] == 0:
                k_prob_matrix[i][j] = float(train_list[k_index_matrix[i][j]][mood]) / 1e-6
            else:
                k_prob_matrix[i][j] = float(train_list[k_index_matrix[i][j]][mood]) /
dist_matrix[i][k_index_matrix[i][j]]
    #得到加权计算后的该情绪所对应的概率（未归一化）
    for row_index in range(len(k_prob_matrix)):
        predicts[row_index][mood - 1] = k_prob_matrix[row_index].sum()

#对每一个文本向量的六种情绪的预测概率进行归一化处理
for row_index in range(len(predicts)):
    row_sum = predicts[row_index].sum()
    for col_index in range(6):
        predicts[row_index][col_index] /= row_sum
```

3.4. 实验结果及分析

- **实验结果展示**

对测试集文本进行六种情绪的概率预测结果如图所示：

	A	B	C	D	E	F	G	H
1		Words (split by space)	anger	disgust	fear	joy	sad	surprise
2	1	senator carl krueger thinks ipods can kill you	0.09501	0.04333	0.0459	0.11871	0.18983	0.50722
3	2	who is prince frederic von anhalt	0.05129	0.14977	0.07485	0.16987	0.25469	0.29954
4	3	prestige has magic touch	0	0	0.02127	0.41466	0.1718	0.39228
5	4	study female seals picky about mates	0.03992	0.05592	0.08158	0.37344	0.06769	0.38145
6	5	no e book for harry potter vii	0.06153	0.05338	0.01753	0.28575	0.25972	0.3221
7	6	blair apologises over friendly fire inquest	0.10466	0.05362	0.21049	0.0327	0.52914	0.0694
8	7	vegetables may boost brain power in older adults	0	0	0.09231	0.44558	7.8E-05	0.46204
9	8	afghan forces retake town that was overrun by taliban	0.19767	0	0.31041	0.07564	0.33247	0.08382
10	9	skip the showers male sweat turns women on study says	0	0.14055	0	0.17749	0	0.68196
11	10	made in china irks some burberry shoppers	0.13121	0.10511	0.16696	0.07252	0.16655	0.35765
12	11	britain to restrict immigrants from new eu members	0.15608	0.01989	0.24508	0.19433	0.20712	0.17751
13	12	canadian breakthrough offers hope on autism	0.05532	0	0.11475	0.42694	0.11237	0.29063
14	13	russia to strengthen its military muscle	0.12127	0.07109	0.31374	0.18172	0.18714	0.12504
15	14	alzheimer s drugs offer no help study finds	0.11251	0.0483	0.08374	0.23905	0.14705	0.36936
16	15	uk police slammed over terror raid	0.1663	0.13437	0.41879	0.01902	0.2133	0.04821
17	16	no rejects police state claim	0.20055	0.09897	0.36325	0.01589	0.16707	0.15428
18	17	oprah announces new book club pick	0.06278	0.11529	0.07134	0.39612	0.16708	0.18739
19	18	smith can t be buried until hearing	0.1276	0.04392	0.1786	0.06027	0.13948	0.45014
20	19	african nation hopes whoopi can help	0.0954	0.0315	0.13623	0.36046	0.17464	0.20177
21	20	tourism lags in bush s hometown	0.06854	0.00455	0.18521	0.43893	0.08567	0.21711
22	21	air france klm profit rises	0.18023	0.07388	0.19152	0.25322	0.19249	0.10866
23	22	au regrets sudan s expulsion of un envoy	0.20341	0.06615	0.07801	0.1018	0.18239	0.36824

- **模型性能展示和分析**

在本次实验中，对相关系数高低的影响因素主要有三个：第一是对文本数据集的处理方式（One-Hot矩阵/TF-IDF矩阵），第二是距离度量方式的选取（曼哈顿距离，欧氏距离，余弦距离，…），第三是KNN最近邻算法中参数K的选取。我对这几个影响因素均进行了尝试与测试，对相关系数的影响具体如下所示：

K	曼哈顿距离		欧氏距离		闵可夫斯基距离 (p=3)		余弦距离	
	One-Hot	TF-IDF	One-Hot	TF-IDF	One-Hot	TF-IDF	One-Hot	TF-IDF
1	0.147322	0.293626	0.142837	0.252234	0.162121	0.212411	0.294915	0.345829
2	0.209687	0.324354	0.180455	0.257667	0.188807	0.258742	0.334789	0.365469
3	0.211490	0.314575	0.200472	0.280143	0.219753	0.261100	0.323735	0.379263
4	0.239170	0.324640	0.217983	0.258431	0.217094	0.253742	0.326405	0.391517
5	0.237610	0.320818	0.222309	0.242641	0.220559	0.252740	0.326922	0.395807
6	0.243778	0.321180	0.223796	0.244816	0.231163	0.232776	0.336496	0.391437
7	0.245411	0.316561	0.229235	0.250893	0.233282	0.233862	0.332345	0.392696
8	0.265402	0.310625	0.241103	0.237915	0.231256	0.235362	0.331389	0.391572
9	0.285548	0.311825	0.246115	0.228966	0.247117	0.213516	0.329175	0.394024
10	0.288856	0.301312	0.246424	0.240908	0.259761	0.210484	0.334079	0.390658
11	0.275611	0.308250	0.257956	0.243996	0.250467	0.206985	0.324887	0.381556
12	0.271852	0.313144	0.242053	0.245906	0.244846	0.204098	0.315799	0.379971
13	0.272644	0.325747	0.236391	0.244945	0.229191	0.199197	0.311730	0.369162
14	0.267186	0.332209	0.233680	0.248928	0.216169	0.213098	0.304309	0.360461
15	0.264857	0.329834	0.235053	0.234807	0.214602	0.237020	0.302952	0.359449
16	0.258043	0.323267	0.237836	0.235526	0.223299	0.229624	0.304308	0.357536
17	0.251074	0.330303	0.239384	0.256953	0.219710	0.237431	0.298474	0.347450
18	0.249894	0.326661	0.242055	0.267119	0.218588	0.228467	0.295326	0.341767
19	0.255011	0.328554	0.245832	0.266092	0.221418	0.241823	0.295217	0.343279
20	0.262014	0.326140	0.237655	0.260814	0.224611	0.250722	0.299651	0.339868
21	0.253542	0.314506	0.241431	0.267879	0.226431	0.255548	0.302315	0.340667
22	0.259183	0.310695	0.235557	0.267334	0.227152	0.249704	0.305635	0.337098
23	0.255282	0.318315	0.240273	0.277162	0.229795	0.240894	0.300862	0.332681
…	…	…	…	…	…	…	…	…
53				0.291459				
60						0.290251		

我们不难发现，采用余弦距离计算两个文本向量之间的距离要明显好于闵可夫斯基距离（包括曼哈顿距离和街区距离）。当K=5时，相关系数最高可达到0.395807。

4．创新点

为了能让准确率和相关系数更大，我考虑了使用不同的文本向量的表达方式，是选用One-Hot矩阵还是TF-IDF矩阵？以及在模型训练时是否把验证集/测试集中特有的单词加入词库中我也进行了尝试与运行。结果发现，在使用欧氏距离计算时，One-Hot矩阵的准确率甚至略高，而在使用其他距离计算方法时，TF-IDF矩阵的准确率要比One-Hot矩阵好不少。而把验证集/测试集中特有的单词加入词库中也能明显提高准确率和相关系数。

同时，我在计算两个文本向量之间的距离时采用了多种计算方式，包括了曼哈顿距离、欧氏距离、p不同取值的闵可夫斯基距离以及余弦距离，经过比较可以发现，在使用余弦距离计算向量之间的距离时能够使得预测准确率达到最大。

此外，我在博客上看到了修正余弦相似度的概念。由于余弦相似度只考虑了两个向量之间的角度，没有考虑到数值对于结果的影响。而修正余弦相似度在计算时减去了某一数据的平均值，并在此基础上有了修正余弦距离，理论上来说修正余弦距离衡量的因素更多。但是我在使用其对验证集进行预测时，发现并没有能够达到想象中更好的效果。因此综合来看，使用余弦距离效果最好。

5. 思考题

5.1. IDF的第二个计算公式中分母多了个1是为什么？

如果所有文件中均不包含某个词语，就会导致除数为0，因此我们让分母加上1以防止除数为0的情况。

5.2. IDF数值有什么含义？TF-IDF数值有什么含义？

IDF指的是逆向文件频率，是一个词语普遍重要性的度量。IDF的值越小，说明这个词在单词库越常见。

而TF-IDF的数值由TF数值和IDF数值共同决定。它与一个词在某条文本中的出现次数成正比，与该词在整个单词库中的出现次数成反比。如果TF-IDF的值足够大，则可以认为该词具有很好的类别区分能力，适合用来分类。

5.3. 为什么使用距离的倒数作为权重？

如果两个文本向量的距离越近，就能够说明这两个文本向量的情绪越相似，其对应的情绪概率的权重应当更大。因此我们使用距离的倒数作为权重，距离越小，距离的倒数越大，所对应的权重越大。

5.4. 同一测试样本的各个情感概率总和应该为1，如何处理？

对同一个文本所预测的概率进行归一化处理，使它们的总和为1，具体有：

$$P_{new}(\vec{X}\ is\ label_i) = \frac{P_{former}(\vec{X}\ is\ label_i)}{\sum_k P_{former}(\vec{X}\ is\ label_k)}$$