第五讲 - 文本聚类

张建章

阿里巴巴商学院
杭州师范大学

2023-02-22

目录

文本聚类的含义和用途

文本聚类：将文本集合划分成不同的类别，使得同一类别内的文本具有较高的主题相似度，而不同类别的文本具有较低的主题相似度。最常用的探索性文本挖掘技术之一，通常采用无监督机器学习算法，广泛用于商业文档管理、新闻聚合、客服反馈分析。

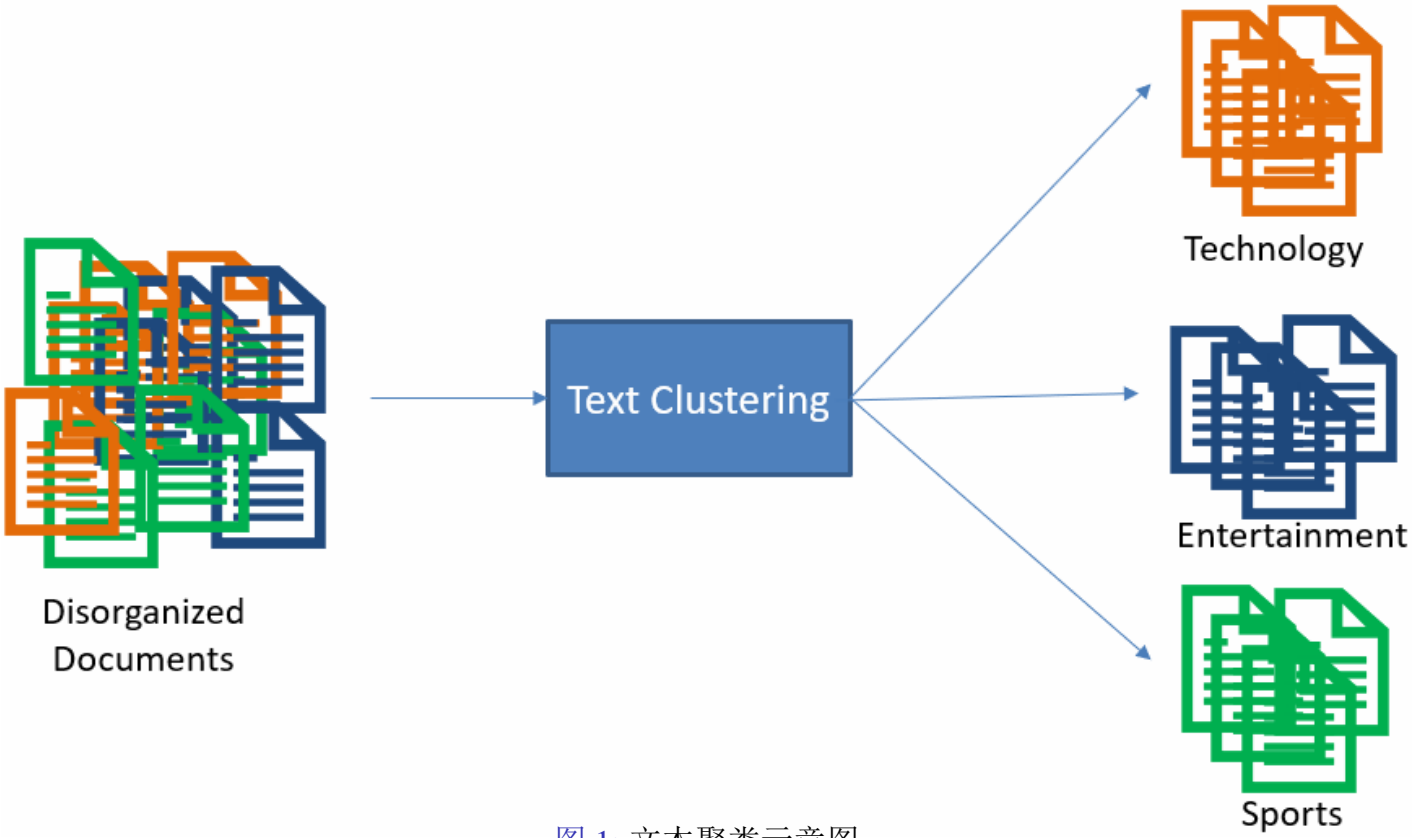
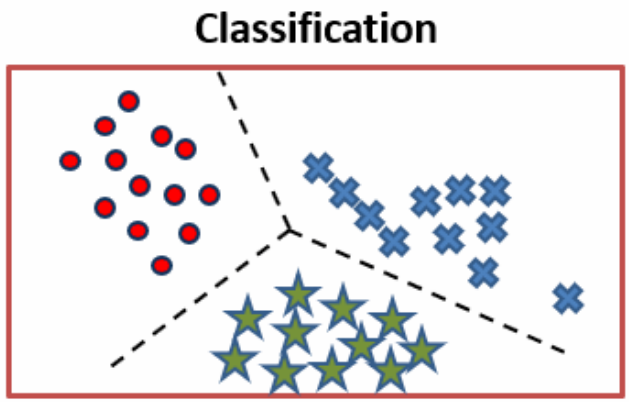


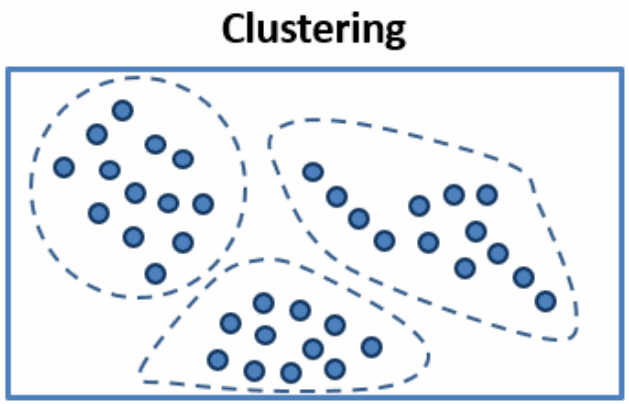
图 1: 文本聚类示意图

文本聚类与文本分类的区别

两者都是常用的文本挖掘方法，其主要区别如下：

- **学习方式**：文本分类是监督学习方法，需要事先已标注的训练集，来训练模型从而对对未知文本分类；文本聚类是无监督学习方法，不需要事先标注数据，根据文本之间的相似度自动将文本分组。
- **应用场景**：文本分类用于有明确分类标准的场景，如垃圾邮件分类、情感分析等；文本聚类适用于探索性数据分析，如新闻聚合、挖掘用户反馈中隐含的主题等。
- **学习算法**：文本分类常用算法有朴素贝叶斯、支持向量机、神经网络等。文本聚类常用算法有 K-均值、层次聚类、DBSCAN 等。





任务形式化定义

给定一个文本集合 $D = \{d_1, d_2, \ldots, d_n\}$，$d_i$ 表示一个文档。聚类结果是通过一个分配函数 $f: D \to C$，将这些文档分成 K 个类簇，$C = \{C_1, C_2, \ldots, C_K\}$，使每个类簇内的文档相似度较高，而不同类簇间的文档相似度较低。

文档相似度通常使用余弦相似度：

$$\text{sim}(d_i, d_j) = \cos(\theta) = \frac{\langle d_i, d_j \rangle}{\|d_i\| \|d_j\|} = \frac{d_i \cdot d_j}{\|d_i\| \|d_j\|}$$

最大化类内相似度和最小化类间相似度分别表示如下：

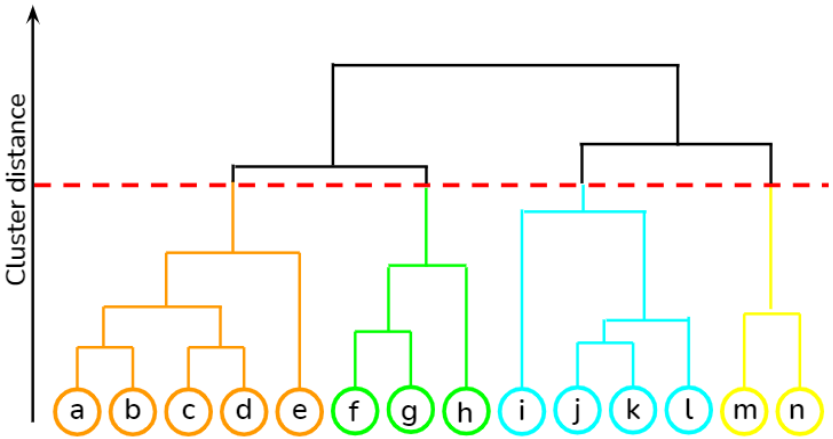
$$\text{argmax} \sum_{k=1}^{K} \sum_{d_i, d_j \in C_k} \text{sim}(d_i, d_j)$$

$$\text{argmin} \sum_{k=1}^{K} \sum_{d_i \in C_k, d_j \notin C_k} \text{sim}(d_i, d_j)$$

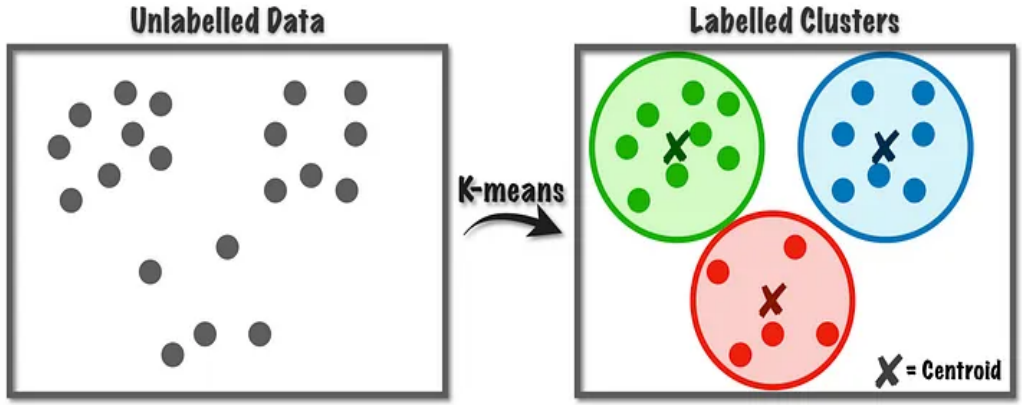
层次聚类和分区聚类

层次聚类：生成树状的聚类结构，可以生成多个聚类层次，展示数据点之间的层次关系，通过截断树状结构来得到不同数量的聚类，代表性算法有自底向上的凝聚层次聚类 (HAC) 和自顶向下的分裂层次聚类。

分区聚类：将数据点分割成一组不重叠的子集或类簇，没有层次结构，通常需要预先确定聚类的数量 K，代表性算法为 K-均值聚类。



层次聚类



分区聚类

HAC 算法原理

输入：一组向量化表示的文档 (N 个文档)，如 TF-IDF，嵌入向量等；

1. **初始化：**将每个数据点看作一个独立的簇；
2. **计算类簇相似性：**通过余弦相似度计算类簇之间的相似性，计算方式有单链接、全链接、平均链接、质心法；
3. **合并类簇：**将相似度最高的两个类簇合并为一个新的类簇，更新类簇相似性，
4. **重复：**重复步骤③ $N-1$ 次，直到全部文档合并成一个类簇。

输出：树形结构表示文档之间的层次关系，可通过截断树状结构得到所需数量的类簇。

HAC 原理动图演示 (点我观看)

HAC 算法流程

SIMPLEHAC$(d_1, \ldots, d_N)$

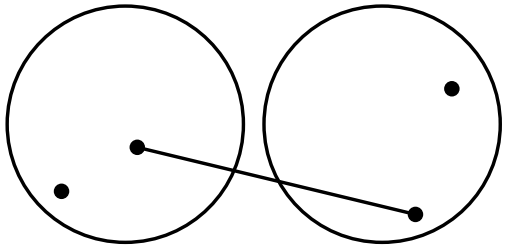
1 **for** $n \leftarrow 1$ **to** N
2 **do for** $i \leftarrow 1$ **to** N
3 **do** $C[n][i] \leftarrow \text{SIM}(d_n, d_i)$
4 $I[n] \leftarrow 1$ *(keeps track of active clusters)*
5 $A \leftarrow []$ *(assembles clustering as a sequence of merges)*
6 **for** $k \leftarrow 1$ **to** $N - 1$
7 **do** $\langle i, m \rangle \leftarrow \arg\max_{\{\langle i,m \rangle : i \neq m \wedge I[i]=1 \wedge I[m]=1\}} C[i][m]$
8 $A.\text{APPEND}(\langle i, m \rangle)$ *(store merge)*
9 **for** $j \leftarrow 1$ **to** N
10 **do** $C[i][j] \leftarrow \text{SIM}(i, m, j)$
11 $C[j][i] \leftarrow \text{SIM}(i, m, j)$
12 $I[m] \leftarrow 0$ *(deactivate cluster)*
13 **return** A

HAC 算法原理解析

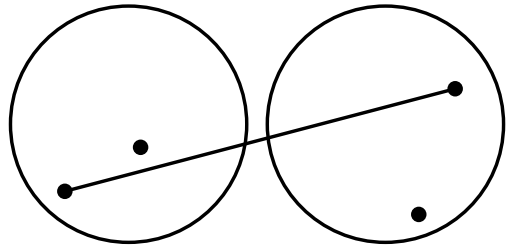
- Line 1-3: 计算一个 $N \times N$ 维的相似度矩阵 C;
- Line 4: 初始化一个 $1 \times N$ 维的 0-1 向量 I，保存可被继续合并的类簇，例如，$I[1] = 1$ 表示第二个文档处于未被合并状态，$I[3] = 0$ 表示第三个文档已被合并到某个类簇中，处于冻结状态;
- Line 5: 初始化一个序列 A，保存类簇合并过程;
- Line 6-12: 合并最相似的类簇，并更新矩阵 C、向量 I、序列 A。

注意: 理解上述算法流程伪代码的关键是将每一个类簇命名为一个数字编号 i，该编号为类簇中索引值最小的文档的索引值。

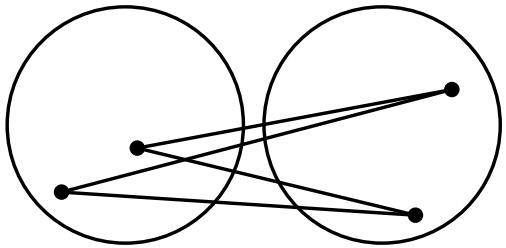
HAC 中的类簇相似度有四种：单链接、全链接、质心法、平均链接：



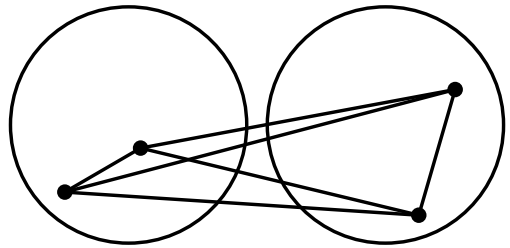
(a) single-link: maximum similarity



(b) complete-link: minimum similarity



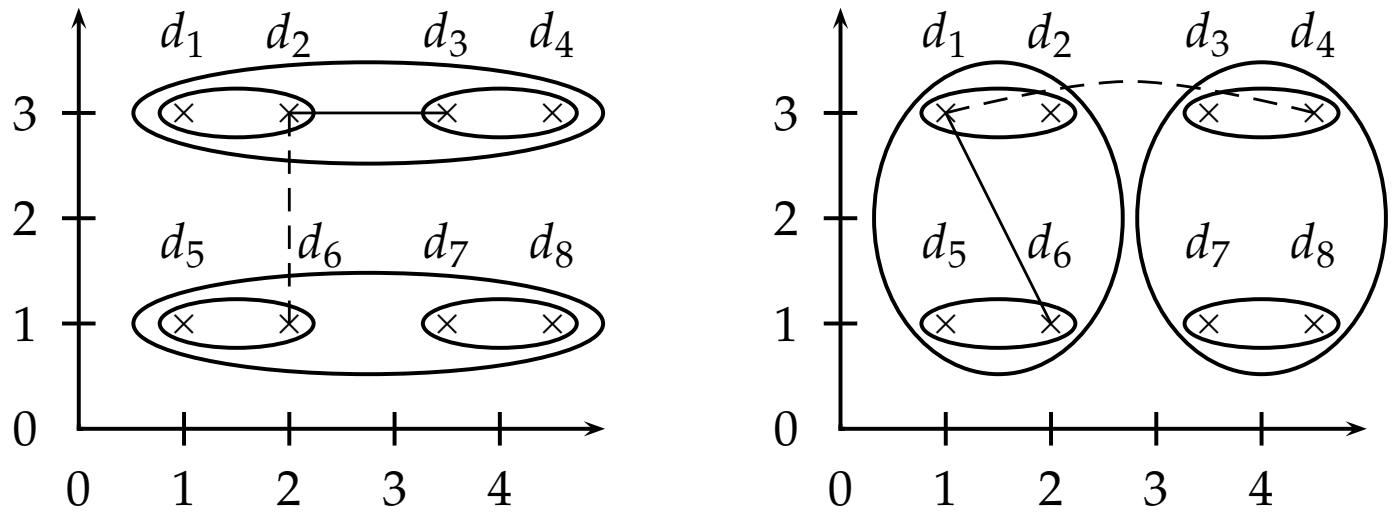
(c) centroid: average inter-similarity



(d) group-average: average of all similarities

类簇相似性示例

使用单链接 (左) 和全链接 (右) 进行 HAC 聚类：



前四次迭代，左右两图中的聚类过程相同，可表示为:

$< 1, 2 > \rightarrow < 3, 4 > \rightarrow < 5, 6 > \rightarrow < 7, 8 >$

单链接第四次之后的迭代聚类过程，可表示为:

$< 1, 3 > \rightarrow < 5, 7 > \rightarrow < 1, 5 >$

全链接第四次之后的迭代聚类过程，可表示为:

$< 1, 5 > \rightarrow < 3, 7 > \rightarrow < 1, 3 >$

平均链接和质心法

单链接和全链接只依赖两个文档的余弦相似度来度量类簇的相似度，无法完整反映一个类簇中的文档的分布情况。平均链接和质心法在计算类簇相似度时会考虑类簇中的全部文档。

平均链接计算两个类簇中全部文档的成对相似度：

$$\frac{1}{(N_i+N_j)(N_i+N_j-1)}\sum_{d_m\in w_i\cup w_j}\sum_{d_n\in w_i\cup w_j, d_n\neq d_m}\vec{d_m}\cdot\vec{d_n}$$

质心法计算两个类簇质心的相似度：

$$(\frac{1}{N_i}\sum_{d_m\in w_i}\vec{d_m})\cdot(\frac{1}{N_j}\sum_{d_n\in w_j}\vec{d_n})$$

```
import numpy as np
from sklearn.cluster import AgglomerativeClustering

#  使用TF-IDF向量表示文档集合
tfidf_matrix = np.array([
[0.1, 0.3, 0.1],
[0.1, 0.5, 0.2],
[0.6, 0.1, 0.1],
[0.7, 0.1, 0.2],
[0.2, 0.4, 0.6],
[0.1, 0.2, 0.7]])

# 设定聚类数量
n_clusters = 3
# 创建AgglomerativeClustering对象
clustering = AgglomerativeClustering(n_clusters=n_clusters,
↪  affinity='cosine', linkage='average')
# 使用TF-IDF矩阵进行聚类
labels = clustering.fit_predict(tfidf_matrix)
# 打印聚类结果
print("聚类标签:", labels)
```

1. 使用本课程提供的中文新闻语料库，使用 scikit-learn 中聚类算法进行文本聚类，具体要求如下：

① 从每个类别的新闻中随机采样 100 条新闻，构造待聚类文档集合；

② 分别使用 HAC 和 K 均值方法对文档集合进行聚类；

③ 分析两种聚类方法聚类结果的差异；

THE END