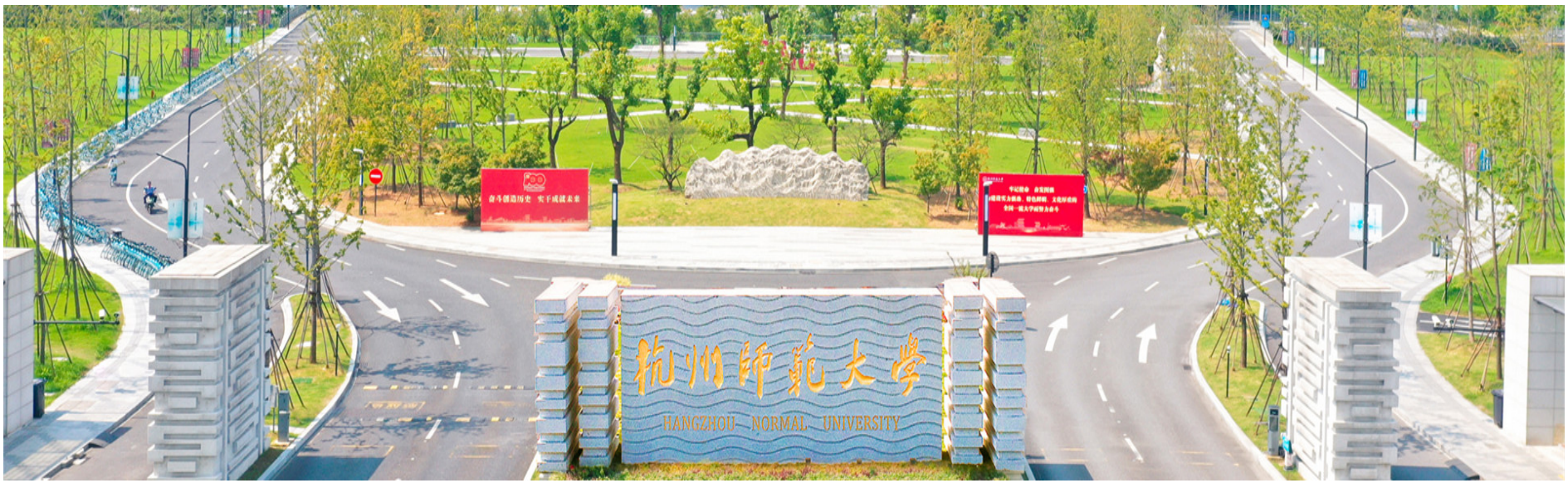
第五讲 - 文本聚类

张建章

阿里巴巴商学院
杭州师范大学

2023-02-22



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。

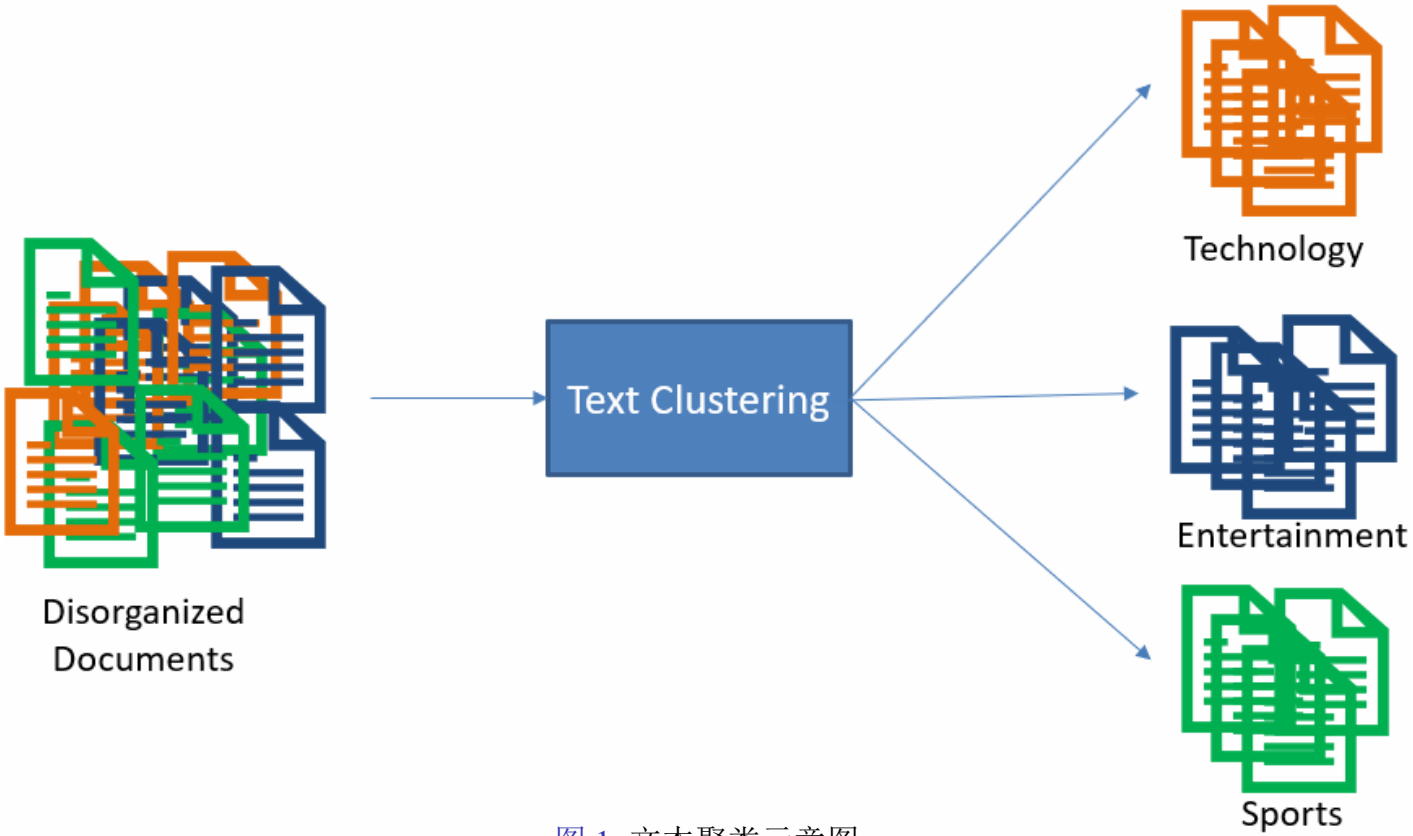
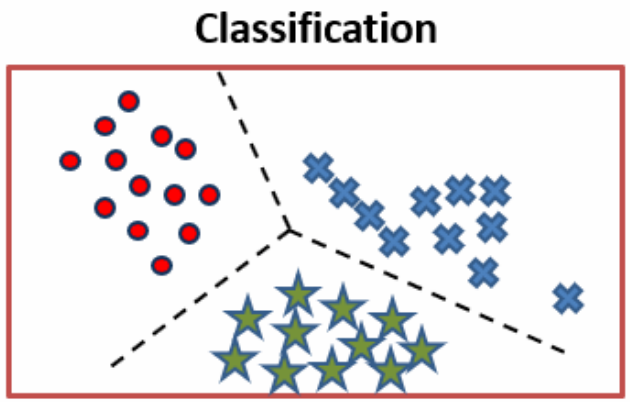


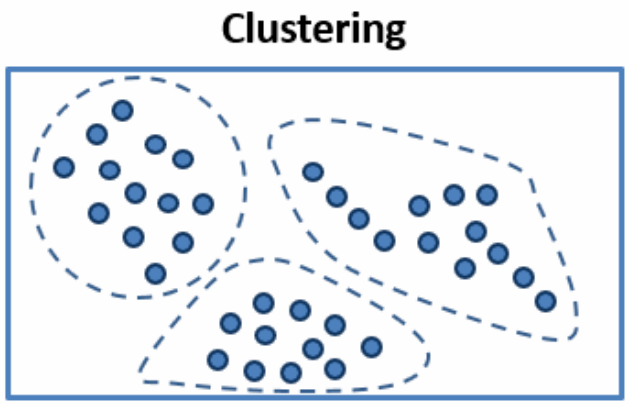
图 1: 文本聚类示意图

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。





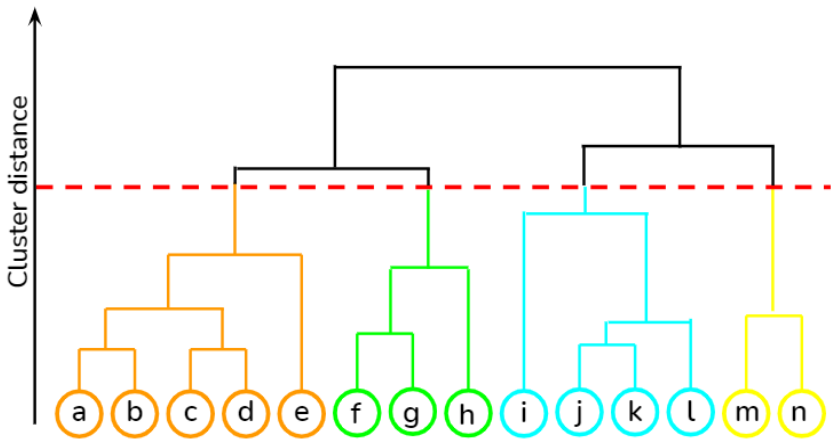
层次聚类和分区聚类

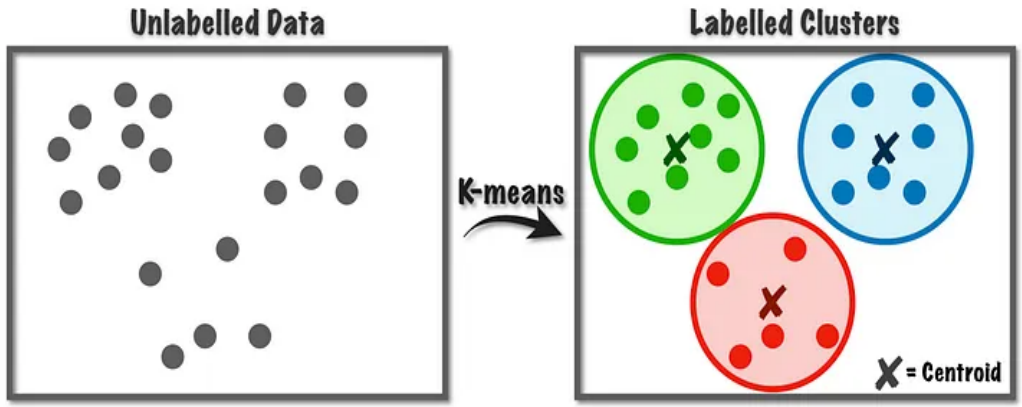
平面聚类：创建一组扁平的类簇，类簇之间没有显式的结构关系，代表性算法为 K-均值聚类。

层次聚类：创建一组具有层次结构的类簇，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。

硬聚类：每个文档最终被划分到一个类簇，K 均值和层次聚类都属于硬聚类。

软聚类：每个文档最终以不同的概率值归属于不同的类簇，代表性算法有潜在语义分析 (LSA)。





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，$\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$，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 (或欧氏距离)：

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

最大化类内相似度和最小化类间相似度分别表示如下：

$$\mathrm{argmax} \sum_{k=1}^{K} \sum_{d_i, d_j \in \omega_k} \mathrm{sim}(d_i, d_j), \quad \mathrm{argmin} \sum_{k=1}^{K} \sum_{d_i \in \omega_k, d_j \notin \omega_k} \mathrm{sim}(d_i, d_j)$$

文档聚类中的两个关键输入参数为：类簇数量 K 和距离函数 (或相似度度量)。文档使用长度归一化的向量表示。

K 均值算法原理

K-means 是最重要的平面聚类算法，其目标是最小化每个文档与其聚类中心的残差平方和 (residual sum of squares, RSS), 即欧氏距离平方和，定义如下：

$$RSS = \sum_{k=1}^{K} \sum_{\vec{x} \in \omega_k} |\vec{x} - \vec{\mu}(\omega_k)|^2$$

其中 $\vec{x}$ 为文档 d 的向量化表示，$\vec{\mu}(\omega_k)$ 表示第 k 个类簇的质心 (centroid)，定义如下：

$$\vec{\mu}(\omega_k) = \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$$

K-Means 中，理想的类簇是一个以质心为重心的球体，其目标函数度量了质心对同类簇中文档的代表性。K-Means 原理动图 (点我观看)

K-Means 算法流程

K-MEANS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$)
1 $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow$ SELECTRANDOMSEEDS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$)
2 **for** $k \leftarrow 1$ **to** K
3 **do** $\vec{\mu}_k \leftarrow \vec{s}_k$
4 **while** stopping criterion has not been met
5 **do for** $k \leftarrow 1$ **to** K
6 **do** $\omega_k \leftarrow \{\}$
7 **for** $n \leftarrow 1$ **to** N
8 **do** $j \leftarrow \arg\min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$
9 $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$ *(reassignment of vectors)*
10 **for** $k \leftarrow 1$ **to** K
11 **do** $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$ *(recomputation of centroids)*
12 **return** $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$

(1) 随机选择 K 个文档作为初始的类簇质心；

重复执行以下两步，直到达到终止条件：

(2) 将每个文档分配到距离其最近的质心所属的类簇；

(3) 重新计算更新类簇质心；

常用的 K-Means 终止条件

(1) 达到预定义的迭代次数；

(2) 聚类结果不再发生变化 (类簇质心不再发生变化)，通常时间长；

(3) RSS 低于某个阈值；

(4) RSS 的降幅低于某个阈值 θ

实践中，条件 (3)、(4) 通常与条件 (1) 结合使用，避免运行时间过长。

K-Means 的收敛性

RSS 在每次迭代中是单调递减的:

$$RSS_k(\vec{\mu}) = \sum_{\vec{x}\in\omega_k} |\vec{\mu} - \vec{x}|^2 = \sum_{\vec{x}\in\omega_k} \sum_{m=1}^{M} (\mu_m - x_m)^2$$

$$\frac{\partial RSS_k(\vec{\mu})}{\mu_m} = \sum_{\vec{x}\in\omega_k} 2(\mu_m - x_m)$$

令上述偏导数为 0，可得:

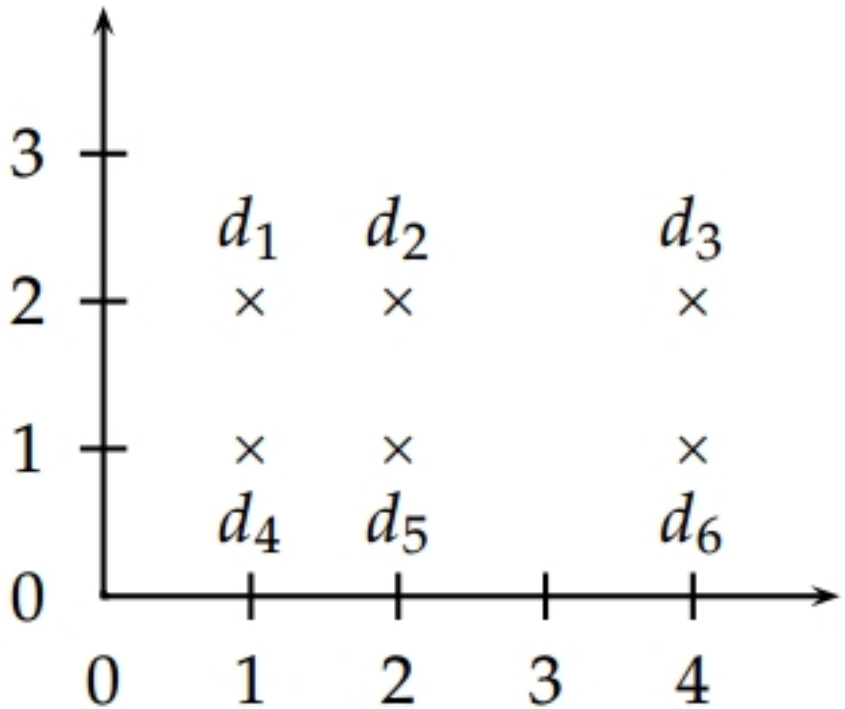
$$\mu_m = \frac{1}{|\omega_k|} \sum_{\vec{x}\in\omega_k} x_m$$

因此，$K\text{-}Means$ 聚类算法是收敛的，但无法保证 RSS 一定达到全局最小值 (global minimum)。当文档集合中存在众多异常点且异常点被选作类簇初始质心时，聚类结果的 RSS 通常只达到局部最小值。

常用的 K-Means 初始类簇质心选择方法

初始质心的选择会影响 $K\text{-}Means$ 聚类算法的输出结果。

对 $K = 2$，选择 d_2 和 d_5 作为初始质心，$K\text{-}Means$ 会收敛到一个次优 (RSS 值没有达到最小值) 的聚类结果 $\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_6\}\}$。选择 d_2 和 d_3 作为初始质心时，$K\text{-}Means$ 会收敛到全局最优 (RSS 值最小) 的聚类结果 $\{\{d_1, d_2, d_4, d_5\}, \{d_3, d_6\}\}$。



常用的 K-Means 初始类簇质心选择方法

选择初始质心的常用启发式方法有：

(1) 避免选择异常点作为初始质心；

(2) 尝试使用多种不同的初始质心进行聚类，选择 RSS 最小的聚类结果；

(3) 使用其他聚类方法 (如，层次聚类) 计算初始质心；

(4) 为每个类簇随机选择 i 个文档，用这 i 个文档向量的质心作为该类簇初始质心，该方法鲁棒性好，适用于多种不同的文档分布。

类别数 K 的选择方法

K-$Means$ 的目标函数 $RSS_{min}(K)$ 是 K 的单调递减函数，当 $K = N$，即类簇数等于文档数时，$RSS_{min}(K) = 0$ 达到最小值，此时每个文档被作为一个类簇，但对实际应用，这种类簇划分并非最优。

一种常用的选择 K 的启发式方法为：递增 K 值，选择 $RSS_{min}(K)$ 值曲线上的拐点对应的 K 值作为类簇数量。

(1) 对每一个 K 值进行 i(如 10) 次聚类实验，选择最小的 RSS 值作为 $RSS_{min}(K)$ 的估计值；

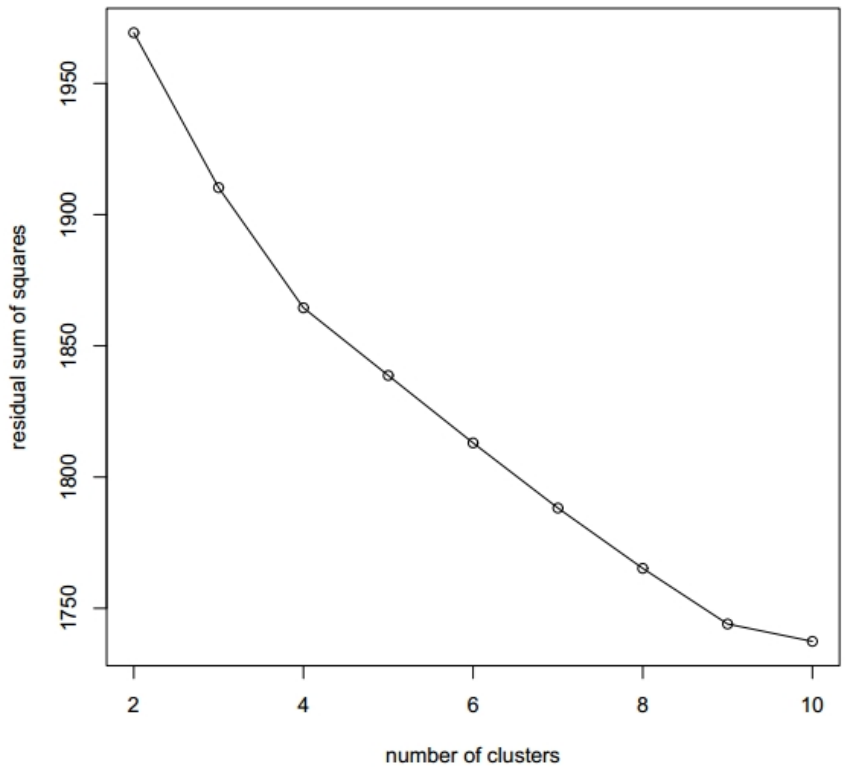
(2) 递增 K 值，绘制 $RSS_{min}(K)$ 值随 K 值变化的曲线图；

(3) 选择 $RSS_{min}(K)$ 曲线上的拐点作为类簇数量候选值；

(4) 结合其他外部约束 (如，对数据的先验知识) 从候选值中确定类簇数量；

类别数 K 的选择方法

下图中，$K = 4$ 和 $K = 9$ 为两个拐点，$RSS_{min}(K)$ 曲线在这两点处变的平坦 (梯度减小)。仍需结合其他条件确定最终类簇数量。



其他确定 K 的方法有赤池信息准则 (Akaike Information Criterion, AIC)，综合考虑了模型失真度 (如，RSS) 和模型复杂度 (如，参数量)。

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 (Dendrogram) 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$(d_1, \ldots, d_N)$

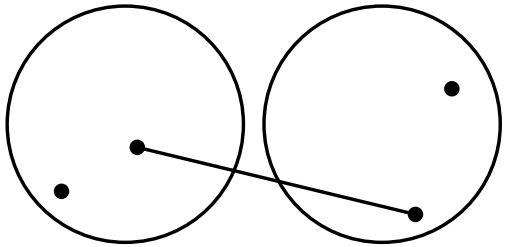
1 **for** $n \leftarrow 1$ **to** N
2 **do for** $i \leftarrow 1$ **to** N
3 **do** $C[n][i] \leftarrow \text{SIM}(d_n, d_i)$
4 $I[n] \leftarrow 1$ *(keeps track of active clusters)*
5 $A \leftarrow []$ *(assembles clustering as a sequence of merges)*
6 **for** $k \leftarrow 1$ **to** $N - 1$
7 **do** $\langle i, m \rangle \leftarrow \arg\max_{\{\langle i,m \rangle : i \neq m \wedge I[i]=1 \wedge I[m]=1\}} C[i][m]$
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

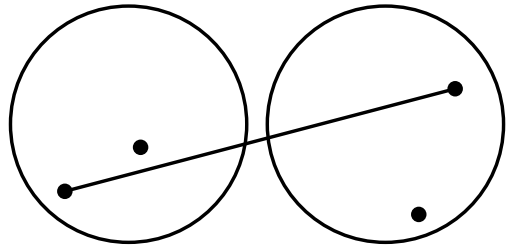
- Line 1-3: 计算一个 $N \times N$ 维的相似度矩阵 C;
- Line 4: 初始化一个 $1 \times N$ 维的 0-1 向量 I，保存可被继续合并的类簇，例如，$I[1] = 1$ 表示第二个文档处于未被合并状态，$I[3] = 0$ 表示第三个文档已被合并到某个类簇中，处于冻结状态;
- Line 5: 初始化一个序列 A，保存类簇合并过程;
- Line 6-12: 合并最相似的类簇，并更新矩阵 C、向量 I、序列 A。

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。

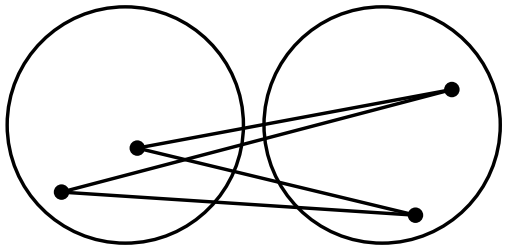
HAC 中的类簇相似度有四种：单链接、全链接、质心法、平均链接：



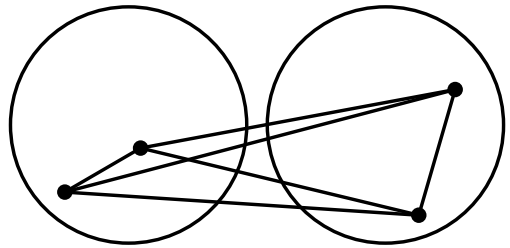
(a) single-link: maximum similarity



(b) complete-link: minimum similarity



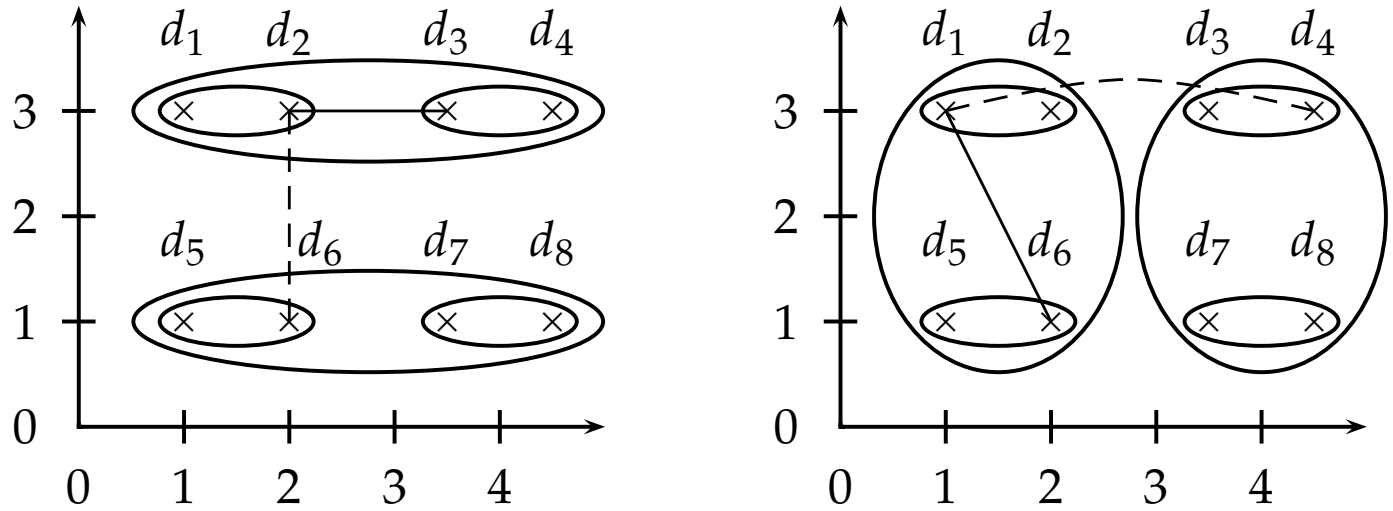
(c) centroid: average inter-similarity



(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。

质心法计算两个类簇质心的相似度：

$$(\frac{1}{N_i}\sum_{d_m\in\omega_i}\vec{d_m})\cdot(\frac{1}{N_j}\sum_{d_n\in\omega_j}\vec{d_n})=\frac{1}{N_iN_j}\sum_{d_m\in\omega_i}\sum_{d_n\in\omega_j}\vec{d_m}\cdot\vec{d_n}$$

平均链接计算两个类簇中全部文档的成对相似度：

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

```
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)
```

外部指标和内部指标

外部指标: 在评估聚类效果时，使用与聚类结果独立的外部信息，通常是数据的真实标签，主要用来衡量聚类结果与真实分类之间的一致性。常见的外部指标包括：

(1) 纯度 (Purity)：衡量每个聚类中占比最大的真实类别的数量；

(2) 标准化互信息: 综合考虑聚类质量和类簇数；

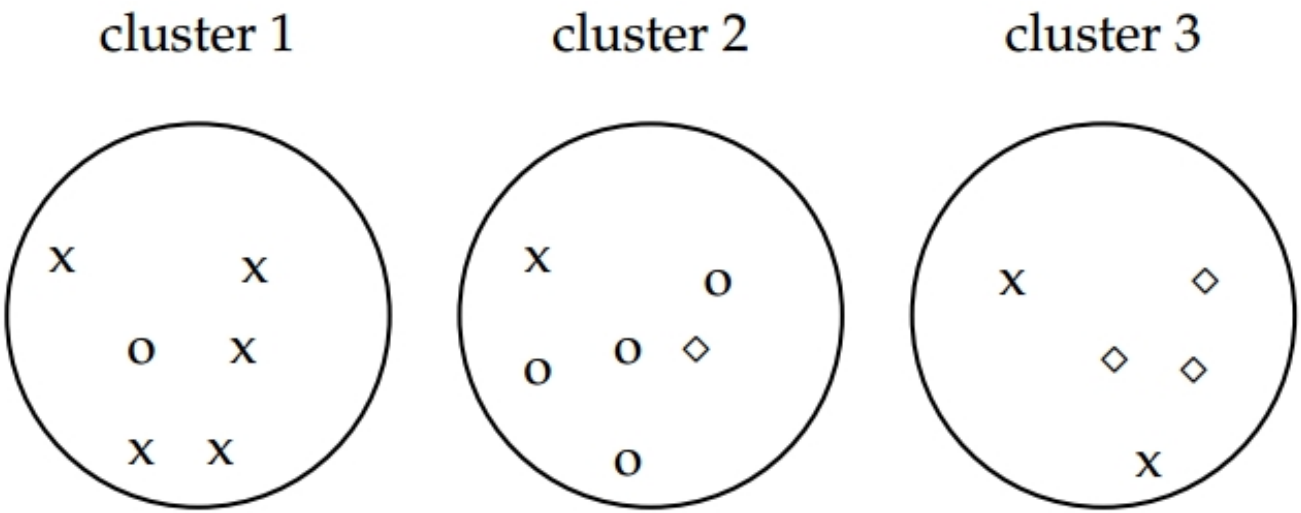
(3) 兰德指数 (Rand Index)：计算所有成对数据点中，聚类分配与真实标签一致的比例；

内部指标: 不依赖于任何外部信息，仅基于数据本身的结构和聚类结果来评估聚类的质量，主要用来衡量聚类的凝聚度和分离度。常见的内部指标包括：

(1) 轮廓系数 (Silhouette Coefficient)：评估每个样本到其聚类内部点的平均距离与到最近聚类的平均距离的比率。

(2) 戴维斯-布尔丁指数 (Davies-Bouldin Index)：基于聚类内的紧密度和聚类间的分离度，较低的值表示更好的聚类效果。

外部指标计算示例



	purity	NMI	RI	F_5
lower bound	0.0	0.0	0.0	0.0
maximum	1	1	1	1
value for Figure 16.4	0.71	0.36	0.68	0.46

图 2: 聚类结果外部评价指标计算示例

聚类纯度 (Purity)

聚类纯度通过检查每个类簇中最多的真实类别成员数来评估聚类结果的纯净度。计算过程为：(1) 对每个聚类，找出数量最多的类别; (2) 将每个聚类的主要类别的数量相加，除以文档集合中的文档总数 N。

$$\text{Purity} = \frac{1}{N}\sum_{k=1}^{K}\max_{j}|\omega_k \cap c_j|$$

K 是类簇数量，ω_k 是第 k 个类簇中的文档集合，c_j 是第 j 个真实类别文档集合，$\max_j$ 表示选取最大交集的操作。

聚类纯度的值范围从 0 到 1，其中 1 表示完全纯净的聚类，每个聚类完全由单一类别的数据点组成。但是，当类簇数与类别数相同时，纯度值也为 1，该度量未能对过多的类簇数施加惩罚。虽然该度量有助于了解聚类与实际类别的对应关系，但它不考虑聚类的结构或形状，也不考虑非主导类别的信息。

标准化互信息 (normalized mutual information, NMI) I

NMI 是一个基于信息论的度量，综合考虑了聚类质量和类簇数量：

$$NMI(\Omega, C) = \frac{I(\Omega; C)}{[H(\Omega) + H(C)]/2}$$

其中，$\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ 表示聚类结果，$C = \{c_1, c_2, ..., c_J\}$ 表示文档的真实类别结果，ω_k 表示类簇 k 中的文档集合，c_j 表示真实类别标签为 j 的文档集合。I 表示互信息，H 表示熵。该度量的分母对过多的类簇数施加了惩罚。NMI 的取值范围为 0-1，越大越好。

互信息 $I(\Omega; C)$ 定义如下：

$$I(\Omega; C) = \sum_k \sum_j P(\omega_k \cap c_j) log \frac{P(\omega_k \cap c_j)}{P(\omega_k) P(c_j)}$$

应用极大似然估计计算：

标准化互信息 (normalized mutual information, NMI) II

$$I(\Omega; C) = \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} log \frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|}$$

信息熵 $H(\Omega)$ 定义如下：

$$H(\Omega) = -\sum_k P(\omega_k) log P(\omega_k)$$

应用极大似然估计计算:

$$H(\Omega) = -\sum_k \frac{|\omega_k|}{N} log \frac{|\omega_k|}{N}$$

标准化互信息 (normalized mutual information, NMI) III

互信息 I 度量已知聚类结果后，对真实类别了解增加的信息量，当每个类簇中的样本点均属于相同类别时，取得最大值，因此，未能对类簇数施加惩罚。

信息熵 H 度量不确定性，通常随类簇数 K 的增加而变大，当 $K = N$ 时，取得最大值 $logN$，因此，可以对过多的类簇数施加惩罚。

兰德指数 (Rand Index, RI)

兰德指数：将聚类过程视为一系列决策，每个决策对应文档集合中的一对文档，因此共有 $\frac{N(N-1)}{2}$ 个决策，正确决策的比例即为兰德指数:

(1) True Positive (TP) 决策表示，两个文档类别标签相同，被聚到同一个类簇中；

(2) True Negativa (TN) 决策表示，两个文档类别标签不同，被聚到不同的类簇中；

(3) False Positive (FP) 决策表示，两个文档类别标签不同，被聚到同一个类簇中；

(4) False Negative (FN) 决策表示，两个文档类别标签相同，被聚到不同的类簇中；

$$RI = \text{Accuracy of Decisions} = \frac{TP + TN}{TP + TN + FP + FN}$$

兰德指数 (Rand Index, RI)

示例中的聚类结果的决策混淆矩阵如下：

	Same cluster	Different clusters
Same class	TP = 20	FN = 24
Different classes	FP = 20	TN = 72

$$RI = \frac{20 + 72}{20 + 24 + 20 + 72} \approx 0.68$$

应用组合数计算混淆矩阵中各分量:

$$TP + FP = C_6^2 + C_6^2 + C_5^2,\ TP = C_5^2 + C_4^2 + C_3^2 + C_2^2$$

基于上述混淆矩阵可进一步计算 F_β 值，$F_1 \approx 0.48$，$F_5 \approx 0.456$。如更希望聚类结果将相似的文档划分到相同的类簇，则取 $\beta > 1$，即，偏好 $Recall$，对 False Negative 施加更大的惩罚。

轮廓系数 I

轮廓系数同时考虑了聚类内的凝聚度和聚类间的分离度。轮廓系数为每个样本点提供了一个度量，用来判断这个点是否被分配到了合适的聚类中。这个系数的值介于-1 到 1 之间，其中接近 1 的值表明样本被很好地分配到了聚类中，接近-1 的值表明样本更适合被分配到另一个聚类中，而接近 0 的值则表明样本在两个聚类的边界上，整体轮廓系数是所有样本点轮廓系数的平均值。单个样本点的轮廓系数计算步骤如下：

1. 计算样本点的凝聚度 a：凝聚度 a 是指一个样本点与其同一个聚类中所有其他点的平均距离，表示了聚类内部的紧密程度。对于类簇 ω_i 中的某个点 x，其凝聚度计算为：

$$a(x) = \frac{1}{|\omega_i| - 1} \sum_{x' \in \omega_i, x' \neq x} d(x, x')$$

其中 $d(x, x')$ 是点 x 和 x' 之间的距离。

轮廓系数 II

2. 计算样本点的分离度 b：分离度 b 是指一个样本点与最近的其他聚类中所有点的平均距离，表示不同聚类之间的分隔程度。对于聚类 ω_i 中的某个点 x，找到与其平均距离最小的其他类簇 $\omega_j\,(j \neq i)$：

$$b(x) = \min_{j \neq i} \frac{1}{|\omega_j|} \sum_{x' \in \omega_j} d(x, x')$$

3. 计算轮廓系数 s：

$$s(x) = \frac{b(x) - a(x)}{\max(a(x), b(x))}$$

$s(x)$ 介于-1 和 1 之间。当 $a(x) < b(x)$（即分离度大于凝聚度），轮廓系数是正的，这表明聚类分配得当；当 $a(x) > b(x)$，轮廓系数是负的，表明 x 可能被分配错了聚类。

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