中山大学计算机学院

人工智能

本科生实验报告

（2022 学年春季学期）

课程名称：Artificial Intelligence

教学班级	信息与计算科学班	专业（方向）	信息与计算科学
学号	21311359	姓名	何凯迪

一、 实验题目

在给定文本数据集完成文本情感聚类训练。

要求：

- 文本的特征可以使用 TF 或 TF-IDF （可以使用 sklearn 库提取特征）
- 利用 K-means，K-means++对文本特征进行聚类
- 计算 calinski_harabasz_score（越高越好）可调用 sklearn.metrics.calinski_harabasz_score 函数计算
- 实验报告应包含样本的可视化展示，可以考虑利用 TSNE 将文本特征投影到直角坐标系中进行展示（可以调用 sklearn 库的 TSNE 投影，可视化工具不限)

二、 实验内容

1. 算法原理

K-means 算法：

K-means 算法是一种常见的聚类算法，用于将数据点分成不同的簇或群组。它的原理相对简单，主要包括以下步骤：

选择 K 个初始聚类中心点：首先需要确定要将数据分成的簇的数量 K。然后从数据集中随机选择 K 个点作为初始的聚类中心。

分配数据点到最近的聚类中心：对于每个数据点，计算它与每个聚类中心的距离，并将其分配到距离最近的聚类中心所在的簇。

更新聚类中心：对于每个簇，计算该簇中所有数据点的平均值，将这个平均值作为新的聚类中心。

重复步骤 2 和步骤 3，直到聚类中心不再改变或达到预定义的停止条件（如达到最大迭代次数）为止。

K-means 算法的目标是最小化数据点与所属簇的聚类中心之间的距离，即最小化簇内的方差或均方误差。它通过迭代地更新聚类中心来优化这个目标函数，直到达到收敛状态。

K-means++算法：

K-means++算法通过改进初始聚类中心的选择过程，使得初始聚类中心能够更好地覆盖数据空间。相比于传统的 K-means 算法，K-means++算法能够更稳定地收敛到全局最优解，并且对于不同的初始选择具有较好的鲁棒性。步骤如下：

初始化第一个聚类中心：从数据集中随机选择一个点作为第一个聚类中心。计算每个数据点到最近聚类中心的距离 D(x)：对于每个数据点 x，计算它与已选择的聚类中心之间的距离，并将最短距离记为 D(x)。

选择下一个聚类中心：根据概率分布选择下一个聚类中心。具体过程如下：对于每个数据点 x，计算它被选择为下一个聚类中心的概率，概率计算公式为：$P(x) = D(x)^2 \ / \ \Sigma(D(x)^2)$，其中 $\Sigma(D(x)^2)$表示所有数据点的 $D(x)^2$ 之和。
通过随机采样的方式，选择下一个聚类中心。选择的概率越大，被选中的概率也越高。

重复步骤 2 和步骤 3，直到选择了 K 个聚类中心。

执行传统的 K-means 算法：使用选择的聚类中心作为初始中心，然后执行传统的 K-means 算法的步骤 2 和步骤 3，即将数据点分配到最近的聚类中心并更新聚类中心，直到达到停止条件。

2. 伪代码

K-means：

```
输入：数据集 X，聚类数量 K
输出：聚类结果

1. 从数据集 X 中随机选择 K 个点作为初始聚类中心
2. 初始化聚类结果 C，将每个数据点分配到最近的聚类中心
repeat:
    3. 更新聚类中心：
       对于每个聚类 i = 1 to K:
           计算聚类 i 中所有数据点的均值，作为新的聚类中心
    4. 更新聚类结果 C，将每个数据点分配到最近的聚类中心
until 聚类结果 C 不再改变或达到停止条件
输出聚类结果
```

K-means++：

```
输入：数据集 X，聚类数量 K
输出：聚类结果

1. 从数据集 X 中随机选择一个点作为第一个聚类中心 c1
2. for i = 2 to K:
       计算每个数据点 x 到已选择的聚类中心的最短距离 D(x)
       选择下一个聚类中心 c_i，使得它的选择概率正比于 D(x)²
```

```
3. 初始化聚类结果 C，将每个数据点分配到最近的聚类中心
repeat:
    4. 更新聚类中心:
       对于每个聚类 i = 1 to K:
           计算聚类 i 中所有数据点的均值，作为新的聚类中心
    5. 更新聚类结果 C，将每个数据点分配到最近的聚类中心
until 聚类结果 C 不再改变或达到停止条件
输出聚类结果
```

3. 关键代码展示（带注释）

K-means：

```python
def kmeans(X, k, max_iterations=100):
    # 随机选择 k 个样本作为初始点
    centroids = X[np.random.choice(X.shape[0], k, replace=False)]
    prev_centroids = np.zeros_like(centroids)
    labels = np.zeros(X.shape[0])

    for _ in range(max_iterations):
        # 计算每个样本到点的距离，并分配到最近的簇
        distances = np.linalg.norm(X[:, np.newaxis] - centroids, axis=-1)
        labels = np.argmin(distances, axis=1)
        # 更新点位置
        for i in range(k):
            centroids[i] = np.mean(X[labels == i], axis=0)
        # 如果点不再改变，提前结束迭代
        if np.allclose(centroids, prev_centroids):
            break
        prev_centroids = centroids.copy()
    return labels, centroids
```

K-means++：

```python
def kmeans_plusplus(X, k, max_iterations=100):
    # 使用 K-means++ 初始化点
    centroids = [X[np.random.choice(X.shape[0])]]
    while len(centroids) < k:
        distances = np.linalg.norm(X[:, np.newaxis] - centroids, axis=-1)
        min_distances = np.min(distances, axis=1)
        probabilities = min_distances / np.sum(min_distances)
        next_centroid = X[np.random.choice(X.shape[0], p=probabilities)]
        centroids.append(next_centroid)
```

```
    prev_centroids = np.zeros_like(centroids)
    labels = np.zeros(X.shape[0])
    for _ in range(max_iterations):
        distances = np.linalg.norm(X[:, np.newaxis] - centroids, axis=-1)
        labels = np.argmin(distances, axis=1)
        for i in range(k):
            centroids[i] = np.mean(X[labels == i], axis=0)
        if np.allclose(centroids, prev_centroids):
            break
        prev_centroids = centroids.copy()
    return labels, centroids
```

降维：

```
# 使用 TSNE 进行降维和可视化
tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X)
```

画散点图：

```
# 绘制聚类结果的散点图
plt.figure(figsize=(10, 6))
colors = ['b', 'g', 'r', 'c', 'm', 'y']  # 聚类簇的颜色
for i in range(k):
    cluster_points = X_tsne[labels == i]
    plt.scatter(cluster_points[:, 0], cluster_points[:, 1], c=colors[i],
label="Cluster "+str(i+1))
plt.legend()
plt.title("Clustering Visualization (K=6)")
plt.show()
```

4. 创新点&优化（如果有）

用新的特征提取方法进行预实验，并选用结果最好的一种（CountVectorizer）

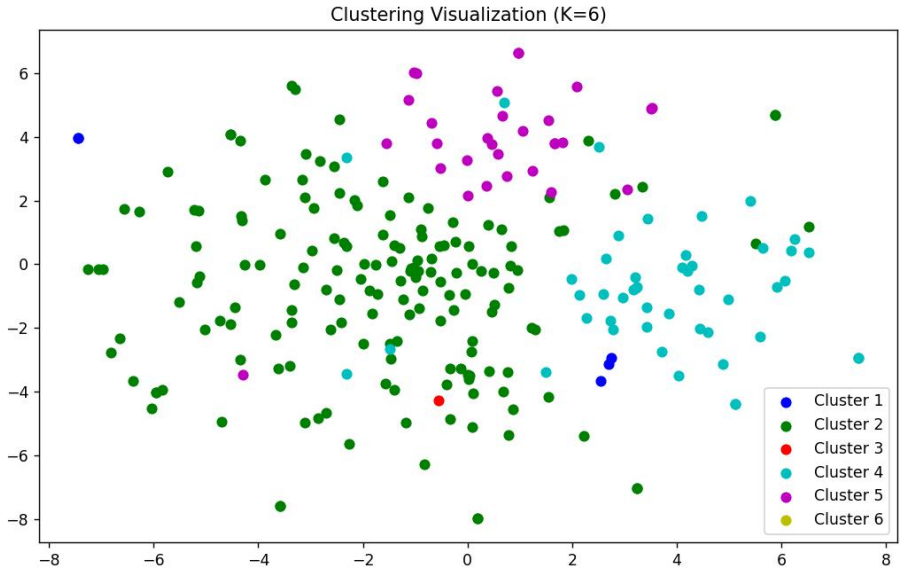
```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(data).toarray()
```

三、 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

对于使用 K-means 的聚类结果：

降维投影得到的聚类图：

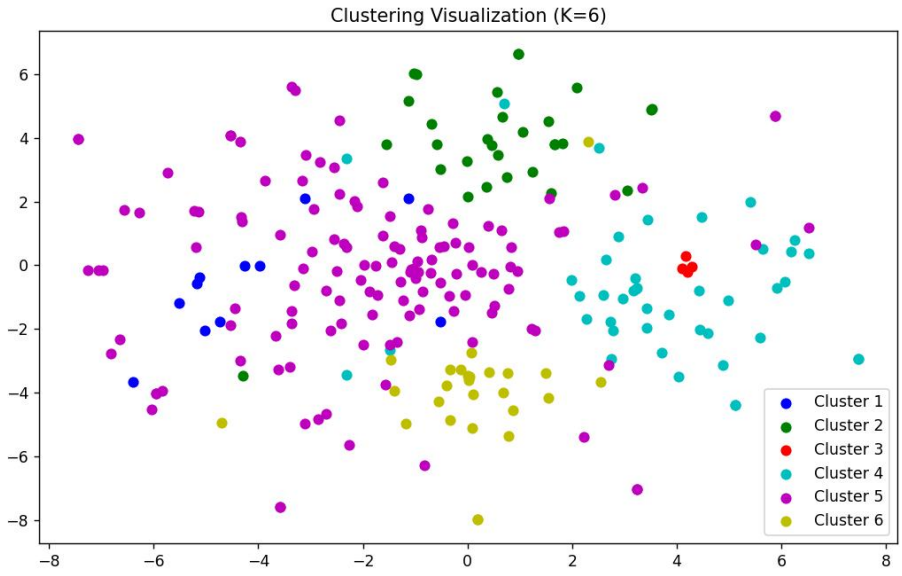


计算所得的 calinski_harabasz_score 值：

```
输出    调试控制台    终端
       'teacher', 'teen', 'tell', 'tension', 'terror', 'test', 'testifi',
       'th', 'that', 'the', 'thei', 'their', 'then', 'think', 'threat',
       'three', 'tiger', 'time', 'titl', 'to', 'toddler', 'tom', 'tool',
       'top', 'torn', 'tot', 'touch', 'town', 'tragedi', 'trailblaz',
       'tre', 'tread', 'trio', 'troop', 'tropic', 'truth', 'tuition',
       'tumor', 'tune', 'turkish', 'turn', 'tv', 'two', 'type', 'uk',
       'ukrain', 'un', 'unarm', 'under', 'unit', 'unlock', 'unveil', 'up',
       'upris', 'uranium', 'urg', 'us', 'vaccin', 'valdez', 'vega',
       'veget', 'victori', 'video', 'vike', 'violenc', 'violent',
       'virginia', 'visit', 'vow', 'wa', 'wai', 'want', 'war', 'warn',
       'warner', 'wash', 'wast', 'watch', 'we', 'weaken', 'weather',
       'web', 'wed', 'weigh', 'were', 'west', 'white', 'who', 'will',
       'win', 'winner', 'with', 'withnail', 'without', 'wive', 'women',
       'worker', 'world', 'worri', 'wors', 'would', 'wrangl', 'writer',
       'yanke', 'year', 'yet', 'you', 'young', 'youtub', 'zero'],
      dtype='<U13')]
Calinski-Harabasz Score: 4.602993849250464
```

对于使用 K-means++的聚类结果：

降维投影得到的聚类图：



计算所得的 calinski_harabasz_score 值：

```
输出    调试控制台    终端
       'date', 'deep', 'delight', 'didn', 'difficult', 'doubt', 'emerg',
       'emot', 'end', 'ex', 'exec', 'expect', 'explain', 'flu', 'for',
       'foreign', 'franc', 'friend', 'full', 'gain', 'game', 'global',
       'goal', 'good', 'gov', 'gunman', 'harri', 'hastert', 'heal',
       'heard', 'holm', 'honour', 'hp', 'hunt', 'in', 'intern', 'into',
       'iran', 'iraq', 'ivori', 'jail', 'kati', 'korean', 'la', 'laugh',
       'letter', 'life', 'madonna', 'mai', 'marathon', 'market', 'massiv',
       'miss', 'mission', 'new', 'not', 'of', 'offici', 'old', 'on',
       'pair', 'potter', 'presid', 'quit', 'rate', 'reach', 'realiz',
       'recal', 'recoveri', 'regret', 'research', 'resolut', 'riot',
       'rocket', 'routin', 'run', 'russian', 'safe', 'scandal',
       'scientist', 'search', 'set', 'sheva', 'ship', 'slow', 'sludg',
       'soni', 'soon', 'sorri', 'space', 'suburb', 'surviv', 'tape',
       'terror', 'tiger', 'time', 'tom', 'tragedi', 'tumor', 'turn',
       'type', 'uranium', 'vaccin', 'vega', 'virginia', 'wa', 'warn',
       'weather', 'wed', 'winner', 'year', 'young'], dtype='<U13')]
Calinski-Harabasz Score: 4.471692520479369
```

2. 评测指标展示及分析

以 K-means 为例，分别对 CountVectorizer 和 TfidfVectorizer 进行五次预实验

不难看出，使用 CountVectorizer 提取特征的效果显著由于后者。

方法	结果1	结果2	结果3	结果4	结果5	均值
CountVectorizer	4.001168051	3.450099073	5.164863706	2.83205834	1.222775084	3.334192851
TfidfVectorizer	1.683028823	1.560706268	1.942972476	2.167023352	2.103600346	1.891466253

四、 参考资料

TSNE: sklearn.manifold.TSNE — scikit-learn 1.0.2 documentation
Matplotlib 可视化教程:
https://www.runoob.com/matplotlib/matplotlib-tutorial.html