
统计自然语言

提纲

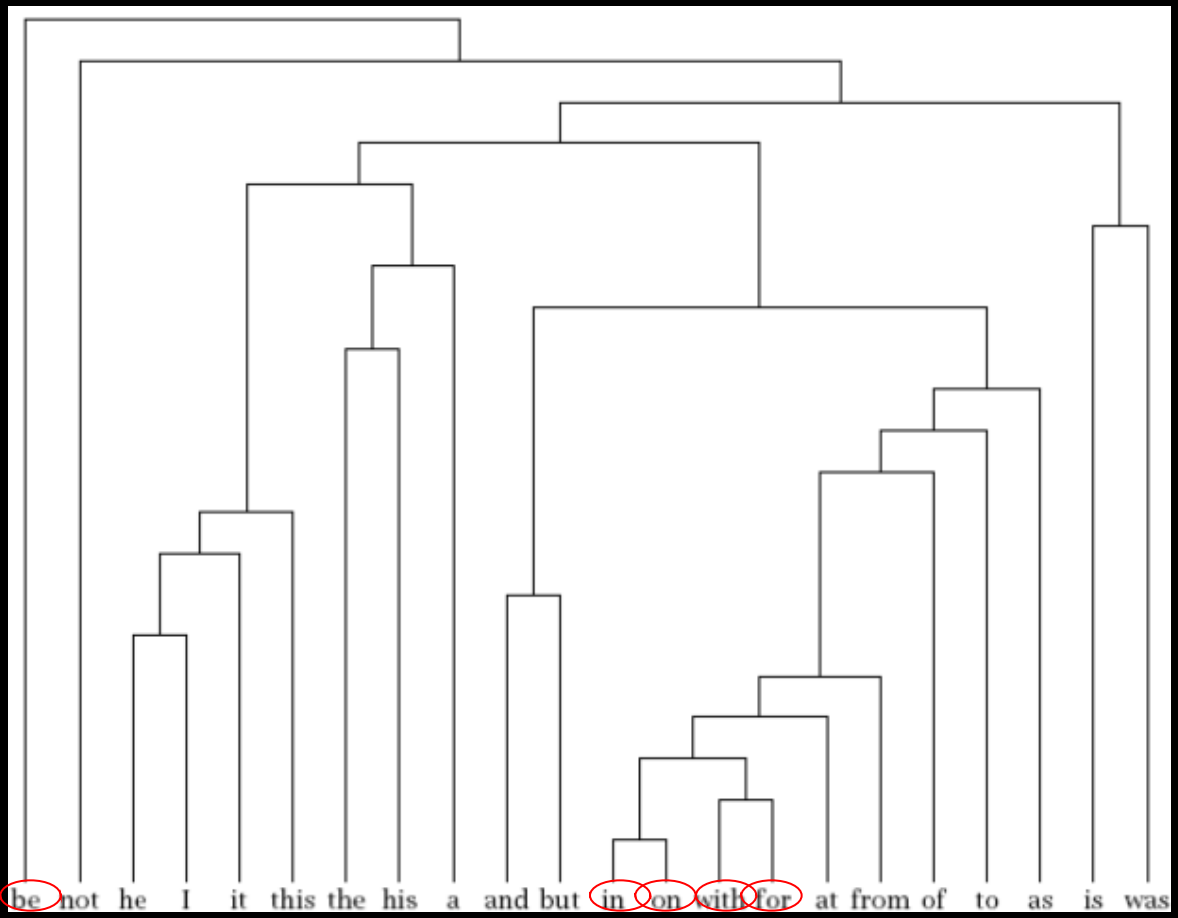
- 聚类概述
  - 用途
  - 种类
    - “软”聚类,”硬”聚类
- 层级聚类
  - 单连通、全连通
  - 平均连通
  - 自顶向下聚类
- 非层级聚类
  - K平均算法
  - EM算法

提纲

- 聚类概述
  - 用途
  - 种类
    - “软”聚类,”硬”聚类
- 层级聚类
  - 单连通、全连通
  - 平均连通
  - 自顶向下聚类
- 非层级聚类
  - K平均算法
  - EM算法

聚类概述

- 聚类算法的目标：
  - 是将一组对象划分成若干组或类别，简单地说就是相似元素同组、相异元素不同组的划分过程。
- 定义：
  - 聚类是一个无指导的学习过程，它是指根据样本之间的某种距离在无监督条件下的聚簇过程。

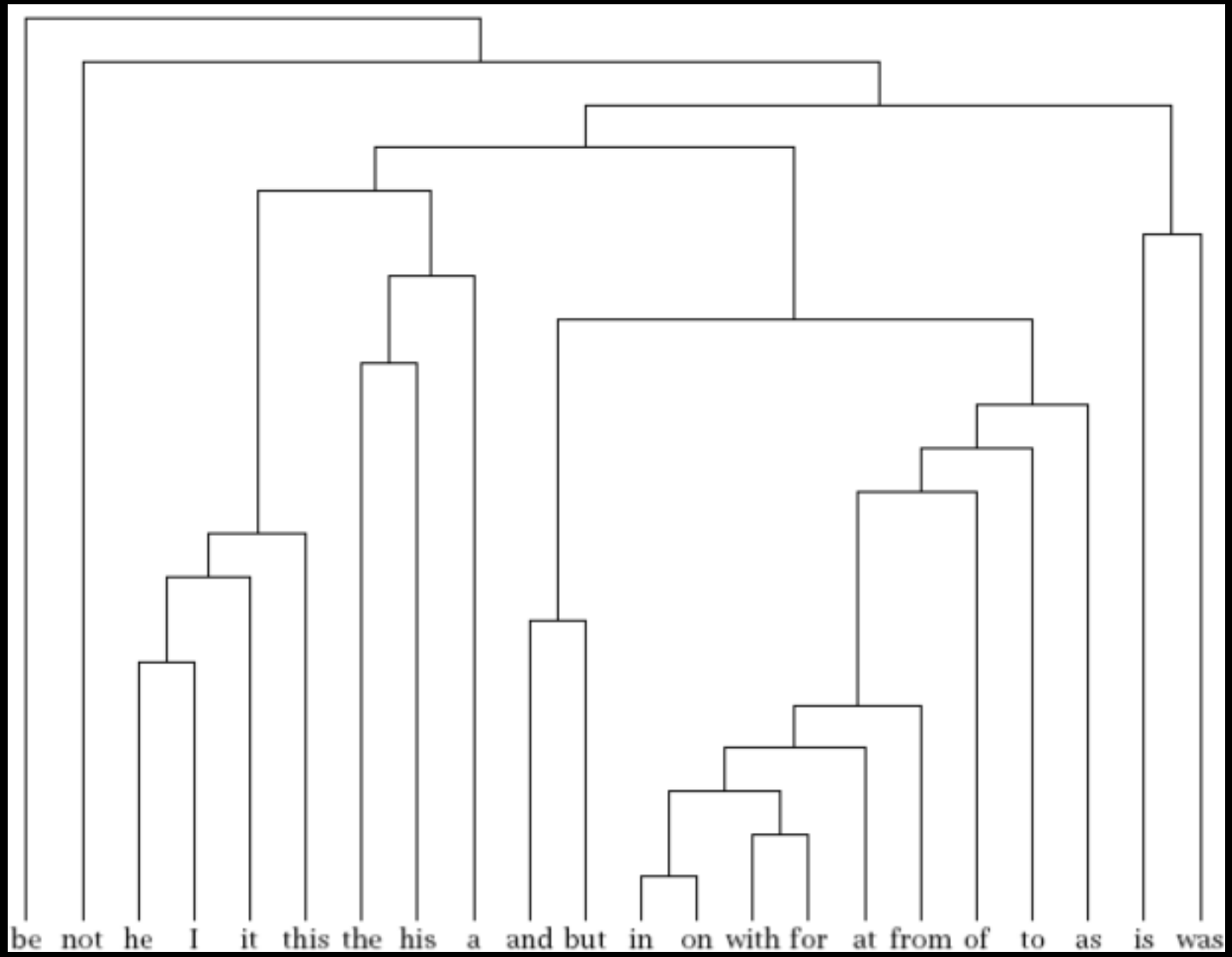

be not he I it this the his a and but in on with for at from of to as is was

聚类概述

- 用途：
  - 在统计自然语言处理中，聚类算法有两个重要的用途：
    - 1.用于试探性数据分析
    - 2.概念一般化

聚类概述

- 用途：
  - 1.用于试探性数据分析
    - 当我们面临一个新问题,并且希望建立一个概率模型或者仅仅是为了理解现象的基本特性时,这是一个首要步骤。
    - 对于不懂英语的人也能通过下面的聚类树图对英文的词性有大致的了解。


be not he I it this the his a and but in on with for at from of to as is was

聚类概述

- 用途：
  - 2.概念一般化
    - 以法英翻译为例，Friday前的介词未知，进行推断。
    - 已有的英文数据:on Sunday, on Monday, on Thursday.
    - 按照语法和语义聚类，Sunday, Monday, Thursday就会被聚到一类，因为它们有相同的上下文模式。
      - Until day-of-the-week, last day-of-the-week, day-of-the-week morning
    - 同类中的元素具有互换性，因此可以推断on Friday的正确性。

聚类概述

- 聚类算法与分类算法的区别:
  - 分类算法是一个有监督的学习过程,它需要对标注数据集合进行训练;
  - 聚类算法则不需要"教师"的指导,不需要提供训练数据,倾向于数据的自然划分,因此被称为无监督的学习或者自动学习.

聚类概述

- 聚类算法的分类：
  - 聚类算法可分为两大类：
    - 层级聚类
    - 非层级聚类

聚类概述

- 层级聚类
  - 每个结点都是父类的一个类；
  - 聚类可以表示成为树图的形式。
- 非层级聚类
  - 类别结构简单；
  - 类别之间的关系没有前者清晰；
  - 是一个迭代过程：
    - 初始聚类
    - 分配样本数据

聚类概述

- 聚类算法的分类：
  - 按照聚类方法不同划分：
    - “硬”聚类；
      - 每个样本只能属于一个聚类集合；
    - “软”聚类；
      - 一个对象可以同时属于几个聚类集合，但是属于各个类别的概率不同；

聚类概述

- "硬"聚类
  - 例：前面的单连通聚类树图所示的聚类。
  - 层级聚类通常都是"硬"聚类；
- "软"聚类
  - 评估单词和某个主题的相关程度时，它体现出来优势。
  - 例：inning和score都是sport类的别中的单词,但是它们的概率分别是0.93和0.65,score属于government的概率为0.12,说明score还和其他类别有关。

提纲

- 聚类概述
  - 用途
  - 种类
    - “软”聚类,”硬”聚类
- 层级聚类
  - 单连通、全连通
  - 平均连通
  - 自顶向下聚类
- 非层级聚类
  - K平均算法
  - EM算法

层级聚类

- 层级聚类算法分为"自底向上"和"自顶向下"两种:
  - "自底向上":
    - 开始时每个对象都被作为一个类别，然后合并两个最相似的类别，直到只存在一个类别为止。
  - "自顶向下":
    - 开始时全体对象作为一个类别，然后每次迭代分割内聚度最小的类别集合，直到每个类别中只有一个对象。
- 在这两类算法中，都要用到相似度函数.

层级聚类

1 Given: a set $X = \{x_1, \ldots x_n\}$ of objects
2 a function sim: $\mathcal{P}(X) \times \mathcal{P}(X) \to \mathbb{R}$
3 for $i := 1$ to n do
4 $c_i := \{x_i\}$ end
5 $C := \{c_1, \ldots, c_n\}$
6 $j := n + 1$
7 while $C > 1$
8 $(c_{n_1}, c_{n_2}) := \arg\max_{(c_u, c_v) \in C \times C} \mathrm{sim}(c_u, c_v)$
9 $c_j = c_{n_1} \cup c_{n_2}$
10 $C := C \backslash \{c_{n_1}, c_{n_2}\} \cup \{c_j\}$
11 $j := j + 1$

- “自底向上”算法
  - （3、4）将每个对象初始化为一个类别；
  - （8）判断最相似的两个聚类；
  - （9）将选出的最相似的聚类进行合并。

层级聚类

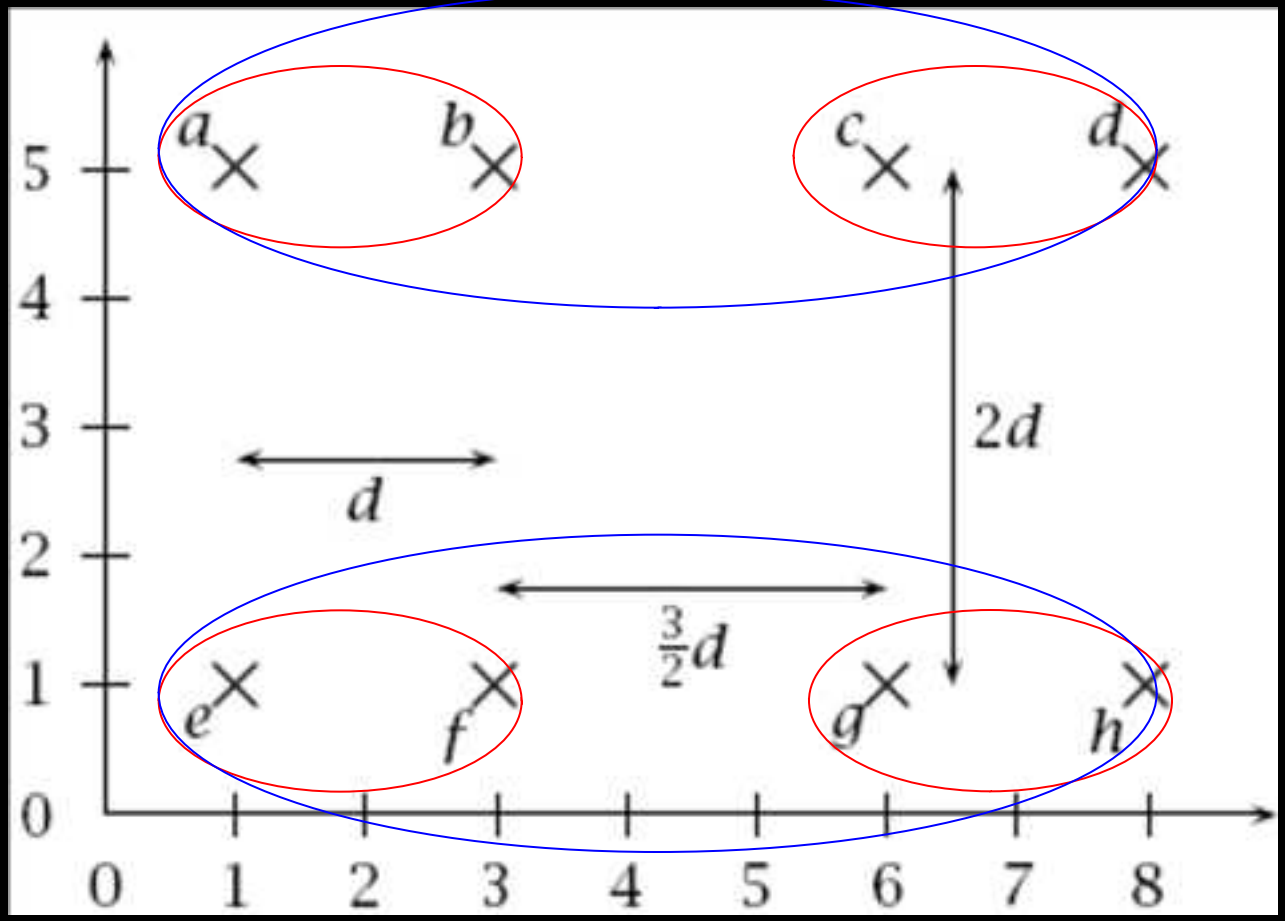
1 Given: a set $X = \{x_1, \ldots x_n\}$ of objects
2 a function coh: $\mathcal{P}(X) \to \mathbb{R}$
3 a function split: $\mathcal{P}(X) \to \mathcal{P}(X) \times \mathcal{P}(X)$
4 $C := \{X\}$ $(= \{c_1\})$
5 $j := 1$
6 while $\exists c_i \in C$ s.t. $|c_i| > 1$
7 $c_u := \arg\min_{c_v \in C} \mathrm{coh}(c_v)$
8 $(c_{j+1}, c_{j+2}) = \mathrm{split}(c_u)$
9 $C := C \backslash \{c_u\} \cup \{c_{j+1}, c_{j+2}\}$
10 $j := j + 2$

- “自顶向下”
  - （4）所有样本做为一个类别；
  - （7）选择最小内聚度的类别；
  - （8）分割最小内聚度的类别集合。

层级聚类

- 三种相似度函数的大概计算原则
  - 1．单连通聚类：
    - 两个集合间最相似样本之间的相似度；
    - 有好的局部一致性；

1．单连通聚类



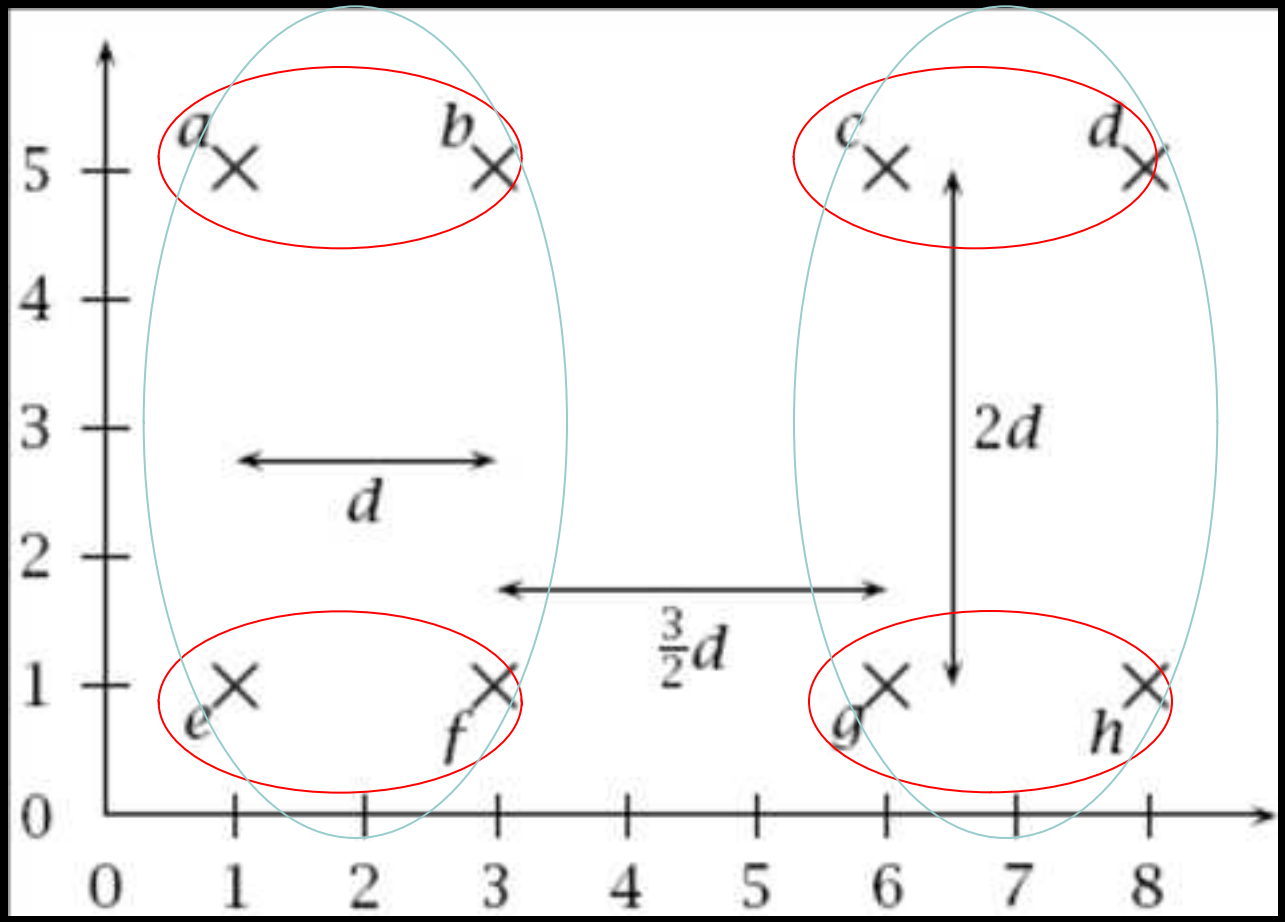
层级聚类

- 三种相似度函数的大概计算原则
  - 1．单连通聚类：
    - 两个集合间最相似样本之间的相似度；
    - 有好的局部一致性；
    - 和最小生成树的方法很类似；

层级聚类

- 三种相似度函数的大概计算原则
  - 2．全连通聚类
    - 两个集合间最不相似样本之间的相似度；
    - 考虑到了全局因素，避免了单连通算法中“拉长”区域的产生；

1．单连通聚类



层级聚类

- 三种相似度函数的大概计算原则
  - 2．全连通聚类
    - 两个集合间最不相似样本之间的相似度；
    - 考虑到了全局因素，避免了单连通算法中"拉长"区域的产生；
    - 假定"内部紧密"比"内部松散"聚类效果好；
      - 例外：夏威夷岛火山；
    - 比较而言,全连通聚类更适合统计自然语言处理的要求；
    - 主要缺点在于它的算法复杂度是$O(n^3)$；

层级聚类

- 三种相似度函数的大概计算原则
  - 3．平均连通聚类
    - 集合内部样本之间的平均相似度;
    - 是上述两种方法的折中方案;
    - 可以替代全连通聚类,它的计算复杂度只有$O(n^2)$;

相似度函数计算原则

- 平均连通聚类
  - 当样本定义在m维空间时，相似度量可以采用余弦法：

$$sim(\vec{x},\vec{y}) = \cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum_{i=1}^{m} x_i \times y_i}{\sqrt{\sum_{i=1}^{m} x_i^2} \times \sqrt{\sum_{i=1}^{m} y_i^2}} = \vec{x}\cdot\vec{y}$$

  - 可以在常量时间内完成平均相似度计算；

相似度函数计算原则

- 平均连通聚类
  - 平均相似度S的定义：

$$S(c_j)=\frac{1}{|c_j|(|c_j|-1)}\sum_{x\in c_j}\sum_{x\neq y\in c_j}sim(x,y)$$

  - $|c_j|(|c_j|-1)$ 为非零相似度的总数

相似度函数计算原则

- 平均连通聚类
  - 算法每次迭代都确定两个集合c_u和c_v，使$S(c_u \cup c_v)$最大；
  - 减少计算量：
    - 先计算：$\vec{s}(c_j) = \sum_{x \in c_j} \vec{x}$，聚类合并时这个值很容易更新；
    - $S(c_j)$的计算可以利用$\vec{s}(c_j)$