提取半结构化信息源中对象间精确语义相似性的方法研究*)

高 英 郭荷清 邹智敏

(华南理工大学计算机科学与工程学院 广州 510641)

摘 要 为了从半结构化的信息源中提取本体,统一地分析和处理所有信息源,本文为数据源提供了一个统一的概念模型,并定义了半结构化信息源到概念模型的转换规则。基于该模型,本文提出了计算模式中对象间精确语义相似性的方法。

关键词 本体,语义距离相关图,0 近邻相似性分析,i 近邻相似性分析

The Research of Method of Extracting Refined Semantic Similarity among Objects of Semi-structure Information Sources

GAO Ying GUO He-Qing ZOU Zhi-Min

(Department of Computer Science and Engineering, South China University of Technology, Guangzhou 510641)

Abstract In order to extract ontology from semi-structure information sources, and analyze and deal with information sources accordantly, this paper provides a new conceptual model for data sources and a sets of rules from semi-structure information sources to conceptual model. Based on this model, this paper provides a method to compute refined semantic similarity among objects of sources.

Keywords Ontology, Semantic-distance-similarity graph (SDS-G), 0 neighbor similarity analysis, i neighbor similarity analysis

随着基于分布式的、异构的、Web 信息系统的逐渐增加,如何对已经发布的信息进行访问和高效地挖掘,已经成为关键的问题。目前,人们已经意识到本体在信息共享中所扮演的角色,已经开发出一些领域和任务驱动的本体用于多个分布式信息源或 agent 之间的交流和协调。但是,"任务驱动的本体设计方法将焦点集中在商业模型中的数据库融合或集成方面,从而导致本体仅仅含有实现任务所需要的知识,不能完整地覆盖整个领域[1]。"

由于互联网存在的大量电子数据通常不具备精确的、定义良好的结构,这些数据称为半结构化信息源。手工从这些数据源中建立本体,是很困难的,而且本体正确性的有效性验证也必须通过人类专家关于领域的知识来完成,这必然导致所建立的本体具有不一致性。因此使用自动的或者至少是半自动化的方法建立本体,已经成为解决该问题的主要方法[2,3]。

本文提出的从半结构化信息源中提取本体的思想是:首先将各信息源转换为一个概念模型(Semantic-distance-similarity Graph,简称 SDS-G),然后对各概念模型中的节点进行相似性分析,依据节点间的相似性形成节点的聚类,最后从聚类中提取本体。整个过程是半自动化的,在方法的早期仅需要少量人类专家的参与。本文将重点介绍从半结构化信息源到 SDS-G 的转换规则,并基于 SDS-G,计算各源间对象的精确语义相似性的方法,为将来提取本体做好准备。

1 SDS-G 的定义

形式上,独立于具体的源格式,SDS-G(Semantic-Distance-Similarity graph)定义为一个带根的标签图:

$$G(IS)=(N(IS),A(IS))=((N_E(IS)\cup N_S(IS)\cup N_R(IS)\cup N_A(IS),A(IS))$$

· $N(IS)$是节点集。每个节点表示 IS 中的一个对象(元素、子元素或属性),并由对象的名字标识。$N(IS)$中的节点由 4 个子集组成:$N_E(IS)$是具有属性或嵌套子元素的元素节点集,向外发出的弧指向它的属性节点或模式中的其它元素节点,是复合节点;$N_S(IS)$是不带属性,也不嵌套子元素的元素节点集;$N_R(IS)$是模式中具有属性的关系节点集;$N_A(IS)$是属性节点集。$N_S(IS)$和 $N_A(IS)$没有向外发出的弧,是原子节点。在本文的论述中,节点和对象可以互换使用。

· $A(IS)$是弧的集合,一条弧代表两个对象之间的一个关系,如从 S 到 T 的弧$\langle S,T,l_{ST}\rangle$表示对象 S 与对象 T 在语义上相关,S 称为源节点,T 称为目标节点;$l_{ST}=[d_{ST},s_{ST}]$,$d_{ST}\in[0,1]$和 $s_{ST}\in[0,1]$,其中 d_{ST} 表示 S 和 T 之间的语义距离系数,s_{ST}表示 S 和 T 之间的语义相关性系数。d_{ST} 表示 T 与 S 在语义上的接近程度,这取决于 T 表达 S 特征的能力。如在 E/R 模式中,A 是实体的一个属性,E_1 是模式中通过关系 R 与 E 发生联系的另一实体,在语义上,A 比 E_1 更接近实体 E。在 XML 中,E_1 是元素 E 的子元素,E_2 是 E 通过 $IDREF$ 属性引向的另一元素,E_1 在语义上比 E_2 更接近 E。T 在语义上越靠近 S,d_{ST} 就越小,反之就越大。s_{ST} 表示 S 的实例中有多少通过一对一、一对多或多对多关系与 T 有关。

2 XML-SDS-G 的转换规则

从 XML 文档 D 中建立 SDS-G:$G(D)=\langle N(D),A(D)\rangle$,

*)国家 973 高科技研究发展计划基金资助项目(G20000263)、广州市重点科技公关项目(B2-109-550)。高 英 博士生,讲师,主要研究领域为信息系统的集成与安全;郭荷清 教授,博士生导师,主要研究领域为信息系统的集成与安全。

如果能得到文档所对应的 DTD 或 Schema，那么可以加速整个构建过程。由于 DTD 和 Schema 中的所有信息可以直接从 XML 文档中推断出来，因此也可以直接从 XML 文档中直接建立 SDS-G，但需要增加额外的计算成本。

设 $S(D)=\{e_1,e_2,\cdots,e_m\}$，$e_i$ 对应于该模式中的元素。

元素 $e_i=\{n(e_i),SE(e_i)REL(e_i),SP(e_i),DP(e_i),C(e_i)\}$，其中：

- $n(e_i)$是元素的名字，为了处理上的简化，本文中假设元素的名字具有唯一性；
- $SE(e_i)=\{se_1,se_2,\cdots,se_n\}$，是元素 e_i 所嵌套的子元素集；
- $REL(e_i)=\phi$；
- $SP(e_i)=\{sp_1,sp_2,\cdots,sp_q\}$是 e_i 的属性集；
- $DP(e_i)=\{dp_1,dp_2,\cdots,dp_r\}$，在 DTD 中，是 e_i 中类型为 *IDREF* 或 *IDREFS* 的属性集；在 XML Schema 中，是 e_i 中类型为自定义类型的属性集；
- $C(e_i)$是 e_i 的所有实例数；

$SP(e_i)\cap DP(e_i)=\phi$，$P(e_i)=SP(e_i)\cup DP(e_i)$

2.1 SDS-G 中节点的定义

设 $N(D)=N_E(D)+N_S(D)+N_R(D)+N_A(D)$，其中

- $N_E(D)=S(D)=\{e_1,e_2,\cdots,e_m\}$，如果 e_i 对应于 DTD 中的一个非终端元素或 XML Schema 中复杂节点型态 complexType 的一个元素，那么有向外发出的弧指向它的属性节点或所嵌套的子元素节点，是复合节点；如果对应于 DTD 中的一个终端元素或 XML Scheme 中简单节点型态 simpleType 的一个元素，没有子元素及属性，那么是原子节点；
- $N_s(D)=\{SE(e_i),i=1,2,\cdots,m\}$，是元素的子元素集。
- $N_R(D)=\phi$；
- $N_A(D)=\{SP(e_i),i=1,2,\cdots,m\}$，是元素的属性集，是原子节点集。

2.2 SDS-G 中弧的定义

SDS-G 弧 $A(D)$由 3 个集合组成：$A(D)=A_1(D)+A_2(D)+A_3(D)$，其中：

- $A_1(D)=\{(x,y)\mid x\in S(D),y\in SE(x)\}$，如果元素 y 是元素 x 的子元素，那么 x 到 y 之间有一条弧，称为嵌套弧；
- $A_2(D)=\{(x,y)\mid x\in S(D),y\in SP(x)\}$，在 DTD 中，如果元素 y 是元素 x 的非 *IDREF* 和 *IDREFS* 属性，那么 x 到 y 之间有一条弧；在 XML Schema 中，如果元素 y 是元素 x 的非 *IDREF* 和 *IDREFS* 的内置数据类型属性，那么 x 到 y 之间有一条弧；
- $A_3(D)=\{(x,y)\mid x,y\in S(D),DP(x)\neq\phi\}$，类型为 *IDREF* 或 *IDREFS* 的属性指向元素 y，x 到 y 之间有一条弧，称为引用弧；在 XML Schema 中，类型为自定义类型的属性指向元素 y，x 到 y 之间有一条弧。

$A_1(D)$和 $A_2(D)$中，弧可以从 DTD 中或 XML Schema 直接得到；$A_3(D)$中，类型为 *IDREF* 和 *IDREFS* 的属性的源节点可以从 DTD 中得到，但是目标节点必须通过分析 XML 文档得到。

2.3 SDS-G 中标签的定义

定义 2.1 XML 文档 D，(1)O 是元素的实例，O'是元素的实例，E'是的子元素；(2)O 是元素 E 的实例，O'是元素 E 的某个属性 A 的实例。如果上述两个条件中任意一个能成立，那么 O'是 O 的组成部分。定义函数

$comp(O)=\{O'\mid O\in ext(E),O'\in ext(E'),E'\in SE(E)\ or\ O\in ext(E),O'\in ext(SP(E)\cup DP(E))\}$

其中 $ext(E)$表示元素的外延(实例)。

定义 2.2 XML 文档 D，对应的模式图 $G(D)$。设 N 为 $G(D)$中的一个节点，$Assoc\text{-}ExtSet(N)=\{ext(N)\}$。设集合：$Assoc\text{-}ExtSet(S)$，$Assoc\text{-}ExtSet(T)$，$RNS_{S,T}=\{O_i\mid \exists O,O\in Assoc\text{-}ExtSet(T),O_i\in Assoc\text{-}ExtSet(S),O\in comp(O_i)\}$，则 S 和 T 的语义距离系数定义为：

$$d_{ST}=\frac{\sum_{O_i\in RNS_{S,T},O\in Assoc\text{-}ExtSet(T)}\gamma_{S,T}(O_i,O)}{|RNS_{S,T}|} \tag{2-1}$$

其中

$$\gamma_{S,T}(O_i,O)=\begin{cases}0 & \text{如果}\exists p,p\in Assoc\text{-}ExtSet(T),p\in comp(O_i),p\text{ 是非 }IDREF\text{ 或 }IDREFS\text{ 的属性的外延，且}\neg\exists q,q\in Assoc\text{-}ExtSet(T),q\in comp(O_i),q\neq p\\ 0.25 & \text{如果}\exists p,p\in Assoc\text{-}ExtSet(T),p\in comp(O_i),p\text{ 是终端元素的外延，且}\neg\exists q,q\in Assoc\text{-}Extset(T),q\in comp(O_i),q\neq p,q\text{ 是元素或者是 }O_i\text{ 的 }DP\text{ 中的属性所引用的对象的外延}\\ 0.5 & \text{如果}\exists p,q\in Assoc\text{-}ExtSet(T),p,q\in comp(O_i,q\neq p,p,q\text{ 是终端元素的外延，}\neg\exists r,r\in Assoc\text{-}ExtSet(T),r\in comp(O_i),r\text{ 是非终端元素或者是 }O_i\text{ 的 }DP\text{ 中的属性所引用的对象的外延}\\ 1 & \text{如果}\exists p\in Assoc\text{-}ExtSet(T),p\in comp(O_i),\text{且 }p\text{ 是非终端元素或者是 }O_i\text{ 的 }DP\text{ 中的属性所引用的对象的外延}\end{cases} \tag{2-2}$$

语义相关系数定义为：$r_{ST}=\frac{|RNS_{S,T}|}{|Assoc\text{-}ExtSet(S)|}$。该公式是直接从 S 和 T 的语义相关性的定义推导出来的，表示与 T 相关的实例在与 S 相关的实例中的比例。

3 语义相似性分析

本文提出的计算模式间语义相似性的方法是通过分析 SDS-G 中节点对应的对象的类型以及数量，挖掘对象之间的术语关系(如同义词关系、同名异义关系、上下位关系、类型冲突关系)，得到模式之间的语义相似性。基于 SDS-G 的语义相似性分析方法由两个阶段组成。

阶段 1：基本语义相似性的计算。

计算节点对 $N_l\in G(IS_1)$和 $N_m\in G(IS_2)$的基本相似性时，仅仅考虑与 N_l 和 N_m 对应的对象名字间的相似性以及对象结构的相似性，及直接与它们相连的属性节点集的相似性。这个阶段所获得的相似程度作为第 2 阶段精确相似计算的基础。

阶段 2：精确语义相似性的计算。

以第 1 阶段的相似性分析结果为基础，按照节点的语义距离，从近到远，计算不同语义距离的近邻的相似性，不断修正相似性的结果，最终得到节点间的精确语义相似性。

在详细介绍语义相似性分析方法之前，首先给出相关的定义。

定义 3-1 $G(IS)$中，路径 P 包含的弧的语义距离系数之和称为路经 P 的路径语义距离。

定义 3-2 $G(IS)$中，路经 P 所包含的弧的语义相关系数之积称为路径 P 语义相关性。

定义 3-3 $D\ \ Path_n$ 定义为$G(IS)$中的一条路径 P，该路

径的语义距离大于或等于 n、小于 $n+1$。

定义 3-4 $G(IS)$中，连接节点 N 和 N' 并且包含弧 A 的所有路经中，具有最小路径语义距离的路径称为 *CD shortest −path*(带条件的最短语义距离路径)，记作：$\llcorner N,N'\lrcorner_A$。如果存在多条这样的路径，那么选择具有最大路径语义相关系数的路径作为 $\llcorner N,N'\lrcorner_A$。

定义 3-5 设 $G(IS)=N(IS)+A(IS)$，节点 $x\in N(IS)$ 的第 i 个近邻定义为：

$$nbh(x,i)=\{A\mid A\in A(IS),A=\langle z,y,L_{zy}\rangle,\llcorner x,y\lrcorner_A\in D\_Path_i,x\neq y,i\geqslant 0\}$$

3.1 基本语义相似性分析

3.1.1 节点名相似性分析

本节提出的分析方法主要基于节点名间的同义关系(*SYN*)、上下位关系(*BT*，一般为特殊关系)、包含关系(*RT*)等 3 种术语关系。在 *SDS-G* 中，节点 $o\in N_E(IS)\cup N_S(IS)\cup N_R(IS)$的 *BT/NT* 关系和 *RT* 关系可以通过以下方式从源模式中自动获得：

- *BT/NT* 关系可以通过源模式的泛化关系获得；
- 在 *ER* 对应的 SDS-G 中，如果 $l\in A_4(IS)$，弧 l 所表示的关系在源实体和目标实体中均与主键有关，那么直接与弧 l 相连的节点对具有 *BT/NT* 关系；
- 在 *ER* 模式对应的 SDS-G 中，如果 $l\in A_1(IS)\cup A_4(IS)$，那么直接与弧 l 相连的节点对具有 *RT* 关系；
- 在 XML 对应的 SDS-G 中，如果 $l\in A_1(IS)\cup A_2(IS)\cup A_3(IS)$，那么直接与弧 l 相连的节点对具有 *RT* 关系。

根据术语关系在相似性分析中的语义含义，赋予每个术语关系一个强度 $\sigma\in[0,1]$。由于同义词所包含的相似性比其他两种术语关系更精确，因此最高的强度分配给 *SYN* 关系；文[31]讨论了“*is-a*”关系表达了对象和关系之间更高的语义连接，因此，$\sigma_{BT/NT}\geqslant\sigma_{RT}$。在本文中，我们设置 $\sigma_{SYN}=1$，$\sigma_{BT/NT=0.8}$，$\sigma_{RT}=0.5$。

设计人员可以借助标准词典和领域知识对得到的术语关系进行修正和补充，并在修正的基础上推导新的术语关系。推导时，限制推导路径中所含有的 *BT/NT* 关系或 *RT* 关系最多不超过两个，*SYN* 关系最多不超过 1 个。推导规则如下：

- $\forall(n_i,n_r,n_j)\langle n_i\ SYN\ n_j\rangle\bigcap\langle n_j\ RT\ n_r\rangle\Rightarrow\langle n_i\ RT\ n_r\rangle$；
- $\forall(n_i,n_r,n_j)\langle n_j\ SYN\ n_i\rangle\bigcap\langle n_i\ RT\ n_r\rangle\Rightarrow\langle n_j\ RT\ n_r\rangle$；
- $\forall(n_i,n_j,n_r)\langle n_i\ SYN\ n_j\rangle\bigcap\langle n_j\ BT\ n_r\rangle\Rightarrow\langle n_i\ BT\ n_r\rangle$；
- $\forall(n_i,n_j,n_r,n_t)\langle n_i\ SYN\ n_j\rangle\bigcap\langle n_j\ BT\ n_r\rangle\bigcap\langle n_i\ RT\ n_t\rangle\Rightarrow\langle n_r\ RT\ n_t\rangle$；
- $\forall(n_i,n_j,n_r)\langle n_i\ BT\ n_j\rangle\bigcap\langle n_i\ RT\ n_r\rangle\Rightarrow\langle n_j\ RT\ n_r\rangle$；
- $\forall(n_i,n_j,n_r,n_t)\langle n_i\ BT\ n_j\rangle\bigcap\langle n_i\ RT\ n_r\rangle\bigcap\langle n_r\ SYN\ n_t\rangle\Rightarrow\langle n_j\ RT\ n_t\rangle$
- $\forall(n_i,n_j,n_r)\langle n_i\ RT\ n_j\rangle\bigcap\langle n_j\ SYS\ n_r\rangle\Rightarrow\langle n_i\ RT\ n_r\rangle$。
- $\forall(n_i,n_j,n_r)\langle n_i\ RT\ n_j\rangle\bigcap\langle n_j\ RT\ n_r\rangle\Rightarrow\langle n_i\ RT\ n_r\rangle$
- $\forall(n_i,n_j,n_r)\langle n_i\ BT\ n_j\rangle\bigcap\langle n_j\ BT\ n_r\rangle\Rightarrow\langle n_i\ BT\ n_r\rangle$

节点名的相似性计算如表 3-1 所示。

$n\rightarrow^m n'$ 表示节点 n 和 n' 之间存在长度为 $m(m\geqslant 1)$的路径。$\sigma_{n,n'}$ 表示 $n\rightarrow^m n'$ 中术语关系的强度。

如果节点 n_i，n_j 间存在多条推导路径，计算所有路径上强度，取连接两点的所有路径中的最大强度作为节点名间相似性。

定义一个公共字典(Common Dictionary)，简称 CD，保存节点名相似性分析的结果。

$$CD=\{\langle n,n',\sigma_{n,n'}\rangle,n,n'\in N(IS_1)\cup N(IS_2)\}。$$

表 3-1 节点名的相似性系数

系数	值	条件
$\sigma_{n,n'}$	1	$n\ SYN\ n'$
	0.8	$n\ BT/NT\ n'$
	0.5	$n\ RT\ n'$
	$0\leqslant\sigma_1\times\sigma_2\times\cdots\times\sigma_{(m-1)}\leqslant 1$	$n\rightarrow^m n'$
	0	其它

3.1.2 0 近邻的相似性分析

设 $N_l\in G(IS_1)$和 $N_m\in G(IS_2)$为任意两个终端节点；设 $NSet_0(N)=\{Y_i,i=0,1,\cdots,p\}$，是通过一条弧与 N 直接相邻的节点集。为了计算 0 近邻对 N_l，N_m 相似性的影响，本节提出的方法不仅考虑了 $NSet_0(N_l)$和 $NSet_0(N_m)$间的相似性，还考虑了 $NSet_0(N_l)$中的节点与 N_l 的相似性以及 $NSet_0(N_m)$中的节点与 N_m 的相似性。0 近邻相似性函数 η_0 的具体定义如下：

$$\eta_0(N_l,N_m)=\psi(\rho(\tau(N_l,G(IS_1)),\tau(N_m,G(IS_2)))) \tag{3-1}$$

函数 $\tau:\tau(N,G(IS))$返回一个序偶(N_i,r_i)的集合，其中 $N_i\in NSet_0(N)$，r_i 是 N 到 N_i 之间弧的语义相关系数。定义为：

$$\tau(N,G(IS))=\{(N_i,r_i)\mid N_i\in NSet_0(N),\langle N,N_i,[d_i,r_i]\rangle\in A(IS)\} \tag{3-2}$$

设 $p=|\{(N_{l1},r_{l1}),\cdots,(N_{lp},r_{lp})\}|$，$q=|\{(N_{m1},r_{m1}),\cdots,(N_{mq},r_{mq})\}|$，如果 $p\neq q$，函数 $p(\{(N_{l1},r_{l1}),\cdots,(N_{lp},r_{lp})\},\{(N_{m1},r_{m1}),\cdots,(N_{mq},r_{mq})\})$负责添加$|p-q|$个虚拟节点到节点数较少的集合中。由于虚拟节点不对应源模式中的对象，为了保证虚拟节点不与任何其它节点名有语义对应关系，特别为虚拟节点分配没有意义的名字，并使连接虚拟节点的弧的权值为 0。函数 φ 将 0 近邻对节点相似性的影响转换为计算二分图 *BG* 的最大权匹配。求 0 近邻组成的二分图的最大权匹配算法如下：

基于 0 近邻节点组成的二分图，本节提出了求最大权匹配的算法。算法步骤如下：

(1)给出初始节点集 V_l 和 $NSet_0(N_l)=\{v_{li},i=1,\cdots,p\}$，$N_l\in N(IS_l)$和 $V_m=NSet_0(N_m)=\{v_{mj},j=1,\cdots,q\}$，$N_m\in N(IS_2)$，$n=\max(p,q)$，$m=\min(p,q)$，如果 $m=0$，程序结束，否则进入下一步；

(2)如果$|p-q|\neq 0$，则在节点数少的节点集合中添加$|p-q|$个虚拟节点；

(3)求出互补节点集中任意两节点间的权值，$w_{ij}=s_{ij}\times r_{ij}$，其中

$$s_{ij}=\begin{cases}\sigma_{ij}, & \langle v_{li},v_{mj},\sigma_{ij}\rangle\in CD\\ 0, & \text{其他}\end{cases}$$

$r_{ij}=\dfrac{r_{li}+r_{mj}}{2}$，与虚拟节点相连的弧的权值为 0；

(4)给出初始标号：$l(V_{li})=\max(w_{ij})$，$l(V_{mj})=0$，$i,j=1,2,\cdots,n$；

(5)求出边集 $E_l=\{\langle v_{li},v_{mj}\rangle\mid l(v_{li})+l(v_{mj})=w_{ij}\}$和等价子图 BG_l，调用匹配算法，找到 BG_l 中的一个匹配 M；

(6)如果 M 已经饱和 V_l 的所有节点，则 M 是 *BG* 的最优匹配，程序结束，否则进入下一步；

(7)在 V_l 中找一 M 非饱和点 v_{l0}，令 $A\leftarrow\{v_{l0}\}$，$B\leftarrow\phi$；

(8)如果 $N_{GI}(A)=B(N_{GI}(A)$为与 A 中节点邻接的节点集合),则转入第 12 步,否则进入下一步;

(9)找一节点 $y\in N_{GI}(A)-B$;

(10)如果 y 是 M 中的饱和点,则找出 y 的配对点 z,令 $A\leftarrow A\cup\{z\}$,$B\leftarrow B\cup\{y\}$,转第 8 步,否则进入下一步;

(11)存在一条从 v_{l0} 到 y 的可增广路 P,令 $M\leftarrow M\oplus E(P)$,转第 6 步;

(12)①按公式 3~5 计算 a 值:

$$a=\min_{\substack{v_{li}\in A\\ v_{mj}\notin N_{GI}(A)}}\{l(v_{li})+l(v_{mj})-w_{ij}\} \tag{3-3}$$

②修改标号:

$$l'(v)=\begin{cases}l(v)-a, & v\in A\\ l(v)+a, & v\in B\\ l(v), & \text{其它}\end{cases} \tag{3-4}$$

(13)根据新的 l' 求 $E_{l'}$ 及 $BG_{l'}$;

(14)$l\leftarrow l'$,$BG_l\leftarrow BG_{l'}$,转第 8 步。

求匹配的算法步骤如下:

(1)初始化 $i=1$,$M=\phi$;

(2)将满足 $l(v_{li})+l(v_{mj})=w_{ij}\neq 0$,$j=1,\cdots,n$ 的节点 v_{mj} 置于候选集合 $CM=\{v_{mi},\cdots,v_{mp}\}$ 中,如果不存在满足条件的节点,则转入第 5 步;

(3)如果 CM 中存在没有处理过的节点,则任取一点 v,并将其打上"已经处理"的标识;

(4)比较 $>i$ 的行中是否存在 $>w_{ij}$ 的行标号,并且与之相连的节点是 v,如果不存在,则 $M=M\cup\{v_{li},v\}$,进入下一步;否则,$CM=CM-v$,转入第 3 步;

(5)$i=i+1$,如果 $i\leqslant n$,则转入第 2 步,否则,程序结束。

根据最后求得的最大权匹配,函数 ψ 定义为:

$$\psi=\begin{cases}\dfrac{\sum_{(v_{li},v_{mj},w_{ij})\in E_l}w_{ij}}{|E_l|}, & |E_l|\neq 0\\ 0, & |E_l|=0\end{cases} \tag{3-5}$$

3.1.3 基本语义相似性字典的建立

节点间的基本语义相似性与节点名相似性分析和 0 近邻相似性分析有关,具体定义如下:

$$bs_{lm}=w_{NA}\times\sigma_{N_l,N_m}+w_{SA}\times\eta_0(N_l,N_m) \tag{3-6}$$

w_{NA},$w_{SA}\in[0,1]$且 $w_{NA}+w_{SA}=1$。w_{NA}定义为:

$$w_{NA}=\begin{cases}0, & (\langle N_l,N_m,\sigma_{N_l,N_m}\rangle\notin CD)\cap((NSet_0(N_l)\neq\phi)\\ & \cap(NSet_0(N_m)\neq\phi))\\ 0.5, & (\langle N_l,N_m,\sigma_{N_l},N_m\rangle\in CD)\cap((NSet_0(N_l)\neq\phi)\\ & \cap(NSet_0(N_m)\neq\phi))\\ 1, & ((NSet_0(N_l)\neq\phi)\cap(NSet_0(N_m)\neq\phi))\\ & =false\end{cases} \tag{3-7}$$

可见,如果 N_l 或 N_m 是原子节点,那么与 N_l 和 N_m 的基本近似性就仅仅取决于节点间名字的相似性了。

计算得到的节点间基本语义相似性保存在基本语义相似性字典(Basic Semantic Similarity Dictionary,简称 BSSD)中。

$$BSSD=\eta_B=\{N(IS_1),N(IS_2)\}=\{\langle N_l,N_m,bs_{lm}\rangle\mid N_l\in N(IS_1),N_m\in N(IS_2)\} \tag{3-8}$$

3.2 精确相似性分析

基于前一节得到的基本相似性,进一步分析节点近邻的相似性,从而提取节点间精确的语义相似性。节点近邻的相似性对节点相似性的影响与它们和节点的语义距离有关,第 i 个近邻与节点的语义距离$\geqslant i$ 且$<(i+1)$。语义距离越近,对节点的影响就越大。为了形式化地表达这种影响,本节定义了一个单调递减函数加权第 i 个近邻的相似性,使得离节点最远的近邻有最轻的影响。

3.2.1 i 邻域相似性分析

本节采用迭代的方法计算 i 近邻相似性对节点相似性的影响,算法的核心思想是:以基本语义相似性字典中存储的元组为起点,用 i 近邻中的目标节点构造二分图,并求其最大权匹配,作为 i 邻域的语义相似性。使用单调递减函数调整 i 近邻的语义相似性对节点相似性的影响力,利用节点近邻的语义相似性以及节点第 $i-1$ 次的语义相似性的加权累加和决定第 i 次的语义相似性。反复迭代,直到其中某个节点的 i 邻域为空。

设节点 $N\in N(IS)$ 的 i 近邻:$NSet_i(N)=\{y_j,j=0,\cdots,p\}$是 $nbh(N,i)$中的弧的目标节点集。i 近邻相似性函数 η 的具体定义如下:

$$\begin{cases}\eta^0(ESSD)=\eta_B\\ \eta^i(ESSD)=\eta(\eta^{i-1}(ESSD),i-1) & i>0,\\ \quad((NSet_i(N_l)\neq\phi)\cup(NSet_i(N_m)\neq\phi))\end{cases} \tag{3-9}$$

其中,$\eta(ESSD,i)=\{\langle N_l,N_m,\phi_i(ESSD,N_l,N_m,bs_{lm},i)\rangle\mid\langle N_l,N_m,bs_{lm}\rangle\in ESSD\}$。

$$\phi_i(ESSD,N_l,N_m,bs_{lm},i)=\begin{cases}bs_{lm}, & i=0\\ \theta(i)\times\phi_i(\rho_i(\tau_i(N_l,N(IS_1)),\tau_i(N_m,N(IS_2))))+(1-\theta(i))\\ \quad\times\phi_{i-1}\text{若}(NSet_i(N_l)\neq\phi)\cap(NSet_i(N_m)\neq\phi)\\ \phi_{i-1}, & \text{其它}\end{cases} \tag{3-10}$$

函数 τ_i 返回序偶(N_i,r_i)的集合:

$$\tau_i(N,i)=\{(N_j,r_j)\mid N_j\in NSet_i(N,i),r_j=\mathfrak{F}(\llcorner N,N_j\lrcorner_{\langle N_k,N_j,t_{kj}\rangle})\} \tag{3-11}$$

r_j 是 N 到 N_j 之间的路径语义相关系数。

与上节相同,函数 ρ 负责添加虚拟节点到节点数较少的集合中,虚拟节点名被设置成没有含义的,与虚拟节点相连的弧的权值为 0,其它弧的权值被设置为:$w_{ij}=\sigma_{li,mj}\times\dfrac{r_{li}+r_{mj}}{2}$。

函数 ψ 将计算 i 近邻相似性转换为求基于 $NSet_i(N_l)$和 $NSet_i(N_m)$中的节点组成的二分图的最大权匹配。基于 SDS-G,求节点间精确语义相似性的算法步骤如下:

(1)如果 $N(IS_1)$中存在没有计算相似性的节点,任取其中一节点 $N_l\notin N_A$,进入第 2 步,否则程序结束;

(2)如果 $N(IS_2)$中还有节点与节点没有组成序偶,则任取其中一点 N_m,如果 $N_m\in N_A$,将两者之间的相似性置 0,并为节点 N_m 打上"已经处理"的标识,转入第 2 步,否则进入下一步;如果 $N(IS_2)$所有的节点都标上"已经处理"的标志,则清除这些标志,转入第 1 步;

(3)初始化 $i=1$;

(4)求出 $Nset_i(N_l)$,$Nset_i(N_m)$,设 $Nset_i(N_l)=Nset_i(N_l)-\{$已经处理过的节点$\}$,$Nset_i(N_m)=Nset_i(N_m)-\{$已经处理过的节点$\}$;

(5)如果$(Nset_i(N_l)\neq\phi)\cap(Nset_i(N_m\neq\phi,)$,则调用上节给出的最大权匹配算法,并为集合中的节点打上"已经处理"的标识,否则转入第 8 步;

(下转第 139 页)

同其它核分析方法一样，选择恰当的核函数将是一个困难的问题。这往往需要对特定的分类问题的深入理解，在此基础上才能选择出合理的核函数。从表 2 中可以看到不同的核函数对分类性能产生了极大的影响，不恰当的核函数将造成分类性能下降而不是提高，如何选择恰当的核函数是需要进一步研究的问题。

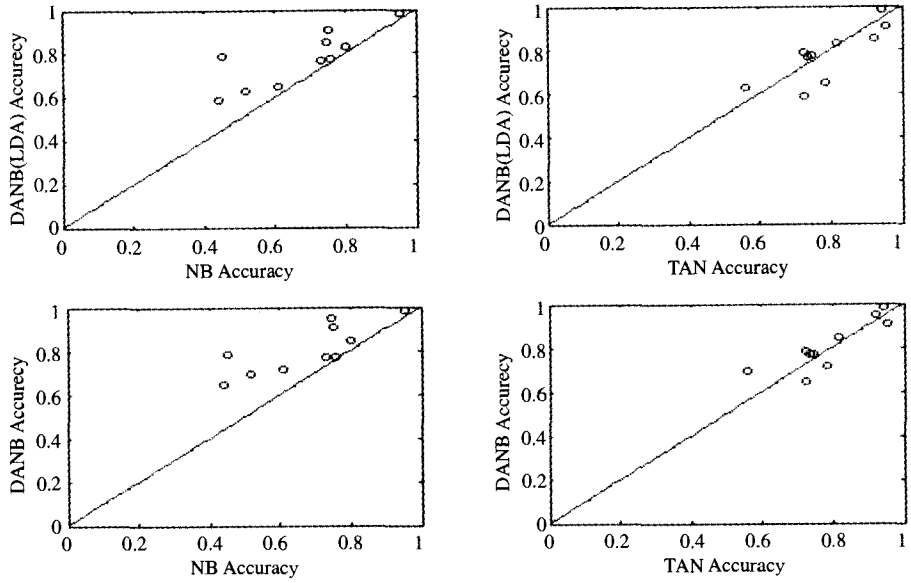

图 1　DANB、NB 和 TAN 分类精度对比散点图

注：对角线以上的点表示 DANB(DANB(LDA))的分类精度大于对比项的分类精度。

参 考 文 献

1 Fried N, Geiger D, Goldszmidt M. Bayesian network classifiers. Machine Learning, 1997, 29(2-3):131～163
2 Langley P, Iba W, Thompson K. An analysis of Bayesian classifiers. In: Proc. of the 10th National Conf on Artificial Intelligence. Menlo Park: AAAI Press, 1992. 223～228
3 Kononenko I. Semi Bayesian classifier. In: Proc. of the 6th European Working Session on Learning. New York: Springer-Verlag, 1991. 206～219
4 Pazzani M J. Searching for dependencies in Bayesian classifiers. In: Learning from Data: Artificial Intelligence and Statistics V. New York: Springer-Verlag, 1996. 239～248
5 Langley P, Sage S. Induction of selective Bayesian classifiers. In: Proc. of the 10th Conf. on Uncertainty in Artificial Intelligence. San Francisco: Morgan Kaufmann Publishers, 1994. 399～406
6 Webb G I, Pazzani M J. Adjusted probability naive Bayesian induction. In: Proc. of the 11th Australian Joint Conf. on Artificial Intelligence. Berlin: Springer-Verlag, 1998. 285～295
7 Kohavi R. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. In: Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining. Menlo Park: AAAI Press, 1996. 202～207
8 Keogh E J, Pazzani M J. A comparison of Distribution-based and Classification-based Approaches. In: Proc. of the Uncertainty' 99: The 7th Int'l Workshop on Artificial Intelligence and Statistics. San Francisco: Morgan Kaufmann Publishers, 1999. 225～230
9 Cheng J, Greiner R. Comparing Bayesian network classifiers. In: Proc of the 15th Conf on Uncertainty in Artificial Intelligence. San Francisco: Morgan Kaufmann Publishers, 1999. 101～108
10 Chickering D M, Geiger D, Heckerman D. Learning Bayesian networks is NP-complete. In: Learning from Data: Artificial Intelligence and Statistics V. New York: Springer-Verlag, 1996. 121～130
11 Xu Y, Yang J, Jin Z. Theory analysis on FSLDA and ULDA. Pattern Recognition, 2003, 36(12): 3031～3033
12 Shawe-Taylor J, Cristianini N. Kernel Methods for Pattern Analysis. Cambridge, UK, New York : Cambridge University Press, 2004
13 Baudat G, Anouar F. Generalized Discriminant Analysis Using a Kernel Approach. Neural Computation, 2000, 12: 2385～2404
14 http://www. ics. uci. edu/～ mlearn/MLRepository. html. Irvine, CA: University of California, Department of Information and Computer Science
15 Johnson R A, Wichern D W. 实用多元统计分析. 陆璇 译. 北京：清华大学出版社，2001

(上接第 101 页)

(6)根据公式(3-12)求出第 i 次的 ϕ_i；

(7)$i=i+1$，转入第 4 步；

(8)将节点上"已经处理"的标识清除；

(9)转入第 2 步。

$\theta(i)$是单调递减的，保证距离 N_l，N_m 越远的近邻对 N_l，N_m 相似性的影响越小。

通过精确语义相似性分析，最终得到的结果保存在精确语义相似性字典中(Exact Semantic Similarity Dictionary，简称 ESSD)。

$$ESSD=\eta^{\infty}=\{\langle N_l,N_m,\phi\rangle \mid N_l\in N(IS_1),N_m\in N(IS_2)\} \quad (3\text{-}12)$$

结论　在从半结构化的信息源建立本体的过程中，为了统一地分析和处理不同的信息源，本文提出了一个统一的概念模型，将各信息源转换为 SDS-G。该模型不仅完整地表现了各信息源的内涵，在计算语义相关性和语义距离时，还考虑了外延的影响。基于转换后得到的 SDS-G 模型，以节点名、节点属性、节点近邻为比较特征，使用节点名相似性分析方法、0 近邻相似性分析方法计算节点的基本语义相似性分析，然后使用近邻相似性分析方法修正已得到的相似性，提炼模式间更精确的语义相似性。本文提出的方法还解决了相似性分析中类型冲突的问题，方法是半自动化的，仅在早期需要少量的人类专家的参与。本文提出的方法在实验中也取得了较好的效果。

参 考 文 献

1 Gruber T. What is an Ontology? http://www-ksl. stanford. edu/kst/whst-is-an-ontology. html, 28/01/2002
2 Bergamaschi S, Castano S, Vincini M. Semantic integration of semistructured and structured data sources. SIGMOD Rec, 1999, 28(1):54～59
3 Buneman P, Davidson S, Fernandez M, et al. Adding structure to unstructured data. In: Proc. of International Conference on Database Theory (ICDT'97), 1997. 336～50