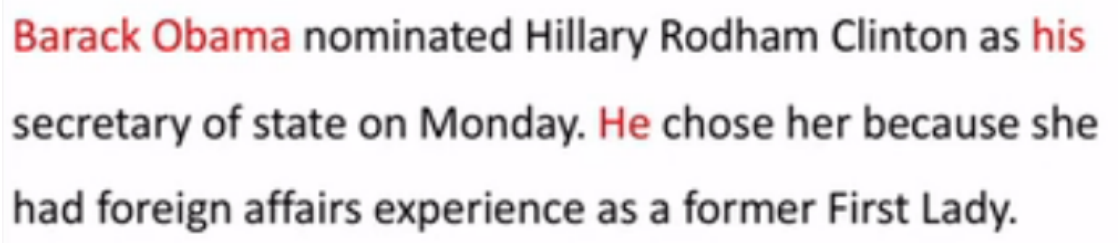
**Lecture16 Coreference Resolution**

---

# Coreference Resolution共指消解

idea：我们有一个文本，找到提到实体的地方，共指消解就是找到提到了同一个实体的那些地方。




应用：
全文理解、机器翻译、对话系统、

## Coreference Resolution in Two Steps

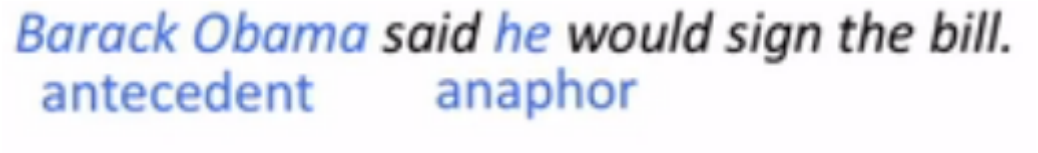
1. 检测提及实体的地方
   mention：指向某个实体的一段文本。
   有三种类型的mention：代词、命名实体、宁次短语
   可以使用其他的NLP系统作为预处理系统来查找mention。对代词而言，它们是言语标记的一部分；对于命名实体使用NER系统；对于名词短语，需要解析器根据句子结构找到名词短语的位置（尤其是选区解析）
   但是存在一些不涉及指代的词，比如
   （Every student、No student、The best donut in the world100、 miles），
   **How to deal with these bad mentions?**
   训练分类器，过滤掉他们或者使所有mentions作为候选mentions，再进行下一步聚类
2. 将这些提及聚类

**Can we avoid a pipelined system?**
训练一个专门用于 mention 检测的分类器，而不是使用POS标记器、NER系统和解析器。甚至端到端共同完成 mention 检测和共指解析，而不是两步。

# Coreference

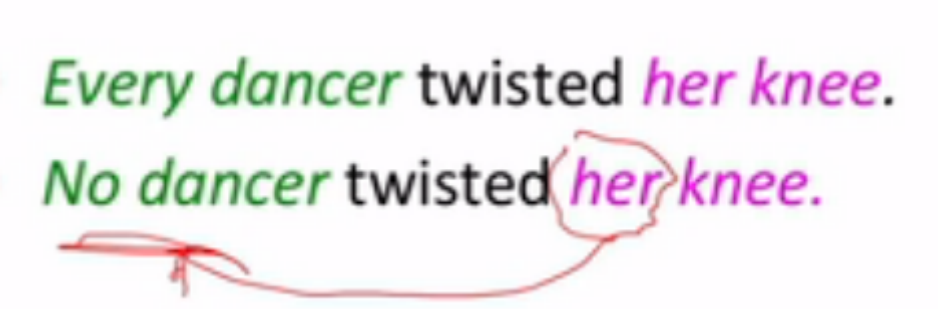
指代是指当两个mention指向同一个实体。
**anaphora**回指：文中的一些术语没有独立的参考，并通过将他们与文中的另一个内容联系

起来来确定他们的参考。

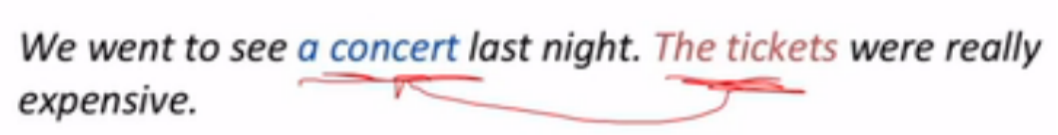



he是一个照应者，可以追溯到Obama


- 前面已经说到了不是所有的名词短语都有指代

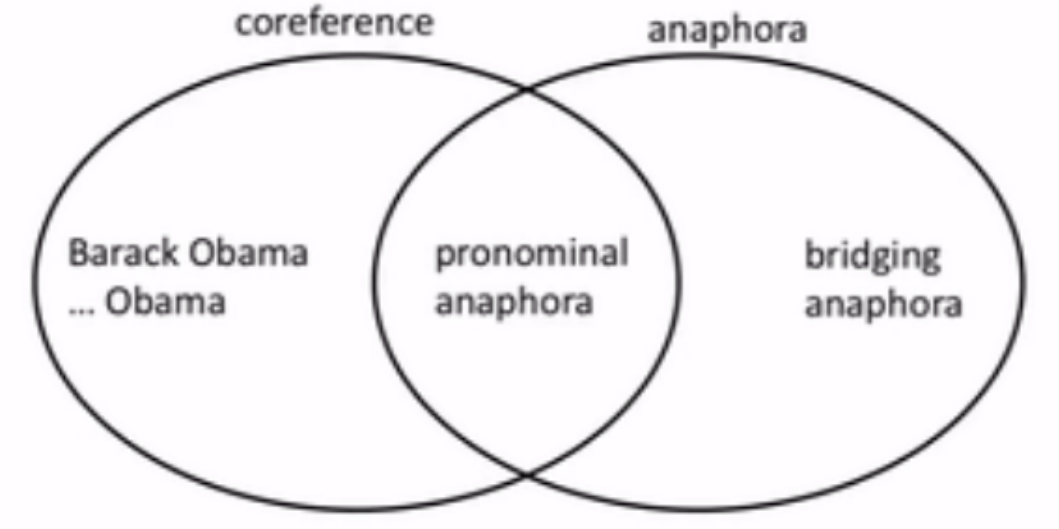



her指向dancer，尽管没有参考仍可以有照应的文本关系




- 这也被称为anaphora关系，但它们不是指代关系，因为音乐会和门票是两个不同的实

体。称为bridging anaphora



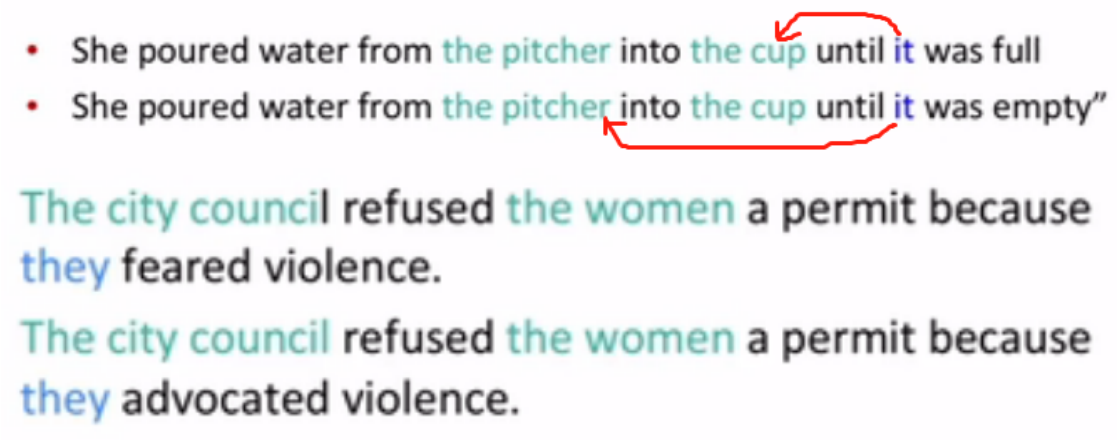

- Cataphora后指

## Coreference Models

- 基于规则的
- mention pair
- mention ranking（最简单）
- 聚类（难以从中获得最佳性能）

**Hobbs’ naive algorithm**
仅用于寻找代词的参考，也可以延伸到其他案例。可以作为共指消解的 baseline

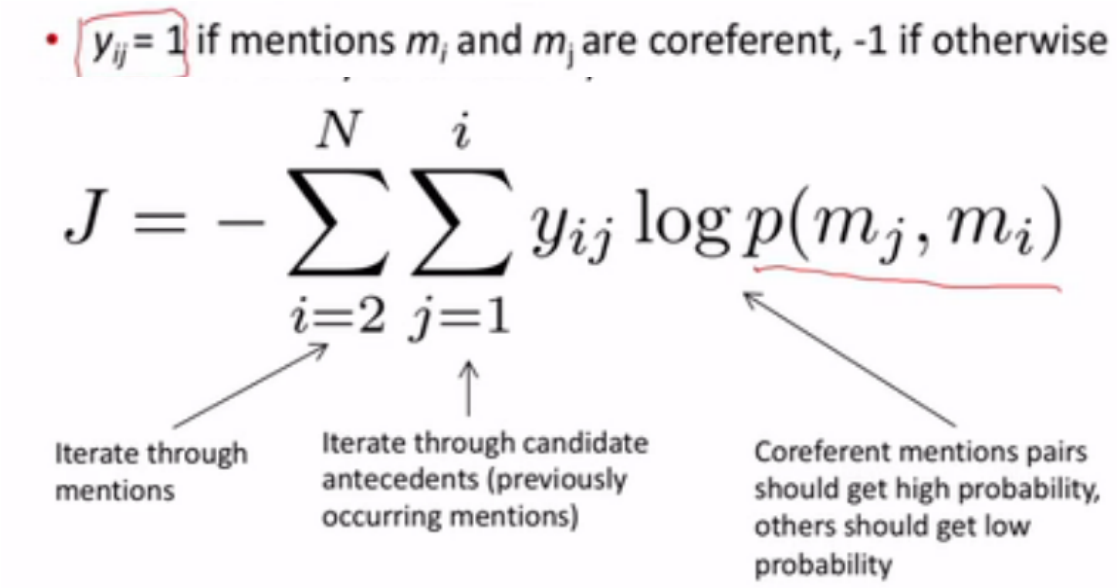
**基于知识的代词指代**




这些例子中的两句话有了相同的句子结构，但不能用Hobbs的方法解决

### mention pair

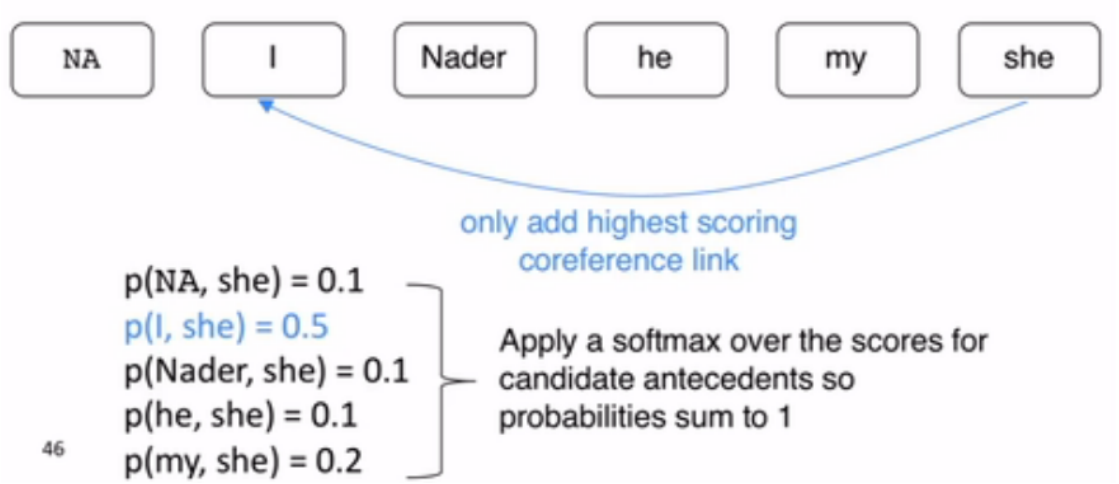
训练一个二进制分类器判断每一对mention是共指的还是非共指的，然后将文本从左到右移动，每当遇到新的mention，根据之前的每一个mention来评估




根据阈值来建立共指连接，使用传递闭包获得集群。但是可能会过度集群，如果有一个共指 连接判断错误，就会导致两个 cluster 被错误地合并了

## mention ranking

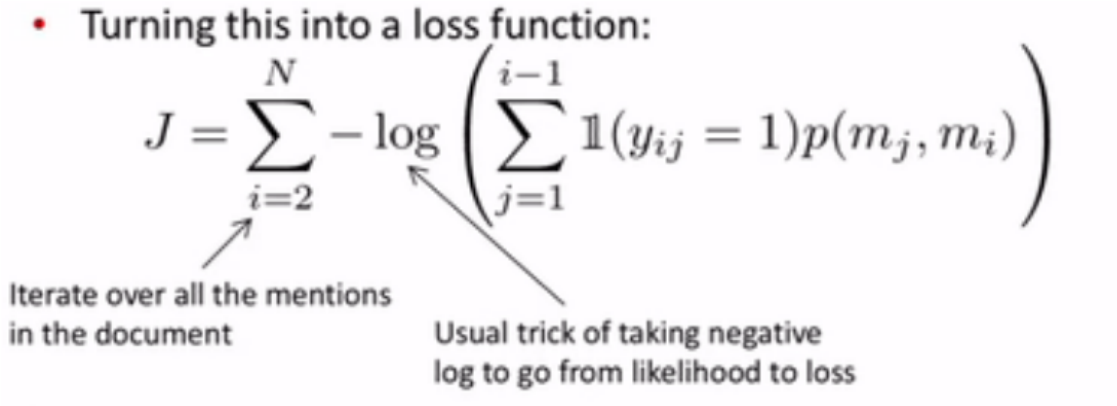
对于每一个mention，试图找到在他之前出现的有关的先行词




I 只能选择NA作为自己的先行词，而she可以选择前面五个作为先行词，分别用softmax计算得分，只添加得分最高的共指连接。

当前mention与它所关联的任何一个候选先行词相关联。将概率最大化：

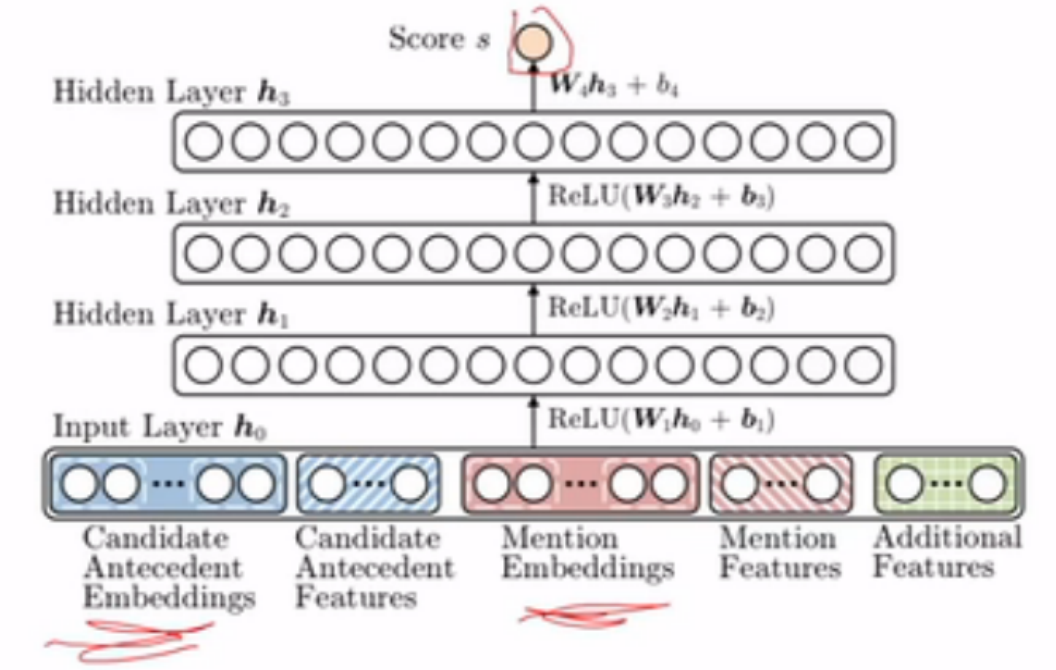
$$\sum_{j=1}^{i-1} \mathbb{1}(y_{ij} = 1) p(m_j, m_i)$$




和 mention-pair 模型几乎一样，除了每个 mention 只分配一个先行词

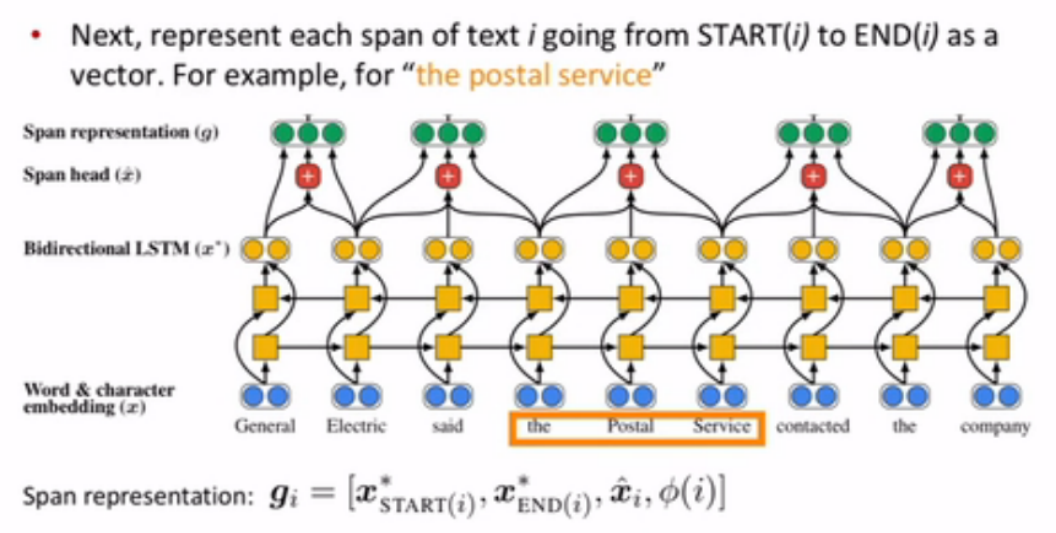
**怎么计算这些概率得分的？**

- 经典做法：有很多特征和基于特征的统计分类器。例如性别、相似语义等
- 神经系统

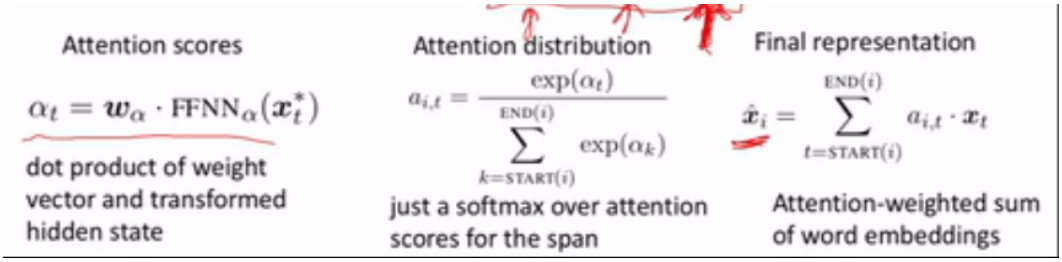



输入层：词嵌入和一些类别特征

- End-to-end Model
  第一步从词开始，对于每一个词，查找他的embedding，还将加入字符级CNN，量他们的连接作为每一个token的表示。
  然后在整个句子中来回运行深度双向LSTM
  接下来，对于每一个子序列，
  将每段文本 i 从 START(i)到 END(i)表示为一个向量



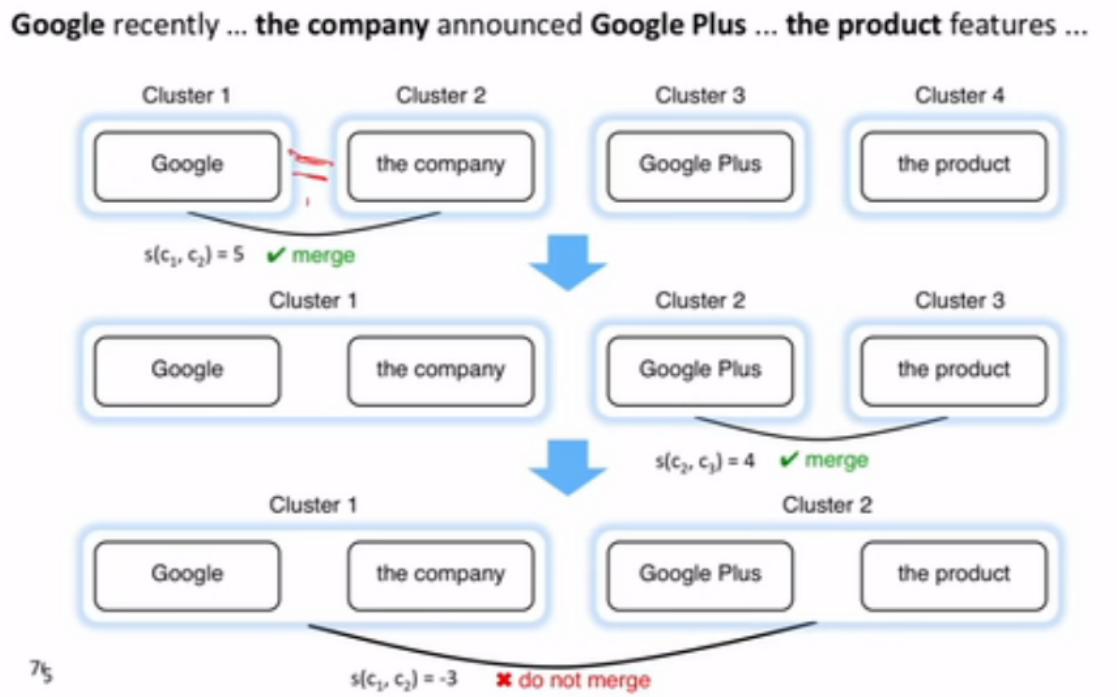

xi 是 span 的注意力加权平均的词向量




最后，为每个 span 对打分来决定他们是不是共指 mentions

## 基于聚类

开始时每个 mention 在它自己的单独集群中，每一步合并两个集群

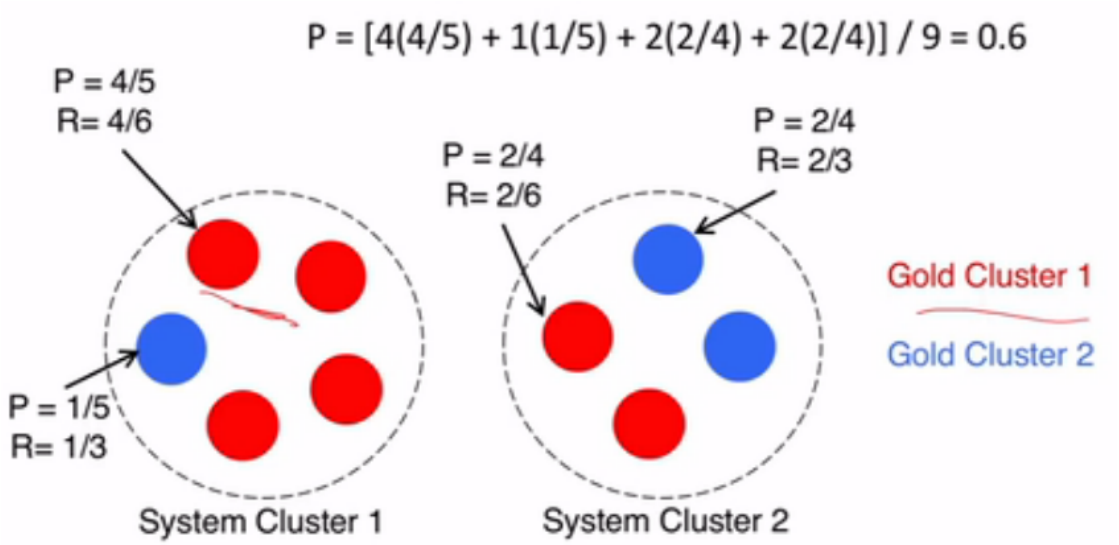



当没有集群要合并时，算法停止。

mention pair很难判断，但是cluster pair比较容易

# Coreference Evaluation

MUC, CEAF, LEA, B-CUBED, BLANC
例如，B-CUBED：
对于每个 mention ，计算其准确率和召回率
然后平均每个个体的准确率和召回率




在集群下，会有很高的精确度，但召回率很差；如果超过集群，会得到很好的召回率，但精确度很低。