中文共指消解中的聚类全局优化*

刘未鹏[1] 周俊生[2] 黄书剑[1] 陈家骏[1]

1. 南京大学计算机软件新技术国家重点实验室, 江苏 南京 210093

2. 南京师范大学计算机科学与技术学院, 江苏 南京 210097

E-mail: liuwp@nlp.nju.edu.cn

摘 要: 共指消解的主流框架分为二元分类和等价类划分两个步骤，围绕第二个步骤进行的全局优化是主要的研究方向之一。本文结合共指消解问题本身的特点提出了一种基于最小化决策错误的损失函数，并利用一种自底向上的聚类模型对其进行优化，此外提出另外两种贪婪聚类模型，在 ACE 中文语料上的实验显示三种聚类模型的效果都优于 baseline 系统，其中自底向上的聚类效果更明显。

关键词: 共指消解，全局优化，聚类，损失函数

Global Optimization Based on Clustering for Coreference Resolution

Liu Weipeng[1], Zhou Junsheng[2], Huang Shujian[1], Chen Jia-Jun[1]

1. State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China

2. Department of Computer Science, Nanjing Normal University, Nanjing 210097, China

E-mail: liuwp@nlp.nju.edu.cn

Abstract: Coreference resolution typically consists of two phases: pairwise classification and partitioning. For the second phase, current research focuses on global optimization techniques. We propose a loss function that aims at minimizing decision errors and propose a bottom-up clustering model and two other greedy clustering models to optimize it. Our experimental results show good performance boosting.

Keywords: Coreference resolution, Global optimization, Clustering, Loss function

1 引言

当前，基于机器学习的篇章共指消解方法一般分为两个步骤[1,2,3]。第一步，使用统计机器学习方法对 Mention Pairs （即一对对的 Mention）进行二元分类，Mention Pair 作为统计学习的实例，其特征包括：词汇、距离、字符串相似度、性别、单复数、语义类别等等。由于共指消解本身特质决定了需要大量的信息才能使之成为适定性的、可解的问题，目前提出的特征总体上来说还是偏向于信息匮乏，并不足以准确判定任一对 Mention 之间的共指与否的关系。一方面，进一步引入深层语义固然能够有望提升共指消解的准确性，然而目前对于句子的深层语义的分析还很难办到，这就使之成为了一个技术上的瓶颈。然而另一方面，在目前既有的特征抽取之下，我们也并非已经利用完了所有抽取出来的信息，事实上，经典

*基金项目：国家自然科学基金项目（60673043）、国家社科基金（07BYY0）、江苏省高校自然科学基金（07KJB520057）。刘未鹏，硕士研究生，主要研究领域为自然语言处理；周俊生，博士，副教授，主要研究领域为自然语言处理、机器学习和信息抽取；黄书剑，博士生，主要研究领域为自然语言处理、机器学习；陈家骏，教授,博士生导师，主要研究领域为自然语言处理、机器翻译和软件工程。

的二元分类缺乏全局信息的指导，将视野仅仅局限于一对 Mention 上，不顾其周边可能有关系的 Mention 上携带的信息，这里的信息缺失导致了分类性能的下降。对此一种被证明很有效的解决方案是将传统的二元分类拓展为 NP-Cluster[4]，这就扩充了可以利用的特征信息，使得分类性能得到了一定程度的提高。

基于机器学习的共指消解的第二个步骤是等价类划分。得到了 Mention Pair 的共指概率之后如何构造最佳的共指链，目前这方面的工作有文献[5]中的 BestCut 方法和文献[6，7]中的整数线性规划方法以及文献[8]中的 Bell 树模型。但它们存在两个问题：1) 损失函数的指定比较随意。2) 使用的最优化模型（如文献[5]使用的图分割，以及 K-means 家族的聚类算法）也倾向于普适的、问题无关的模型，并没有充分结合共指消解问题本身的特质。

本文提出了一种基于最小化决策错误的损失函数，并利用类似文献[9]中提到的自底向上的聚类方法对其进行最优化，与文献[5]不同的是，我们给出了一种与损失函数匹配的，自然的终止条件。我们的实验结果显示性能得到了明显的提高。此外我们还提出了另外两种贪心聚类方法并得到了近似的结果。

2 一种基于最小化决策错误的损失函数

我们的损失函数建立在这样一种假设之上：

假设 1：二元分类的准确程度已经相当高了（一般超过 90%），而全局优化方法则应该尽量利用那些正确的二元的分类结果来纠正那些错误的分类结果。

从某种意义上说，就是要用多数的（正确）决策纠正少数的（错误）决策。

基于这样的理念，我们提出一种基于最小化决策错误的损失函数：假设我们的聚类结果得出了 K 个独立聚类，我们用集合 C_{ij} 来表示聚类 i 和聚类 j 之间所有的边，用集合 C_{ii} 表示聚类 i 内部的所有的边，e_{ij} 表示聚类 i 和聚类 j 之间的一条边，e_{ii} 则表示聚类 i 内部的一条边。使用 P(e)来表示某条边的权重，即两个 Mention 之间的共指概率。

我们试图寻找这样的聚类，使得：

$$\text{Argmin Q ; where Q} = \sum_{i,j;i\neq j}\ \sum_{e_{ij}\in C_{ij};P(e_{ij})>0.5} P(e_{ij}) + \sum_{i}\ \sum_{e_{ii}\in C_{ii};P(e_{ii})<0.5} \bar{P}(e_{ii})$$

即，我们既要利用二元分类决策中的多数者意见中含有的信息来纠正少数错误的分类，又要使得总体上尽量少地违反原先二元分类器的决策。

传统的最优化目标函数或损失函数允许一个 Mention Pair 的共指概率用以代偿另一个 Mention Pair 的共指概率，只要聚类（等价类）内部的总和最小或最大即可，这就将聚类内部所有边的权值都混同起来考虑了，某些权值小的可以由某些权值大的代偿，反之同理。而我们这里的损失函数则重点考虑那些被“错误”地划分了的 Mention Pair，而不是所有的 Mention Pair：如果一个 Mention Pair 在二元分类器眼中的共指概率是大于设定阈值的，那么二元分类器就将其看作是共指的，考虑到二元分类器的结论的正确性较大，那么在划分共指链的时候如果偏偏将这个 Mention Pair 没有划分在同一个等价类中的话，就被视为一个损失，这个损失的大小取决于二元分类的结果，二元分类器认为这对 Mention 共指的概率越大，错误划分的代价相应也就越大。

此外，这个损失不可以由被“被正确地”划分了的 Mention Pair 来“弥补”，例如有另一对 Mention 在二元分类器中的分类结果是“共指”，而且在划分等价类的时候也的确被划分至同一个等价类了，并且和刚才提到的前一个 Mention Pair 属于同一个等价类，那么它的

正确划分并不能“弥补”刚才那对 Mention 的错误划分。我们的损失函数通过滤掉那些被正确划分的 Mention Pair 来实现这一点。

3 自底向上的聚类

文献[10]中提出一种自底向上的聚类方法：初始的时候每个节点属于它们自身的聚类，算法每次寻找两个聚类合并，这个合并必须是所有可能的合并中使得目标函数下降最多的合并。当所有的节点都被合并到一个聚类中时，算法的第一阶段便宣告结束，这个时候我们便得到了一个聚类的层次结构图，在得到的聚类层次体系中停在不同的层次上得到的聚类数目是不一样的。如图 1 所示：

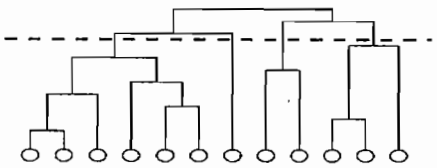

图 1 选择一个使得 Q 最小（最大）的聚类层次

在图 1 中，整个图被切分为 4 个聚类。

文献[10]中提到这一思路可以被泛化用作一类广泛的问题，其中待聚类的问题可以看作一张图，各节点之间以带权值的边连接起来，这正是我们的问题的模型。只要将[10]中的目标函数换成我们自己的目标函数即可。

因此，我们的算法描述如下：

1) 初始化：每个 Mention 作为一个单个的聚类。
2) 不断寻找能够使得 Q 递降最多的合并（梯度下降）。（这里利用类似文献[10]中的自底向上的聚类方法）
3) 在得到的聚类层次中找到那个使得 Q 最小的聚类层次。

需要注意的是，在不断合并两个聚类的过程中，可能出现这样的情况：没有任何一对合并能够使得 Q 减小，这时可以停止凝聚过程，终止算法。也可以继续凝聚，选择使 Q 增大最少的合并即可，但这是没有必要的，因为如果后面 Q 还有减小的可能性，即如果后面存在一个合并$<C_1, C_2>$能够使 Q 减小的话，要么 C_1 和 C_2 与刚才使 Q 增大的合并不相干，这就与“当时选不出使 Q 减小的合并”这个前提相矛盾。要么 C_1 或 C_2 是刚才使 Q 增大的合并的结果类，这时不妨设 C_1 合并自 U_1 和 U_2，那么既然 C_2 和 C_1 合并之后能够使 Q 减小，就意味着 C_2 和 U_1，与 C_2 和 U_2 中至少有一对合并能够使 Q 减小（否则 C_1 和 C_2 合并就会使 Q 增大），但这就违反了最初的前提，即没有合并能够减小 Q。

该算法有几个良好的性质：

1) 每次都选择最可能共指的一对 Mention 进行合并，属于非常保守的合并策略。
2) 并非自左向右地合并，而是全局选择最优合并。
3) 前面合并起来的等价类由于含有多个 Mention，因此内部信息比单个 Mention 要丰富，在判断接下来的合并与否的时候能够提供更可靠的决策信息。

4 Family-Vote 聚类

以上方法是全局聚类，在建立聚类层次的时候算法能够访问到全局信息。然而实际上，

我们注意到篇章中的共指链呈小簇分布，各个共指链互相之间没有联系，共指链之间则联系紧密。因此我们可以发现，当其中一个共指链得出来了之后，对其它共指链的划分便与它完全没有关系了，可以将其从图当中抹去，避免对剩下来的聚类过程产生干扰。

该方法基于的假设和文献[4]中的一样，只不过将操作过程从分类阶段移到了聚类阶段。传统基于二元分类的共指消解具有信息匮乏性，Mention Pair 携带的上下文信息往往是不够的，在不够的时候，应该求助于与这个 Pair 联系最紧密的那些周边 Mention 所携带的信息来完成正确的消解，后者的信息能够构成很好的补充。为此我们提出两种基于投票的聚类方法。

这两种方法都基于这样一个假设：

假设 2： 一个 Mention Pair 携带的信息可能不足，而能够对其进行补充的信息则存储于跟它共指概率最高的周边 Mention 中。

基于这一假设，我们提出一种凝聚式的 Family-Vote（记为 Family-Vote(1)），其执行流程如下：

1) 选择共指概率最高的一对 Mention 作为初始聚类。
2) 在其余所有 Mention 中寻找 Acceptance（下文定义）最高的 Mention 加入这个聚类。
3) 重复这个过程。直到再也找不到 Acceptance > 0 的 Mention，该 Cluster 便构造结束。
4) 重复刚才的步骤构造剩下的 Clusters。

算法中的 Acceptance = CostOfNotLinking - CostOfLinking

其中

$$\text{CostOfNotLinking} = \sum_{m_i \in C} -\log^{1-p\langle i,k\rangle}$$

$$\text{CostOfLinking} = \sum_{m_i \in C} -\log^{p\langle i,k\rangle}$$

这里 C 代表已经建立的聚类，m_i 代表聚类中的任一 Mention，k 代表待判断是否应该加入 C 的 Mention。$p\langle i,k\rangle$ 代表 Mention i 和 Mention j 之间的共指概率。

我们可以沿用文献[5]中的例子来描述该算法的关键流程：

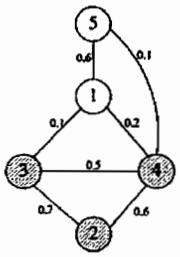

图 2 凝聚式 Family-Vote 的算法执行图示

图 2 对应的文本为：**Mary**$_1$ has **a brother**$_2$ **John**$_3$. **The boy**$_4$ is older than **the girl**$_5$ 。其中 3 和 5、1 和 2 之间的权值因为太小被删除，以简化图片。为简化下文的描述，我们将原图 2 和 4 之间的权值从 0.5 调整为 0.6（不失一般性）。

根据前文的算法描述，首先我们确定 3 和 2 之间的共指关系，因为它们的共指概率最大，然后我们在剩下的节点中寻找被 3 和 2 构成的聚类接受程度最高的节点。

根据前文 Acceptance 的定义可见：Acceptance(4) >0 ，Acceptance(1) < 0 ，

Acceptance(5) < 0 。那么节点 4 被归并入 2 和 3 构成的聚类，并且该聚类关闭并从整个图中剥离，因为没有其他节点能被接受。然后我们再次从剩下的节点中寻找共指概率最大的 Mention Pair，即 1 和 5。以此类推。

与凝聚式的 Family-Vote 类似，我们还提出一种分裂式的 Family-Vote（记为 Family-Vote(2)），分裂式的 Family-Vote 算法的重点在于寻找那些被“错误地”二元分类的边（利用其周边 Mention 所携带的信息）。其主要思想如下：

对于任意一条位于两个 Mention 之间的边，其权值是共指消解第一阶段的二元分类器给出的，这里由于二元分类器的只考虑局部上下文的性质，可能会给出错误的结果，而从前面的分析可以看出，这个错误可以通过考虑周边的 Mention 来得到修正，例如 A 和 C 的共指概率被判定为 0.5 左右，这可能是因为 A 和 C 上的特征较少，所以机器学习算法很难判定，但如果我们考虑和 A 强共指的所有 Mention（记为 S1，以及和 B 强共指的所有 Mention（记为 S2），可见 S1 和 S2 上携带的信息理应大于单单 A 和 C 这两个单独的 NP，我们通过考察 S1 和 S2 是否应该合并来判断是否应该合并 A 和 C。

算法梗概如下：

1) 对每一个 <Mention_i, Mention_j> 对，我们寻找到与 Mention_i 一阶共指（即直接共指，而非通过传递关系共指）的所有 Mention （记为 S_i），同时也找到与 Mention_j 一阶共指的所有 Mention（记为 S_j），计算：

$$\text{CostOfNotLinking} = \sum_{m_p \in S_i; m_q \in S_j} -\log^{1-p\langle p,q\rangle}$$

$$\text{CostOfLinking} = \sum_{m_p \in S_i; m_q \in S_j} -\log^{p\langle p,q\rangle}$$

Acceptance = CostOfNotLinking – CostOfLinking 。若 Acceptance > 0 ，则切断 Mention_i 和 Mention_j 之间的边，否则保留这条边。

2) 对经过上一步处理之后的图求出各连通分量。

简单地说，该算法就是让与 Mention_i 关系紧密的 Mention 和 Mention_j 关系紧密的 Mention 来参与评估 Mention_i 和 Mention_j 的关系。如果最终综合后的结果大于一个阈值，就说 Mention_i 和 Mention_j 是共指的，否则则切断 Mention_i 和 Mention_j 的连接。

值得注意的是，这里如何寻找所有与 Mention_i 一阶共指的所有 Mention 是一个较困难的任务，如果单单设定所有与 Mention_i 共指的概率大于 0.5 的 Mention 的话，可能会导致一些被错误划分为与 Mention_i 共指的 Mention（实际与 Mention 不共指）被考虑进来，这些 Mention 会导致 S_i 和 S_j 之间的“排斥性”增大，从而出现聚类过小的情况，为此我们可以设定一个阈值，譬如设定与 Mention_i 共指概率大于 0.8 的 Mention 才被认为是 S_i 的成员，这就一定程度上保证了我们能够利用的信息的有效性，这个阈值设定得越高，利用的周边信息越精确，然而也同时导致能够被利用的信息越少。

5 实验设计

首先我们进行了特征选择，选出一个合适的特征集。然后我们建立了实验中所使用的语料库，以及基于 link-first 和 link-best 的 Baseline 系统，最后我们比较并分析了实验的结果。

5.1 特征选择

我们选择目前被共指消解系统普遍采用的一些特征：如两个 Mention 之间的距离特征、性别信息、单复数信息和语义类别信息等。表 1 列举了这些特征：

表 1 实验中使用的特征

(1)	距离特征：它可能的取值有 0、1、2、……，距离特征就是获取 i 和 j 之间的距离。
(2)	i-代词特征：它的取值类型为布尔型。如果 i 是一个代词，就返回 true；否则就返回 false。代词包括反身代词（自己）和人称代词（我、你、他）。
(3)	j-代词特征：它的取值类型也是布尔型。如果 j 是一个代词（如上所述），那么返回 true；否则返回 false。
(4)	字符串匹配特征：它的取值类型是布尔型。我们仅对 Mention i 和 j 中心词的字符串进行匹配，如果两者匹配，就返回 true；否则返回 false。
(5)	指示性的 Mention 特征：它的取值类型是布尔型。以指示词“这”、“那”、“这些”或“那些”引导的 Mention 就是指示性的 Mention。
(6)	单复数一致性特征：它的取值类型是布尔型。如果 Mention i 和 j 的单复数信息一致就返回 true；否则返回 false。
(7)	语义类别一致性特征：它可能的取值有 true、false 或 unknown 三种。如果 Mention i 和 j 的语义类别一致，那么特征值就取 true；如果不一致，特征值就取 false；如果其中任何一个的语义类别为“unknown”，那么就比较这两个 Mention 的中心词。
(8)	性别一致性特征：它可能的取值有 true、false 或 unknown 三种。
(9)	专有名词特征：它的取值类型是布尔型。一般来说，人名、地名和机构名这样的命名实体类型是专有名词。如果 i 和 j 都是专有名词，那么返回 true；否则返回 false。
(10)	同位语特征：它的取值是布尔型。如果 Mention i 和 j 之间具有同位语的关系，则返回 true；否则返回 false。

5.2 实验数据与评测指标

我们使用 2005 年 ACE 评测的训练数据作为实验数据。ACE 评测是指由美国国家标准技术研究院(NIST)组织的自动内容抽取(Automatic Content Extraction) 评测，ACE 语料是目前唯一可公开利用的中文共指标注语料。我们从 ACE 中文语料中随机选择 229 篇文档组成了训练语料，69 篇文档组成了测试语料。

为了评测实验结果，本文采用MUC-6中所定义的召回率(*Recall*) 、准确率(*Precision*) 和 *F*值(其中计算*F*值中的β参数取为1)作为评测指标[11]。

5.3 Baseline 系统

我们的 Baseline 系统使用了最近优先策略(link-first)，最近优先策略选择前文离指示语最邻近的置信度大于 0.5 的候选先行语作为最终先行语。以及对最近优先策略的改进：最佳优先策略(link-best)，最佳优先策略则是从置信度大于 0.5 的候选先行语中选择置信度最大的作为最终先行语。和文献[4]一样，我们在二元分类阶段也使用 C4.5。表 2 显示了该 Baseline 系统在我们的语料上的实验结果：

表 2 Baseline 系统的性能

Approaches	Precision(%)	Recall(%)	F(%)
link-first	77.56	75.54	76.54
link-best	78.42	75.36	76.86

5.4 实验结果及分析

我们分别使用了上文提到的三种算法在同样的语料库上进行了实验，表 3 给出了实验的结果对照：

表 3 三种聚类算法的共指消解实验结果

Approaches	Precision(%)	Recall(%)	F(%)
Bottom-up	77.91	78.06	77.98
Family-Vote(1)	77.59	78.01	77.80
Family-Vote(2)	77.60	77.89	77.74

从实验结果我们可以看出，三种聚类方法与 Baseline 相比都有明显的提升。其中基于自底向上的聚类的算法提升约为 1.1%，另两者在 0.8%、0.9%左右。考虑到基于自底向上的聚类算法在聚类的过程中一直能够利用全局信息，而 Family-Vote(1)和 Family-Vote(2)则利用局部信息，但后两者的准确度和前者相差并不大，我们认为这是由共指消解问题的本身的特质所决定的，共指消解中的实体表述或名词短语往往呈现小簇集中分布，其中一个实体内部往往有几个关键的，富含信息的节点，最先考虑高共指概率的贪心聚类算法能够有效地利用这些节点的信息丰富性，而当一个聚类已经被识别出来之后，它对于剩下的节点的聚类结果便不会有太大的影响，因此 Family-Vote(1)这样通过在一个图中依次识别出聚类的算法也能够得到与全局聚类相近的结果。

6 结束语

全局优化方法主要针对的是如何充分利用上下文信息来辅助共指消解。文献[4]中提到的一种基于 NP-Cluster 的方法，该方法工作在二元分类阶段，其核心思想是认为 Mention Pair 携带的信息不足，所以它试图通过从周边的 Mention “借”来信息辅助分类；我们将其想法延伸到后期的聚类阶段，提出一种基于最小化决策错误的损失函数，并且使用了[10]中提到的一种自底向上的聚类方法进行聚类，此外提出另两种贪心聚类方法，实验结果显示该方法在全局优化阶段对结果产生了明显的提升。

参 考 文 献

[1] W. M. Soon, H. T. Ng, and D. Lim. A machine learning approach to coreference resolution of noun phrases. Computational Linguistics. 2001，27(4):521–544.

[2] 庞宁, 杨尔弘. 基于最大熵模型的共指消解研究. 中文信息学报，2008，22(2).

[3] 李国臣，罗云飞. 采用优先选择策略的中文人称代词的指代消解. 中文信息学报, 2005, 19(4)

[4] Yang, X., G. Zhou, J. Su, and C. L. Tan. An NP-Cluster Based Approach to Coreference Resolution. Proceedings of the 20th International Conference on Computational Linguistics, Geneva, Switzerland. 2004, pp. 226-232.

[5] C. Nicolae and G. Nicolae. Bestcut: A graph algorithm for coreference resolution. In: D. Jurafsky and E. Gaussier eds. Proc. of the 2006 Conference on Empirical Methods in Natural Language Processing. Sydney, Australia: Association for Computational Linguistics. 2006, 275-283.

[6] Pascal Denis and Jason Baldridge. Joint determination of anaphoricity and coreference resolution using integer programming. In Proceedings of the North American Association for Computational Linguistics and the Conference on Human Language Technology. 2007.

[7] Jenny Rose Finkel and Christopher D. Manning. 2008. Enforcing Transitivity in Coreference Resolution. Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics. 2008, Short Papers, pp. 45-48.

[8] X. Luo, et al. A mention-synchronous coreference resolution algorithm based on the bell tree. In: D. Scott ed. Proc. of the 42th annual meeting on Association for Computational Linguistics. Barcelona, Spain: Association for Computational Linguistics. 2004, 135-142.

[9] Newman, MEJ; Girvan, M. Finding and evaluating community structure in networks. Phys Rev E. 2004;69(no 026113)

[10] Newman, MEJ. Fast algorithm for detecting community structure in networks. Phys Rev E. 2004;69(no 066133)

[11] M Vilain , J Aberdeen et al. A model theoretic coreference scoring scheme. In : Proc. of the 6th Message Understanding Conf (MUC6) , San Francisco : Morgan Kaufmann Publishers. 1995, 45-52.