自然语言文本共指消解性能评测算法研究*)

史树敏[1,3] 黄河燕[2] 刘东升[3]
(南京理工大学计算机科学与技术学院 南京 210094)[1]
(中科院计算机语言信息工程研究中心 北京 100097)[2]
(内蒙古师范大学计算机与信息工程学院 呼和浩特 010022)[3]

摘 要 在自然语言文本处理中,共指研究处于基础且关键的一环,它的有效解决将为众多的语言工程处理问题提供有力支持,因此对于共指消解性能的评测就显得尤为重要。针对共指消解任务的几种评测算法进行了深入研究,分析各种方法优缺点,指出CEAF方法是目前缺乏用于指代任务处理开放语料情况下,较适合汉语自然语言文本共指消解性能评价的一种方法。

关键词 共指消解评测,MUC-6算法,B-CUBE算法,ACE-value评测,CEAF算法

Research on Natural Language Text Coreference Resolution Evaluation Algorithms

SHI Shu-min[1,3] HUANG He-yan[2] LIU Dong-sheng[3]
(School of Computer Science and Technology, Nanjing University of Science & Technology, Nanjing 210094, China)[1]
(Research Center of Computer & Language Information Engineering, CAS, Beijing 100097, China)[2]
(Computer & Information Engineering College, Inner Mongolia Normal University, Huhhot 010022, China)[3]

Abstract Coreference resolution is a basal and important task in Nature Language Processing (NLP), its availability can give strongly the support for many language engineering applications. Meanwhile, performance evaluation methods and algorithms are crucial in coreference resolution tasks. This paper discusses in detail several current evaluation methods and algorithms, and analyzes their merits and shortcomings, points out that CEAF is better than other algorithms in Chinese natural language text coreference resolution processing under the situation that there are not mass open and sharable annotated corpus focusing on Chinese coreference relation recognition.

Keywords Coreference resolution evaluation, MUC-6 algorithm, B-CUBE algorithm, ACE-value evaluation, CEAF evaluation

1 引言

在自然语言文本处理中,共指问题研究处于基础且关键的一环,它的有效解决将为众多的语言工程处理问题提供有力支持。共指消解的评价方法和算法非常重要,它可以给出不同系统之间性能优劣的指标,同时为进一步提高系统性能指明方向。一个好的用于评价共指消解性能的标准应该具备两点:一是辨别能力;二是解释能力。前者是指能够判断出系统共指关系消解处理好坏的能力;后者是指具备对评价结果合理直观解释的能力[1]。由于各种评测方法产生的背景不同,切入角度不同,定义的概念和特征以及对共指关系的识别要求也不尽相同,为了更好地分析比较各种评测算法的特点,本文不采用真实语料,而是采取人工构造统一的数据样例的方式进行分析比较。

2 MUC-6算法

MUC-6算法是一种基于链(link-based)的评价标准算法,曾广泛用于MUC(Message Understanding Conference)会议的指代消解任务[2]。算法定义了两个术语"key"和"response"。key是指手工标注的共指链(参照标注共指链);response是指由某一参评系统输出的共指链(系统输出共指链)。同时,定义等价类为共指链的闭包。算法基本思路如下:

首先,用S表示由key生成的一个共指类集,用$R_1,\cdots,R_m$表示由response生成的等价类,然后在S上定义如下函数:

• $P(S)$:一个相对于输出共指链的S的划分集。在划分集中的每一个S的子集都是通过将那些与S有重叠关系的输出共指链R_i与S作交集运算后产生。需要注意的是由response产生的等价类可能包含有关系不明确的单集,即在key中有标注但未在response中出现的元素集,也就是在后来的ACE评测会议中定义的漏标情况(第4节详细介绍)。

• $C(S)$:生成等价类集S所需的正确链接(Links)的最小值。该值是集合S的基数减一,即$C(S)=|S|-1$。

• $m(S)$:相对于S,在response中"遗漏(missing)"的链接数目。该值为能完全使任一划分$P(S)$重聚合所必需的链接的数目,即$m(S)=|P(S)|-1$。

单独看参照标注链key中的一个等价类,召回错误是漏标的链接数目除以正确的链接数目,即$m(S)/C(S)$。MUC-6算法定义召回率为式(1),进一步简化为式(2)。将单一情况扩展至整个等价类集,最终召回率R_T为式(3)。

*)基金项目:国家863基金项目(2006AA01Z152),国家自然科学基金项目(60672149)资助。**史树敏** 博士研究生,主要研究兴趣为自然语言处理、信息抽取、机器翻译。

$$\frac{C(S)-m(S)}{C(S)} \tag{1}$$

$$\frac{|S|-|P(S)|}{|S|-1} \tag{2}$$

$$R_T=\frac{\sum(|S_i|-|P(S_i)|)}{\sum(|S_i|-1)} \tag{3}$$

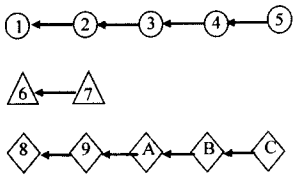

图 1　参照实体(标准共指链)图

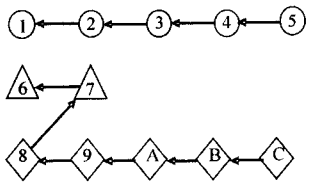

图 2　系统输出情况图

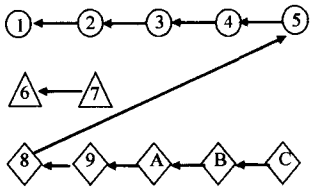

图 3　系统输出情况图

准确率是将标注共指链和输出共指链两个参数的位置互换计算得到的。

举例来看:参照标注共指链包含三个等价类(如图 1 所示),分别表示为 $S_1\{1,2,3,4,5\}$、$S_2\{6,7\}$和 $S_3\{8,9,A,B,C\}$(图中用不同形状以示区别,箭头代表存在共指),则相应的一个划分为 $P(S_1)$、$P(S_2)$和 $P(S_3)$。系统输出共指情况,如图 2 所示。根据式(3),得到召回率 Recall=[(5−1)+(2−1)+(5−1)]/[(5−1)+(2−1)+(5−1)]=9/9=100%。准确率 Precision 的计算将图 1 和图 2 角色互换,即图 2 中的共指链为参照标准,将图 1 中的情况视为系统的一种输出,则有两个共指链 $S_1\{1,2,3,4,5\}$、$S_2\{6,7,8,9,A,B,C\}$,相应划分为 $P(S_1)$:{1,2,3,4,5}和 $P(S2)$:{{6,7},{8,9,A,B,C}}。此时,按照式(3)得到准确率 Precision=[(5−1)+(7−2)]/[(5−1)+(7−1)]=9/10=90%。

MUC-6 算法存在两个明显的不足:一个是算法没有考虑单集(仅一个实体元素,不存在链关系)对共指链的信度指标,且默认遵循在共指标注中,不再去识别那些与其他实体可能已有共指关系的实体的规则,使得若这些实体确实与文本中的其他实体存在共指性,也只能被标注为当前的一种情况,无法优化处理。第二个不足是,所有的错误都被均等看待,解释力差。这样使得以共指消解处理为基础工作的其他具体应用,如信息抽取、机器翻译等工程实践中存在了错误累计和传播扩散的潜在问题,因为有的共指识别错误对识别性能更有害。直观上,图 3 中的错误就比图 2 中的更严重,图 3 中更多的实体被赋予了错误的共指关系,但使用 MUC-6 算法,图 3 中准确率也为 90%(Precision=[(2−1)+(10−2)]/[(2−1)+(10−1)]=9/10=90%),与图 2 的识别结果相同。

3　B-CUBE(B^3)算法

B-CUBE(B^3)是一种改进的 MUC-6 算法[3],能够克服 MUC-6 算法以上两个不足。算法采用的是直接针对实体(Entity)的计算,而不是 MUC-6 方法中的基于实体链(Link)的计算。准确率和召回率的计算公式叙述如下:

$$precision_i=\frac{\text{包含实体 } i \text{ 的输出链中正确的元素个数}}{\text{包含实体 } i \text{ 的输出链全部元素的个数}}$$

$$recall_i=\frac{\text{包含实体 } i \text{ 的输出链中正确的元素个数}}{\text{包含实体 } i \text{ 的参照输出链中的全部元素的个数}}$$

整体公式为加权和,参见式(4)和式(5),其中 N 是文档中实体的数目,w_i 是分配给每个实体的权值,克服了 MUC-6 方案中错误率均等的不足,不同的权重方案产生不同的算法。

$$Precision=\sum_{i=1}^{N}w_i * precision_i \tag{4}$$

$$Recall=\sum_{i=1}^{N}w_i * recall_i \tag{5}$$

$$\frac{1}{|S|}\sum_{j=1}^{m}\sum_{\text{for each } e\in P_j}\frac{m_j(S)}{|S|} \tag{6}$$

B^3 算法的具体思想如下:首先,S 表示由标注共指链生成的一个共指类集,$R_1,\cdots,R_m$ 表示由输出共指链生成的等价类,然后在 S 上定义如下函数:

• $P(S)$:是一个关于输出共指链的 S 的划分集合。即 $P(S)$是 S 的子集的集合,集合的构成方式也是将 S 与那些和 S 有交迭的输出共指链集 R_i 取交。令 $P(S)=\{P_1,P_2,\cdots,P_m\}$,其中 P_i 是 S 的一个子集。

• $m_j(S)$:每一个 P_j 相对于标注共指链集 S 漏标的元素的数目,因此,$m_j(S)=(|S|-|P_j|)$。因为 B^3 算法考虑的是相对于每一个不同的其他实体,当前考察实体的存在与否,整个等价类的漏标数目是等价类中相对于每个实体的漏标数目之和。召回错误为:$m_j(S)/|S|$,整个等价类的召回错误为式(6)。进而所得召回率为式(7)。从单一等价类集 S 扩展到所有的等价类 $T=\{S_1,S_2,\cdots,S_n\}$,每一个等价类 S_i 的召回率为式(8)。其中,P_{ij} 是分割 $P(S_i)$的第 j 个元素,因此是 S_i 的一个子集。不同的整合策略得到不同的算法。如果分配等权重到等价类,则召回率 R_T 直接取均值参见式(9),若等权重分配到每个实体,则权重在对应等价类中处理后再取均值参见式(10)。

$$1-\frac{1}{|S|}\sum_{j=1}^{m}\sum_{\text{for each } e\in P_j}\frac{m_j(S)}{|S|} \tag{7}$$

$$R_i=1-\frac{\sum_{j=1}^{m}\sum_{\text{for each } e\in p_{ij}}|S|-|P_{ij}|}{|S_i|^2} \tag{8}$$

$$R_T=\frac{1}{n}\sum_{i=1}^{n}R_i \tag{9}$$

$$R_T=\sum_{i=1}^{n}\frac{|S_i|}{\sum_{j=1}^{n}|S_j|}R_i \tag{10}$$

按照 B^3 思想,图 1 取标准参照的均值权重 1/12,则图 2 和图 3 中的准确率分别为 76%(1/12 * [5 * 1+2 * (2/7)+5 * (5/7)]=16/21)和 58%(1/12 * [5 * (1/2)+2 * 1+5 * (1/2)=7/12]),召回率仍保持为 100%。也就是说针对图 2 和图 3 的共指识别情况,B^3 算法在精确率上对不同错误的惩罚是不同的。同时,B^3 算法还可以针对不同类型应用采用不同的权值分配方案,如信息抽取属于准确率敏感型任务(precision-sensitive),取得实体均值权重,因为在每一等价类中的每一实体的有效指称都是重要的,如前述情况;而通常信息检

索相对属于召回率敏感(recall-sensitive),权重可以分配给每个等价类,不必要给每个实体,因此图2中的权重方案可调整为:第一个等价类为1/10、第二个等价类为1/14。尽管B^3算法采取的是直接面对实体进行计算的方案,使得它克服了MUC-6算法的一个"痼疾",但是在具体实施过程中,同一实体被多次匹配重复计算严重,产生调和F-值出现不合理问题。

4 ACE-value方法

ACE(Automatic Content Extraction)评测是当前正推动信息抽取研究进一步发展的主要动力之一,由美国DARPA资助。ACE-value是其中不针对某个具体的领域或场景(不同于MUC评测的基于特定场景和主题),采用基于漏报和误报为基础的一套评价体系[4]。需要指出的是,在MUC和ACE中针对共指消解任务,有些概念含义不同。按照ACE评测的定义,每个单独的短语称为"提及(mention)",每个等价类或共指链称为"实体(entity)",与前文中MUC的定义,字面意义上正好相反。本节为了叙述方便,按照ACE的定义来进行算法说明。

在ACE任务中,采用的是基于值(value-based)的衡量标准,称为ACE-value。ACE-value通过统计误报、漏报和错报的实体数目计算得到。所谓误报(false-alarm)是指并没有指称关系的实体被系统识别输出;漏报(missing)是指本应该有指称关系的实体,未能被系统识别输出;错报就是纯粹的实体指称关系错误。每一类错误都与一个代价因子(cost factor)相关联,该因子根据不同的实体类型(如:LOCATION,PERSON,etc.)和提及类型(NAME,NOMINAL,PRONOUN)制定。总代价是各类代价之和,归一化处理后与1比较,得到最后的ACE-value。评测规则中指出最好的共指消解系统ACE-value可获值100%,未得到任何正确实体输出的系统值为0,而一个系统若得到的错误实体太多,甚至可能得到负值。

EDR(Entity Detection and Recognition)任务是ACE评测中共指消解的主要处理环节,具体看一下其对应的ACE-value评测方法。EDR的分值是系统输出全部的实体分值总和与全部参照实体分值总和做归一化除处理后的结果,参见式(11)。每个系统输出实体分值是由其特征与参照实体对应特征的匹配程度决定的。服从系统输出与参照输出一对一匹配限定,取得最大EDR_Value结果的二者之间的全局最佳关联被采用,其中系统输出标注的分值由两类实值决定:一类表示与参照标注情况一致:Mentions_Value(sys,ref);一类表示与参照不一致:Mentions_Value($\overline{sys}$,ref),参见式(12)。W_{FA}参数默认取0.75。Element_Value是系统输出标注的特征函数,反映了与参照标注的匹配程度,由实体特征(如:type,subtype和class)决定,这些特征值使用不同权重参数$W_{err-attribute}$,根据任一实体特征的错误按式(13)和式(14)进行约减处理。

$$EDR\_Value_{sys}=\sum_i value\_of\_sys\_token_i / \sum_j value\_of\_ref\_token_j \tag{11}$$

$$Value(sys,ref)=Element\_Value(sys,ref)\cdot Mentions\_Value(sys,ref)-W_{FA}\cdot Element\_Value(sys)\cdot Mentions\_Value(\overline{sys},ref) \tag{12}$$

$$Element\_Value(sys,ref)=\prod_{\substack{attribute=\\type,subtype,class}}\left\{\min\left(\frac{AttrValue(attribute_{sys})}{AttrValue(attribute_{ref})}\right)\cdot W_{err-attribute}\right\} \tag{13}$$

$$Element\_Value(sys)=\prod_{\substack{attribute=\\type,subtype,class}} AttrValue(attribute_{sys}) \tag{14}$$

Mentions_Value是一个反映了系统输出标注与参照标注提及共有提及值Mutual Mention Value(MMV)的函数。一个提及的MMV由提及的类型参数:MTypeValue决定,该特征值也使用一个权重参数W_{Merr},根据提及特征(type,role,style)的错误情况按式(15)和式(16)进行约减处理。同样,在提及层次,也要确定每一系统标注与参照标注对二者之间的最佳关联,取得最大Mentions_Value的匹配也要服从一对一限制。Mentions_Value的计算使用的两种公式取决于两类不同的评价权重:mention_weighted和level_weighted。按照mention_weighted计算,Mentions_Value是所有文档中全部提及的MMV特征之和,参见式(17)或(18)。

$$MMV(mention_{sys})=MTypeValue(mention_{sys}) \tag{15}$$

$$MMV(mention_{sys},mention_{ref})=\begin{cases}\min\left(\frac{MTypeValue(mention_{sys})}{MTypeValue(mention_{ref})}\right)\cdot\prod_{\substack{attribute=\\type,subtype,class}} W_{Merr-attribute} & \text{if } mention_{sys} \text{ and } mention_{ref} \text{ correspond}\\ 0 & \text{otherwise}\end{cases} \tag{16}$$

$$Mentions\_Value(sys,ref)=\sum_{all\ docs}\left(\sum_{\substack{all\ sys mentions\\ indocthatmap\\ to\ ref mentions}} MMV(mention_{sys},mention_{ref})\right) \tag{17}$$

$$Mentions\_Value(\overline{sys},ref)=\sum_{all\ docs}\left(\sum_{\substack{all\ sys mentions\\ indocthatdon't\\ map\ to\ ref mentions}} MMV(mention_{sys})\right) \tag{18}$$

按照level_weighted计算,Mentions_Value由系统输出标注(及相应参照标注)的层次、系统与参照标注的关联程度和提及所在文档的数目等众多因素决定,参见式(19)和(20)。系统输出和参照提及的关联至少满足最小中心词重叠:min_overlap(默认0.3)(参见式(21))和最小中心词所在文本片断共享连续字(词)串:min_text_match(默认0)(参见式(22)两个参数后才被认可。

$$Mentions\_Value(sys,ref)=MTypeValue(level_{doc,ref})\cdot\sum_{all\ docs}\left(\frac{\sum_{\substack{all\ sys mentions\\ indocthatmap\\ to\ ref mentions}} MMV(mention_{sys},mention_{ref})}{\sum_{\substack{all\ ref mentions\\ indoc}} MMV(menton_{ref})}\right) \tag{19}$$

$$Mentions\_Value(\overline{sys},ref)=MTypeValue(level_{doc,sys})\cdot\sum_{all\ docs}\left(\frac{\sum_{\substack{all\ sys mentions\\ indocthatdon't\\ map\ to\ ref mentions}} MMV(mention_{sys})}{\sum_{\substack{all\ sys mentions\\ in\ doc}} MMV(menton_{sys})}\right) \tag{20}$$

$$mutual\_overlap=\frac{sys\_head\cap ref\_head}{\max(sys\_head,ref\_head)} \tag{21}$$

$$fractional\_consecu-tive\_string=\frac{\left(\begin{array}{l}\#\text{ of characters in the longest consecu-tive string of characters}\\ \text{that is contained in both system and reference mention head texts}\end{array}\right)}{\max\left(\begin{array}{l}\#\text{ of characters in syetem mention head text}\\ \#\text{ of characters in reference mention head text}\end{array}\right)} \tag{22}$$

由于 ACE-value 也是面向实体的，因此避免了在 MUC-6 中存在的错误。但是 ACE-value 的评分机制严格依赖其标注语料（其中实体特征、提及类型均是专门定义，语料不对外开放）。同时它的评价结果不易理解，因为它通过权重惩罚因子，对分值进行的是代价计算，如一个系统得到 85% 的 ACE-value 结果，并不意味着系统输出的实体或提及达到 85% 的正确率，只是说明了该系统相对于一个没有任何实体输出的系统而言，成本消耗是 15%（ACE-Value 算法认为没有实体输出没有成本消耗）。这与人们对于评测分值的直观理解有很大偏差。

5 CEAF 方法

由于 ACE-value 结果并不能直观解释得分高的系统性能优于得分低的系统，因此 Luo 提出了一种 CEAF（Constrains Entity Aligned F-Measure）方法衡量共指消解性能[1]。其核心思想是在系统输出和标准参照两个子集中做实体一对一的匹配：在每一个参照实体最多与一个系统输出实体对齐，反之亦然的限定条件下，通过最大化实体相似度将系统实体与参照实体对齐。在实体对齐上的限定条件保证了不会产生“欺骗性”的结果（因为通过输出过多实体得到高召回率的系统会在准确率上受到“惩罚”，而依靠输出少量实体获得高准确率的系统会遭到召回率上的惩罚）。一旦定义好相似度，准确率、召回率和 F-值就能够直接计算，算法的含义明确。实际操作中，又进一步分为两种：基于提及（mention-based）的 CEAF，用以直接反映在正确的实体（共指链）中提及的比例；基于实体（entity-based）的 CEAF，用于直接反映正确识别到的实体（共指链）的比例。

具体看一下 CEAF 方法：设文档 d 中的参照实体为：$R(d)=\{R_i: i=1,2,\cdots,|R(d)|\}$；系统输出实体为：$S(d)=\{S_i: i=1,2,\cdots,|S(d)|\}$。为了简化，$R(d)$ 记为 R；$S(d)$ 记为 S。令：$m=\min\{|R|,|S|\}$；$M=\max\{|R|,|S|\}$。令 $R_m \subset R$ 和 $S_m \subset S$ 分别是 R 和 S 的任一拥有 m 个实体的子集，即 $|R_m|=m$，$|S_m|=m$。$G(R_m,S_m)$ 是从 R_m 到 S_m 的一对一实体匹配集，G_m 是 R 和 S 中所有大小为 m 的子集的一对一匹配集，记为：$G(R_m,S_m)=\{g: R_m(S_m\}$；$G_m=\bigcup_{(Rm,Sm)} G(R_m,S_m)$。一对一匹配的限制表明对于任意的 $g\in G(R_m,S_m)$ 和 $R\in R_m$，$R'\in R_m$，我们能够推出：如果 $R\neq R'$，则 $g(R)\neq g(R')$，反之亦然的结论。很明显从 R_m 到 S_m 共有一对一映射 $m!$ 个，即 $G(R_m,S_m)=m!$。令 $\phi(R,S)$ 表示实体 R 和 S 间“相似度”，取非负值。若值为 0，表明 R 与 S 之间没有任何共有部分。如：$\phi(R,S)$ 是 R 和 S 共享的共有提及数目，$\phi(R,R)$ 是实体 R 中提及的数目。

对于任意 $g\in G_m$，一个匹配 g 的总相似度 $\Phi(g)$ 是对齐实体对间的相似度之和：$\Phi(g)=\sum_{R\in Rm}\phi(R,g(R))$。给定文档 d 和它的参照实体 R 与系统实体 S，可得最大总相似度的最佳对齐为 g^*，参见式(23)。令 R_m^* 和 $S_m^*=g^*(R_m^*)$ 分别表示取得 g^* 时，参照实体和系统实体的子集，则最大的总相似度为 $\Phi(g^*)$，参见式(24)。当 R 或者 S 中有一个为空集时，$\phi(R,S)=0$。而 $\phi(R,S)=0$ 非负的要求使得没有必要考虑一个实体会与一个空实体进行匹配的情况，因为最大化一对一匹配的 $\Phi(g)$ 一定在 G_m 中。对任一 $R\in R$ 和 $S\in S$，计算实体自相似度为 $\phi(R,R)$ 和 $\phi(S,S)$。

$$g^*=\arg\max_{g\in G_m}\Phi(g)=\arg\max_{g\in G_m}\sum_{R\in R_m}\phi(R,g(R)) \tag{23}$$

$$\Phi(g^*)=\sum_{R\in R_m^*}\phi(R,g^*(R)) \tag{24}$$

$$p=\frac{\Phi(g^*)}{\sum_i\phi(S_i,S_i)} \tag{25}$$

$$r=\frac{\Phi(g^*)}{\sum_i\phi(R_i,R_i)} \tag{26}$$

$$F=\frac{2pr}{p+r} \tag{27}$$

$$\phi_1(R,S)=\begin{cases}1,\text{if } R=S\\0,\text{otherwise}\end{cases} \tag{28}$$

$$\phi_2(R,S)=\begin{cases}1,\text{if } R\cap S\neq\Theta\\0,\text{otherwise}\end{cases} \tag{29}$$

$$\phi_3(R,S)=|R\cap S| \tag{30}$$

$$\phi_4(R,S)=\frac{2|R\cap S|}{|R|+|S|} \tag{31}$$

定义准确率、召回率和 F-值分别为式(25)－(27)。优化的对齐 g^* 仅考虑了 $m=\min\{|R|,|S|\}$ 的情况，未对齐的实体没有得到信度，因此式(27)的 F-值对产生过多（低准确率）或者过少实体（低召回率）的共指消解系统都进行了惩罚。

在上面的讨论中，是假设对全部的实体对 (R,S)，计算实体相似度 $\phi(R,S)$。在实际操作中，如果已知 R 和 S 没有共有实体，可以避免 $\phi(R,S)$ 的计算（因为此时 $\phi(R,S)=0$）。这些彼此没有链接的实体在最佳对齐的搜索中不予考虑，当有大量 0 值相似度的实体对存在的时候，F-值计算的速度将大大加快。在最佳对齐的过程中仅有少于 m 的参照和系统实体被涉及到。说明一点，式(27)中，仅是针对语料中的一篇文档，扩展到多文档情况，只需依据式(24)和式(26)累加分子、分母，再计算 F-值。

CEAF 中关键部分是定义在实体对 (R,S) 上的实体相似度 $\phi(R,S)$ 的计算。先看两个极端的情况，式(28)表示若 R 和 S 中的所有实体都相同，则相似度为 1；式(29)表示若 R 和 S 中有至少一个相同的实体，则相似度为 1，参见式(28)－(29)。很明显这两种情况都是极端条件，一个共同特点是无法对不同的输出系统性能进行区别判断。举个简单的例子，参照实体集 $R=\{a,b,c\}$，某一个输出系统 $S_1=\{a,b\}$，另一个输出系统 $S_2=\{a\}$，直观上 S_1 比 S_2 要更接近 R，即相似程度更好，但利用式(28)二者的相似度 $\phi(R,S)$ 均为 0；而利用式(29)，二者的相似度都是 1。

为了更清晰地获得实体相似性能的测度，而不仅仅是通过二元测度简单判断，Luo 选择了一种自然的方式来表征实体的相似程度，即统计实体中的相同的“提及”，采用两种方式：绝对数目（absolute number）和相对数目（relative number），得到式(30)和式(31)。式(30)计算 R 和 S 中的相同的提及数目，式(31)计算 R 和 S 中的共享提及的调和 F-值，是一个相对数目。分别计算 ϕ_i。得 $\phi_3(R,S_1)=\phi_3(\{a,b,\},\{a,b\})=2$；$\phi_3(R,S_2)=\phi_3(\{a,b,c\},\{a\})=1$；$\phi_4(R,S_1)=\phi_4(\{a,b,c\},\{a,b\})=0.8$；$\phi_4(R,S_2)=\phi_4(\{a,b,c\},\{a\})=0.5$。得到一个合理的等级排列：$\phi_3(R,S_1)>\phi_3(R,S_2)$，$\phi_4(R,S_1)>\phi_4(R,S_2)$。

进而当式(25)和式(26)中分母分别取参照实体数目和系统实体数目时，采用 $\phi_3(\cdot,\cdot)$ 的相似度计算方法，$\Phi(g^*)$ 就是相应地满足了最佳一对一匹配 $g*$ 的共有提及的总数目，式(27)得到的 F 值就可解释为在“正确”实体链中的提及的比率；类似的如果采用 $\phi_4(\cdot,\cdot)$，则式(27)产生的 F-值就可

（下转第 177 页）

gantown, WV. June 1977

[2] Godbole S. An Efficient Computation of Matrix China Products. IEEE Trans. on Computers, Sept. 1973: 864-866

[3] Hu T C, Shing M T. Computation of Matrix China Products. part Ⅰ. SIAM J. Compute, May 1982, 11: 362-373

[4] Hu T C, Shing M T. Computation of Matrix China Products. part Ⅱ. SIAM J. Compute, May 1984, 13: 228-251

[5] Chin F. An O(n) Algorithm for Determining a Near-Optimal Computation Order of Matrix China Product. Comm. ACM, 1978: 544-549

[6] Valiant L, Skyum S, Berkowitz S, et al. Fast Parallel Computation of Polynomials Using Few Processors. SIAM Journal on Computing, 1983, 12: 641-644

[7] Rytter W. Note on Efficient Parallel Computations for Some Dynamic Programming Problems. Theoret. Comp. Sci., 1988, 59: 297-307

[8] Huang S-H S, Liu H, Viswanathan V. Parallel Dynamic Programming. IEEE Trans. on Parallel and Distributed Systems, Mar. 1994, 5: 326-328

[9] Bradford P G, Rawlins G J, Shannon G E. Efficient Matrix China O-rdering in Polylogtime // Proc. of Int'l Parallel Processing Symp. 1994: 234-241

[10] Czumaj A. Parallel Algorithm for the Matrix China Product and t-he Optimal Triangulation Problems // Proc. of Symp. on Theoret. Aspects of Computer Science. Springer Verlag, 1993: 294-305

[11] Czumaj A. Very Fast Approximation of the Matrix China Product Problem. J. Algorithms, 1996, 21(1): 71-79

[12] Ramanan P. An Efficient Parallel Algorithm for the Matrix C-hain Product Problem. SIAM J. Comput. Aug. 1996, 25: 874-893

[13] Strate S A, Wainwright R L. Parallelization of the Dynamic Programming Algorithm for the Matrix China Product on a Hypercube. IEEE, 1990

[14] Lee H, Kim J, Hong S, et al. Parallelizing Matrix China Products. Tech. Rep. CS-HPC-97003. Pohang University of Science and Technology, 1997

(上接第 171 页)

理解为正确实体链的比率。因此 Luo 将由式(30)产生的式(27)称为基于提及(mention-based)的 CEAF 方法；由式(31)得到的式(27)称为基于实体(entity-based)的 CEAF 方法。不同的层次上，可对 $\phi_3(\cdot,\cdot)$和 $\phi_4(\cdot,\cdot)$定义相应的权重值，如根据不同实体类(ACE 中为 NAME, NOMINAL, PRONOUN)定义不同权重值。

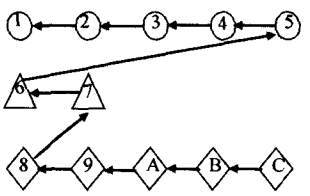

图 4 系统输出情况图

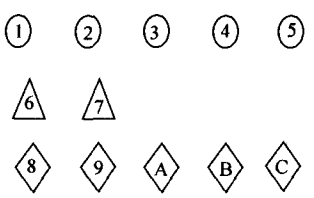

图 5 系统输出情况图

表 1 各类评测算法准确率、召回率及调和 F-值情况比较

图例	MUC			B^3			CEAF					
	P	R	F	P	R	F	ϕ_3-P	ϕ_3-R	ϕ_3-F	ϕ_4-P	ϕ_4-R	ϕ_4-F
2	0.9	1	0.947	0.76	1	0.864	0.833	0.833	0.833	0.917	0.611	0.733
3	0.9	1	0.947	0.58	1	0.734	0.583	0.583	0.583	0.833	0.555	0.667
4	0.918	1	0.9	0.375	1	0.54	0.417	0.417	0.417	0.588	0.196	0.294
5	—	—	—	1	0.25	0.4	0.25	0.25	0.25	0.111	0.444	0.178

最后依然以图 2,3 中数据为基准，同时引入图 4,5 两种极端输出，针对不同 CEAF 方法进行计算，结果见表 1，表中第三行至第六行分别对应图 2—5 情况。为了方便比较分析，将 B^3 与 MUC-6 两种方法针对图 4,5 的情况一同给出。分析可以看到，尽管 B^3 最终的 F 值比 MUC-6 理想一些，但它在极端条件下，准确率和召回率上均出现了不合理值。而 CEAF 方法的三类指标的结果更具说服力，也更为合理直观。最后需要指出的是 Luo 中提出的最佳对齐方法是一个经典的最大双向匹配问题，其算法思想得益于 Kuhn(1955)和 Munkres(1957)提出的 Kuhn-Munkres 算法[5,6]。

结束语 MUC-6 算法的缺点主要有两个：一个是算法没有区分单集，因此无法正确处理类似图 5 中的输出情况；一个是所有的错误都被等同看待，算法仅是平等的对各种类型错误进行惩罚。B^3 算法采用基于实体的策略，但它与 MUC-6 一样存在同一实体被多次匹配重复计算问题，仍存在 F-值不合理现象。ACE-Value 是近年来共指消解任务的官方评测算法，但是在分值解释方面不直观，除此之外，由于该算法严格依赖其 ACE 评测标注语料，该语料无法公开共享使用，评测中的诸多细节也少见诸公开文献。CEAF 方法含义清晰，理解直观，评分解释合乎实际。目前，汉语共指消解处理尚无大规模共享标注语料，依靠实体与提及的绝对和相对数目的统计进行客观比较，综上所述，CEAF 不失为一种最佳选择，特别是它能够根据实际需要决定不同权重参与的灵活性，更加适合于当前共指消解问题仅能在限定领域开展较为深入的工作的实际情况。

参 考 文 献

[1] Luo Xiaoqiang. On Coreference Resolution Performance Metrics // Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP). Vancouver, October 2005: 25-32

[2] Vilain, Mare, et al. A Model-Theoretic Coreference Scoring Scheme // Proceedings of the 6th Message Understanding Conference (MUC-6). Nov. 1995: 45-52

[3] Bagga A, Baldwin B. Algorithms for scoring coreference chains // Proceedings of Linguistic Coreference Workshop at the 1st International Conference on Language Resources and Evaluation. 1998: 563-566

[4] ACE 07. The ACE evaluation plan. www.nist.gov/speech/tests/ace/index.htm.-ace07-evalplan.v1.3a.pdf.

[5] Kuhn H W. The hungarian method for the assignment problem. Naval Research Logistics Quarterly, 1955 (2): 83

[6] Munkres J. Algorithms for the assignment and transportation problems. Journal of SIAM, 1957 (5): 32-38