第 43 卷 第 12 期 计 算 机 工 程 2017 年 12 月
Vol. 43 No. 12 Computer Engineering December 2017

·人工智能及识别技术· 文章编号:1000-3428(2017)12-0165-08 文献标志码:A 中图分类号:TP301.6

Web 表格的实体列发现算法

张丽方,王 宁,齐 飞

(北京交通大学 计算机与信息技术学院,北京 100044)

摘 要: 针对机器无法理解 Web 表格语义信息的问题,传统的实体列发现方法通常依靠表头信息和知识库发现实体列,不适用于没有表头的 Web 表格。为此,提出一种基于列值间近似依赖关系和规范化的 Web 表格实体列发现算法,对无表头或者无法恢复出完整表头的表格甚至多实体列表格进行实体列标注。由 Web 表格中的属性值探测出 Web 表格属性间内在的近似函数依赖关系,根据 Web 表格的特点对噪声函数依赖进行删减,通过函数依赖集进行规范化,得到 Web 表格的实体列。与利用知识库进行实体列探测的算法相比,该算法不依赖表头信息,召回率和精确度均提高了 3% ~5%,适用性更强。

关键词: Web 表格;实体列;近似函数依赖;语义恢复;规范化

中文引用格式:张丽方,王 宁,齐 飞. Web 表格的实体列发现算法[J]. 计算机工程,2017,43(12):165-172.

英文引用格式:ZHANG Lifang,WANG Ning,QI Fei. Entity Column Discovery Algorithm of Web Table[J]. Computer Engineering,2017,43(12):165-172.

Entity Column Discovery Algorithm of Web Table

ZHANG Lifang,WANG Ning,QI Fei

(School of Computer and Information Technology,Beijing Jiaotong University,Beijing 100044,China)

【Abstract】 Semantic information for Web tables is not understood by machines. Traditional entity column detection methods find entity columns with header information and knowledge base. They are not applicable for tables without headers. This paper proposes an entity column discovery algorithm of Web table based on column value of approximate functional dependencies and normalization, which is used to annotate entity column for tables that have no header or cannot restore a full header even multiple entity column tables. The approximate function dependency relations between Web table attributes are detected according to attribute values in Web tables. The noisy function dependency relations are filtered according to the characteristics of Web tables. The entity columns of the Web table are obtained by normalization of the function dependency set. Compared with entity column detection algorithm based on knowledge base, the proposed algorithm is independent of header information, 3% ~5% higher in precision and recall, and can be applied in more scenes.

【Key words】 Web table;entity column;approximate functional dependency;semantic recovery;normalization

DOI:10.3969/j.issn.1000-3428.2017.12.031

0 概述

随着信息技术的发展,互联网上的资源越来越丰富,除了非结构化数据外,还有大量的网络表格存在。这些网络表格较文本而言,具有更好的结构化特性,因此受到人们的极大关注。2008 年 Google 公司启动 WebTable 项目[1],研究如何更好地抽取和利用网络上广泛存在的结构化数据[2]。2015 年 Google 公司将 Web 表格搜索技术应用到他们的搜索引擎,同时也指出如何让机器更好地理解 Web 表格的语义仍是未来提高表格搜索覆盖率和准确率的重大挑战[3]。

为了让机器更容易地理解来自网络的表格数据,文献[4]采用机器学习的技术标注 Web 表格的列类型和两列之间的语义关系。文献[5]利用知识库对 Web 表格进行列概念探测。Microsoft 公司的研究人员也利用知识库对 Web 表格进行语义恢复,完成表头的恢复以及实体列的探测[6-7]。实体列能够标识 Web 表格所描述的实体,其列标签描述了整张 Web 表格的主题,通过它可以确定 Web 表格的语义信息。如果准确地探测 Web 表格的实体列,就可以大大提升机器对 Web 表格语义的理解程度。

基金项目:国家自然科学基金(61370060)。

作者简介:张丽方(1991—),女,硕士研究生,主研方向为 Web 数据集成、数据挖掘;王 宁(通信作者),教授、博士;齐 飞,硕士研究生。

收稿日期:2016-11-17 **修回日期:**2016-12-19 **E-mail:**14120447@bjtu.edu.cn

目前对 Web 表格实体列检测方面的研究很少，最具有代表性的研究是文献[6]的研究。尝试将 Probase 作为知识库，依赖 2 个证据实现 Web 表格的实体列发现，该方法又称为基于证据的实体列发现算法。依据的证据是：首先，实体列作为 Web 表格的描述主题，可以向上抽象为一个具体概念；其次，其他非实体列的列标签应该与实体列列标签存在属性关系。

值得注意的是，文献[6]提出的方法基于如下假设：1）Web 表格都具有表头或者通过语义恢复技术可以恢复出表头；2）Web 表格中所有非实体列都与实体列存在属性关系，且这种概念-属性关系在 Probase 库中存在；3）Web 表格中只存在一个实体列。事实上，一方面，由于 Probase 库很难覆盖网络上全部的实体、属性、概念以及它们之间的关系，对于 Web 表格中的某些列，仅靠知识库无法恢复其表头，例如数字、日期等列。另一方面，网络上的许多表格不止一个实体列，文献[6]也指出实体列探测失败的表格大部分是拥有 2 个或者更多实体列的表格。文献[8]提出的方法虽然可以发现多个实体列，但是该方法和文献[6]的研究一样，都是依靠 Web 表格的表头和知识库进行实体列探测。这 2 种算法都无法对没有表头的 Web 表格或者利用语义恢复也恢复不出表头的表格进行实体列发现。

针对上述问题，本文提出一种基于近似函数依赖的 Web 表格实体列发现算法（AFD_Model）。其主要设计思想是：首先，根据 Web 表格中的属性值探测出 Web 表格属性间内在的函数依赖关系；然后，根据 Web 表格的特点，对噪声函数依赖进行删减；最后，根据函数依赖集进行规范化，得到 Web 表格的实体列。

1 Web 表格及其实体列

定义 1 对于一张 Web 表格，如果其中的一列或者多列能标识 Web 表格所描述的实体，则将这一列或者多列定义为实体列，实体列以外的其他列定义为属性列。

如表 1 中的第 1 列和第 4 列代表了整个表 1 所描述的实体是作者（Author）和书（book），则表 1 的实体列就是作者（Author）列和选择的书（Selected book）列，其余列为属性列。其中第 2 列和第 3 列负责描述 Author 实体列，第 5 列负责描述 Selected book 实体列。

表 1 作者和书籍属性列表

Author	Sex	Nationality	Selected book	Theme
Douglas Adams	M	English	Great Ape Project	Conservation movement
Lawrence Anthony	M	South African	Babylon's Ark	Conservation movement
Lady Eve Balfour	F	English	The Living Soil	Organic movement
E. F. Schumacher	M	British	Small Is Beautiful	Appropriate technology
Lawrence Solomon	M	Canadian	The Deniers	Various themes
David Suzuki	M	Canadian	The Sacred Balance	Various themes

与文献[6]相比，AFD_Model 算法弥补了文献[6]不能对多实体列 Web 表格进行实体列探测的缺陷。与文献[8]相比，AFD_Model 算法不依赖表头和知识库，适用性更强。AFD_Model 算法可以根据属性列间的关系发现 Author 和 Selected book 2 个实体列，而基于证据的实体列发现算法[6]由于该表格存在多个实体列，并非所有属性均与其中某列存在概念-属性关系，导致探测实体列失败。

本文拟解决的问题为给定 Web 表格 T（可以不包含表头），找出 T 中所有的实体列。

2 基于近似函数依赖的实体列发现

在关系表中，函数依赖[9]（Functional Dependency，FD）$X\rightarrow A$ 表示了属性组 X 和 A 之间的一种数据依赖关系，这种关系要求若 2 个实体 X 中的值相同，那么对应 A 中的值也必须相同。函数依赖是关系数据库中的一个重要概念，它为属性间的关系提供了语义线索。然而 Web 表格不像关系型数据库表格那样具有规范性。Web 表格不严格遵循关系型数据库中范式的约束，而且表中往往有噪声存在。因此，怎样排除噪声数据的干扰，获取优质的函数依赖集是本文研究的重点。

为了利用函数依赖检测数据源中的脏数据，文献[10]提出概率函数依赖概念。将概率函数依赖的形式定义为 $X \xrightarrow{p} A$，其中，p 为 $X\rightarrow A$ 成立的可能度，p 的计算方法如式（1）所示。D_x 为 X 属性上不同属性值的集合，$x \in D_x$。在 X 属性值中，相同的值 x，在 A 属性中可能存在不同的值与其相对应，a_x 为 A 中个数最多的值。

$$\Pr(X\rightarrow A)=\frac{\sum_{x\in D_x}|x,a_x|}{|D_x|} \tag{1}$$

其中，$|x,a_x|$ 表示 X 中属性值取 x 同时 A 属性值取 a_x 的元组数目。

Web 表格中内容可能出现错误值，比如将“nba Jordan”写成“Jardan nba”等错误，所以在进行实体列发现时必须考虑表格内容写错的情况，并重新设计适用于 Web 表格的近似函数依赖的计算方法。

定义 2 设 X 是 Web 表格 T 中的某个属性，A

是 T 中不同于 X 的属性。当 T 中存在部分元组的 (X,A) 属性值对,使得 $X\rightarrow A$ 成立,则称 X 近似函数确定 A 或 A 近似函数依赖于 X,记作 $X \xrightarrow{\sim} A$。$p(X \xrightarrow{\sim} A)$ 表示 $X\rightarrow A$ 在 T 上成立的可能性,即近似函数依赖概率。(X,A) 属性值对中使得 $X\rightarrow A$ 成立的数据称为一致性数据,其余称为不一致性数据。

利用近似函数依赖,探测 Web 表格属性间的函数依赖关系,进而发现实体列。具体做法分为以下 3 步:

步骤 1　获取候选近似函数依赖集。此阶段的主要任务是根据 Web 表格的特点量化所有可能的函数依赖,得到候选近似函数依赖集合。

步骤 2　删减噪声近似函数依赖。该阶段的主要任务是根据 Web 表格的特点,从候选近似函数依赖集合中删减噪声函数依赖,从而得到能更加准确地表达 Web 表格间函数依赖关系的近似函数依赖集合。

步骤 3　语义规范化。这一阶段根据 Armstrong 公理系统对 Web 表格进行规范化,从而获得 Web 表格的实体列。

2.1　候选函数依赖集的获取

Web 表格可能出现错误或噪声数据,此阶段的主要任务是将这些错误或噪声对近似函数依赖计算的影响降到最小。为了准确获取属性间的函数依赖关系,在计算所有可能的近似函数依赖时,必须考虑函数依赖一致性数据的可靠性和函数依赖不一致性数据的误写可能性;此外,在过滤噪声函数依赖时也必须考虑到 Web 表格中函数依赖的特点。

2.1.1　一致性数据

在 Web 表格 T 中,对于 X 属性值为 v_x 的元组,其 A 属性列中可能存在不同的值,假设该不同值的集合为 V_A。

如果集合 V_A 中个数最多的值唯一,则将该值作为一致性数据,如果个数最多的值不唯一,则将这些个数最多的值分别作为类中心,计算其他值和类中心值相似度的和,选择和最大时的类中心值 v_a 作为一致性数据。对于任意类中心值 v_j,具体计算方法如式(2)所示。

$$v_a = \arg\max_{v_j} \sum_{v_i \in V_A} Sim(v_j, v_i) \tag{2}$$

例如,X 列中值为 U. S. A 的元组,其 A 属性列中存在不同的值。显然,如果出现表 2 中的情况,直接将 Washington 作为一致性数据,将 New York 作为不一致性数据。如果出现表 3 中的情况,个数最多的值为 Washington 和 New York,则分别将 Washington 和 New York 作为类中心值,求其他值与类中心值的相似度之和。相似度之和最大的类中心值 Washington 作为一致性数据,其余数据作为不一致性数据。

表 2　不一致数据 1

X	A
U. S. A	Washington
U. S. A	Washington
U. S. A	New York

表 3　不一致性数据 2

X	A
U. S. A	Washington
U. S. A	Washington
U. S. A	Washingtan
U. S. A	New York
U. S. A	New York

所有不同 X 值在 A 属性列中对应的一致性数据的集合构成了整张表格中使得 $X\rightarrow A$ 成立的一致性数据。

2.1.2　一致性数据因素

一致性数据所占比例越大,说明 $X\rightarrow A$ 成立的可能性越大,则一致性数据对 $X\rightarrow A$ 成立的支持度越高,同时一致性数据所占比例越大,说明该一致性数据为真正一致性数据的可靠性越大。X 中值为 v_x 的所有元组,其中的一致性数据 v_a 对 $X\rightarrow A$ 成立的支持度和一致性数据的可靠性均为:

$$S_c(X\rightarrow A, V_X, V_{A'}) = \frac{|V_X, V_{A'}|}{|V_X|} \tag{3}$$

其中,$V_X = \{X.r \mid X.r = v_x\}$,$V_{A'} = \{A.r \mid X.r = v_x \& A.r = v_a\}$,$|V_X, V_{A'}| = |\{\langle X.r, A.r\rangle \mid X.r = v_x \ \& A.r = v_a\}|$,$X.r$ 为 X 列 r 行单元格的值,$A.r$ 为 A 列 r 行单元格的值。

2.1.3　不一致性数据因素

不一致性数据和一致性数据越相似,且一致性数据的可靠性越大,则不一致性数据对 $X\rightarrow A$ 成立的支持度越大。关于不一致性数据和一致性数据相似度的计算方法,采用一种类似于文献[11]提出的模糊字符匹配的字符串相似度计算方法。给定字符串 x 和 y(x 和 y 为表格中字符串类型的单元格的值),以单词为单位分别将 x 和 y 切割成单词集合 $S_1 = \{x_1, x_2, \cdots, x_m\}$ 和 $S_2 = \{y_1, y_2, \cdots, y_n\}$,将 2 个集合中的值作为二部图的顶点,2 个顶点间的边上的值为这 2 个顶点间的相似度(本文采用式(4)的相似度计算方法)。只保留相似度大于阈值 σ 的边。最后求得这 2 个集合的最大二部图匹配作为集合 S_1 和 S_2 的模糊覆盖,记作 $S_1 \overset{\sim}{\underset{\sigma}{\cap}} S_2$。根据文献[11]的 FConsine 相似度计算公式,得到式(5)。

$$P_{LCSim}(a,b) = L_c(a,b)/L_{\max}(a,b) \tag{4}$$

$$FConsine(x,y) = \frac{|S_1 \overset{\sim}{\underset{\sigma}{\cap}} S_2|}{\sqrt{|S_1| \times |S_2|}} \tag{5}$$

其中，$L_c(a,b)$表示单词 a 和单词 b 的最长公共子序列，$L_{\max}(a,b)$表示 a 和 b 两者中较长单词的字符串长度。

例如 x = “nba Jordan”，y = “Jardan nba”。x 和 y 切割后得到 $S_1 = \{\text{nba}, \text{Jordan}\}$，$S_2 = \{\text{Jardan}, \text{nba}\}$。然后利用式(4)计算 2 个集合间单词对的相似度为：$P_{LCSim}(\text{nba},\text{nba}) = 1$，$P_{LCSim}(\text{nba},\text{Jardan}) = 1/6$，$P_{LCSim}(\text{Jordan},\text{Jardan}) = 5/6$，$P_{LCSim}(\text{Jordan},\text{nba}) = 1/6$。因此，这 2 个集合间的模糊覆盖 $S_1 \underset{0.8}{\tilde{\cap}} S_2$ 为 $\{P_{LCSim}(\text{nba},\text{nba}) = 1, P_{LCSim}(\text{Jordan},\text{Jardan}) = 5/6\}$，其值为 $|S_1 \underset{0.8}{\tilde{\cap}} S_2| = 1 + 5/6 = 11/6$。最后求得 $FConsine(x,y) = 11/12$。

定义 X 中值为 v_x 的所有元组中不一致性数据的集合 V_{A*} 对 $X \rightarrow A$ 成立的支持度为：

$$S_{nc}(X \rightarrow A, V_X, V_{A*}) = \frac{1}{|V_X|} \sum_{v_i \in V_{A*}} FConsine(v_a, v_i) \times S_c(X \rightarrow A, V_X, V_{A'}) \tag{6}$$

其中，$V_{A*} = \{A.r \mid X.r = v_x \ \&\ A.r \neq v_a\}$。

2.1.4 近似函数依赖的评分

集合 V_X 对 $X \rightarrow A$ 成立的支持度可以通过一致性数据和不一致性数据的加权平均和表示，记为 $P(X \overset{\sim}{\longrightarrow} A, V_X)$，如式(7)所示。

$$P(X \overset{\sim}{\longrightarrow} A, V_X) = \omega_1 S_c(X \rightarrow A, V_X, V_A) + \omega_2 S_{nc}(X \rightarrow A, V_X, V_{A*}) \tag{7}$$

其中，$\omega_1 + \omega_2 = 1$。

最后，取 X 中所有不同 v_x 元组的支持度，将它们的平均值 $P(X \overset{\sim}{\longrightarrow} A, T)$ 作为 Web 表格 T 中 $X \rightarrow A$ 成立的概率，如式(8)所示。

$$P(X \overset{\sim}{\longrightarrow} A, T) = \frac{\sum_{V_X \in D_X} P(X \overset{\sim}{\longrightarrow} A, V_X)}{|D_X|} \tag{8}$$

其中，$|D_X|$表示 X 中有区别的 V_X 的个数。

2.2 候选函数依赖集的删减

根据 2.1 节中的计算公式，得到的候选函数依赖集包含了 Web 表格 T 中任意属性之间的近似函数依赖，是所有可能的函数依赖的全集。事实上，这些近似函数依赖集中存在很多噪声函数依赖，这一节将根据 Web 表格的特点设计规则删减掉噪声近似函数依赖。

如果 $X \overset{\sim}{\longrightarrow} A$ 满足以下 3 条规则中的任一条，就将 $X \overset{\sim}{\longrightarrow} A$ 从候选近似函数依赖集中删去。

规则 1 若 X 列的属性值的类型为日期类型、浮点类型或者布尔类型。

规则 2 若在 T 中存在属性 Y，使得 $P(Y \overset{\sim}{\longrightarrow} A, T) > P(X \overset{\sim}{\longrightarrow} A, T)$ 成立。

规则 3 若在候选近似函数依赖集中，存在这样的属性 X 和 A，使得 $P(A \overset{\sim}{\longrightarrow} X, T) > P(X \overset{\sim}{\longrightarrow} A, T)$ 且 $P(X \overset{\sim}{\longrightarrow} A, T) < \theta$。

对于规则 1，考虑到日期类型、浮点类型、布尔类型的列在 Web 表格中不可能是实体列。

对于规则 2，考虑到一般情况下，Web 表格的一个属性列只负责描述一个实体列，如表 1 中的 Theme 属性列只描述 Selected book 实体列，所以每个属性列应该由它描述的实体列函数确定。因此利用规则 2 过滤掉函数依赖成立可能性比较小的近似函数依赖。

在规则 3 中考虑到若某些属性相互函数确定，则可能是这些属性均为实体列，或者实体列函数依赖某个属性。一般来讲，Web 表格中实体列近似函数依赖属性列的概率值比较小，为了排除后者的情况，设置阈值 θ 过滤这种噪声近似函数依赖。经过多次实验得到，在 Wiki Table 数据集上，θ 取 0.8 时，得到的 Web 表格近似函数依赖集最理想。因此，在实验中 θ 均取 0.8。

2.3 基于规范化的实体列发现

基于这样的事实：Web 表格中属性列近似函数依赖于它所描述的实体列，所以根据关系数据库理论的规范化原理[12]，将 2.2 节得到的近似函数依赖集进行 3NF 规范化。最后产生的主键集合就是所要的实体列[13-15]。

首先，将 2.2 节得到的近似函数依赖集的关系映射到关系矩阵 $\boldsymbol{FD}[m][n]$；将决定属性间的近似函数依赖关系映射到关系矩阵 $\boldsymbol{KK}[m][m]$。其中，m 是位于近似函数依赖蕴含左边的属性数目，即决定属性数，n 为 Web 表格中所有属性列的数目。为了方便，用不同的数字表示属性间的不同关系，矩阵中元素产生如下：

1) $\boldsymbol{FD}[m][n]$的元素产生如下：

设 $\alpha \in$ {决定属性集}，$\beta \in$ {所有列属性集}，有：

(1) 如果 $\alpha = \beta$，则 $\boldsymbol{FD}[\alpha][\beta] \leftarrow 2$；

(2) 如果 $\alpha \overset{\sim}{\longrightarrow} \beta$，则 $\boldsymbol{FD}[\alpha][\beta] \leftarrow 1$；

(3) 其他情况，则 $\boldsymbol{FD}[\alpha][\beta] \leftarrow 0$。

2) $\boldsymbol{KK}[m][m]$的元素产生如下：

设 $\alpha, \gamma \in$ {决定属性集}，有：

(1) 如果 $\alpha = \gamma$ 或者 $\alpha \overset{\sim}{\longrightarrow} \gamma$，则 $\boldsymbol{KK}[\alpha][\gamma] \leftarrow 1$；

(2) 其他情况，则 $\boldsymbol{KK}[\alpha][\gamma] \leftarrow -1$。

为方便描述，给出近似传递函数依赖的定义如下：

定义 3 在 Web 表格 T 中，如果 $X \overset{\sim}{\longrightarrow} Y$，$Y \overset{\sim}{\not\longrightarrow} X$，$Y \overset{\sim}{\longrightarrow} Z$，则称 Z 对 X 近似传递函数依赖，记为 $X \overset{Y}{\longrightarrow} Z$，其中 Y 为近似传递函数依赖的中介键。

根据 Armstrong 公理系统，$\boldsymbol{FD}[m][n]$和 $\boldsymbol{KK}[m][m]$，

得到近似函数依赖闭包关系矩阵,根据 3NF 规则获得 Web 表格的实体列集合 $Entity\{e_1,e_2,\cdots\}$。

发现实体列算法描述如算法 1、算法 2 所示。

算法 1　函数依赖集的闭包求解算法

输入　$\boldsymbol{FD}[m][n]$,$\boldsymbol{KK}[m][m]$

输出　$\boldsymbol{DC}[m][n]$

```
while(DC[m][n]发生变化)
    for each 函数依赖 α ~→ β in KK[m][m]
        If β ~→ γ in FD[m][n]
            DC[m][n]←β
Output(DC[m][n])
```

算法 2　探测实体列算法

输入　$\boldsymbol{DC}[m][n]$,$\boldsymbol{FD}[m][n]$

输出　$Entity\{e_1,e_2,\cdots\}$

```
for each DC[i][j] in DC[m][n]
    if DC[i][j]! ={0,1,2} & &FD[i][j]=1 & &FD[j][i]=1
        DC[i][j]←1
for eachDC[i][j] in DC[m][n]
    if DC[i][j]! ={0,1,2}
        Entity{}←DC[i][j]
    if DC[i][j]=1 & &i! =j
        Entity{}←i/*i 为 i 行的决定属性*/
Output (Entity)
```

算法 1 描述了如何根据近似函数依赖集寻找近似函数依赖集闭包的过程。在算法 2 中修正被误标记的近似传递依赖(行 1 ~ 行 3)。最后,将中介键和只存在直接近似函数依赖中的决定属性作为实体列输出(行 4 ~ 行 9)。

3　实验结果与分析

3.1　实验设置

实验环境:硬件环境 Intel Core i5-4590 CPU@3.30 GHz,内存 4 GB,操作系统 Windows 7;开发工具 Eclipse 4.2,语言 Java。

实验数据:Wiki Table 数据集和 Web Table 数据集。Wiki Table 数据集是当今主流的开源数据集,从维基百科中抽取得到表格数据,方便与其他算法进行对比;Web Table 数据集从目前流行的开放数据源中抽取得到。

按照数据集的行数,将数据集分为大表数据集(100 行以上),简称 L 数据集;中表格数据集(50 行 ~ 100 行),简称 M 数据集;小表数据集(10 行以下),简称 S 数据集。L 数据集、M 数据集和 S 数据集的统计特征如表 4 ~ 表 6 所示。为方便进行单实体列和多实体列发现的实验,将 L 数据集分成 L 单实体集(Wiki_LS 和 Web_LS)和 L 多实体集(Wiki_LM 和 Web_LM);M 数据集分成 M 单实体集(Wiki_MS 和 Web_MS)和 M 多实体集(Wiki_MM 和 Web_MM);S 数据集分成 S 单实体集(Wiki_SS 和 Web_SS)和 S 多实体集(Wiki_SM 和 Web_SM)。

表 4　S 数据集的特征统计

统计对象	最小值	最大值	平均值
行数	3	10	6.45
列数	3	6	4.00
实体数	1	2	1.18

表 5　M 数据集的特征统计

统计对象	最小值	最大值	平均值
行数	52	100	76.48
列数	3	8	5.43
实体数	1	2	1.39

表 6　L 数据集的特征统计

统计对象	最小值	最大值	平均值
行数	156	1 244	403.30
列数	4	12	9.74
实体数	1	3	1.65

3.2　单实体列 Web 表格的实体列发现

为了评估提出的近似函数依赖对单实体列发现的有效性,将基于近似函数依赖的实体列发现算法 AFD_Model与文献[6]提出的 ED_Model 算法、文献[8]提出的 PG_Model 算法进行实验对比。实验在 Wiki Table 数据集上进行,该数据集本身带有完整的表头。AFD_Model 算法不依赖于表头信息,为避免恢复表头对 ED_Model 算法检测实体列的精度和召回率的影响,ED_Model 算法和 PG_Model 算法利用表格已有的表头实现;实验中,AFD_Model 算法会返回 1 个 ~2 个候选实体列,因此,放宽了 ED_Model 算法只返回可能性得分最高的列作为实体列的限制,允许返回 top-2 个可能性最高的实体列,也允许 PG_Model 算法只返回 top-2 个最强实体列。

为了评估提出的近似函数依赖的评分方式的有效性,将算法中计算 Web 表格近似函数依赖算法替换为文献[10]给出的概率函数依赖算法,其余的删减噪声函数依赖等算法均不变,称其为 PFD_Model 算法。

本节从精确度、召回率、F 值、运行时间 4 个方面对 ED_Model 算法、PG_Model 算法、PFD_Model 算法与提出的基于近似函数依赖的实体列发现算法 AFD_Model 进行比较。因为 ED_Model 算法只能对单实体列的表格进行实体列发现,所以本节仅选取单实体列数据集进行实验对比。首先按照规则 4 为单实体列实验数据集进行人工标注实体列。

规则 4　选取 3 位志愿者参与单实体列数据集实体列的标注,每一名志愿者为表格标注一个实体列,只有当 2 个以上的志愿者都认为某一列是实体列时,才将该列标注为实体列;否则,认为该表没有实体列。

规则 5　对于某一张 Web 表格 T,如果发现有实

体列输出,判断算法发现的实体列组成与志愿者标注的实体集是否存在交集,如果存在交集,则对该表的实体列发现结果打分为 $Score(T)=1$,如果不存在交集,则对该表的实体列发现结果打分为 $Score(T)=0$。

统计实验结果,N 为单实体列 Web 表格总数,$\overline{N}$ 为算法没有发现实体列的表格数,实验结果的精确度、召回率、F 值由式(9)~式(11)计算。

$$precision=\frac{\sum_{i\in(1,N)}Score(T_i)}{N-\overline{N}} \tag{9}$$

$$recall=\frac{\sum_{i\in(1,N)}Score(T_i)}{N} \tag{10}$$

$$F\text{-}measure=\frac{2\times precision\times recall}{precision+recall} \tag{11}$$

近似函数依赖的评分计算时,设置了参数 ω_1 和 ω_2,图 1 是在 Wiki Table 数据集上,θ 取 0.8,ω_1 取不同值的实验结果。由实验可知,当 ω_1 取 0.9 时,算法的准确率最高。因此,下文实验中 ω_1 均取 0.9。

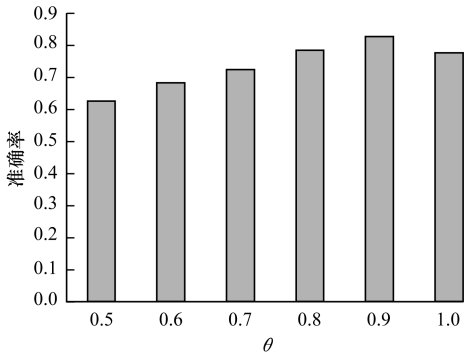

图 1 阈值 θ 与算法准确率的关系

通过实验分析对比 AFD_Model 算法、PG_Model 算法和 ED_Model 算法的实验结果,来验证本文提出的基于 Web 表格的近似函数依赖发现实体列算法的有效性。通过对比 AFD_Model 算法分别在 Wiki_SS 数据集、Wiki_MS 数据集和 Wiki_LS 数据集上与 PFD_Model 算法的实验结果,来验证本文提出的 Web 表格近似函数依赖评分方式的有效性。实验结果如图 2~图 5 所示。

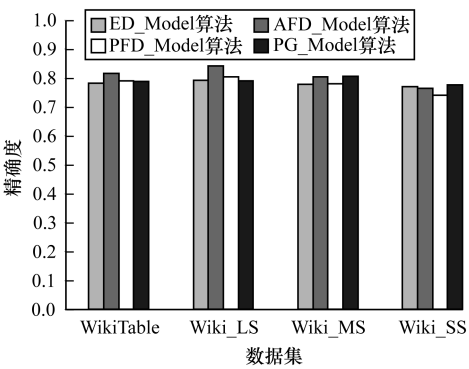

图 2 不同数据集上各算法精确度的对比

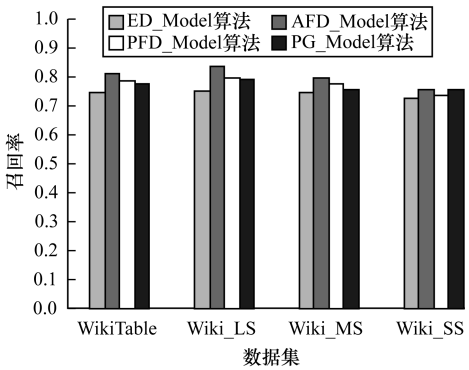

图 3 不同数据集上各算法召回率的对比

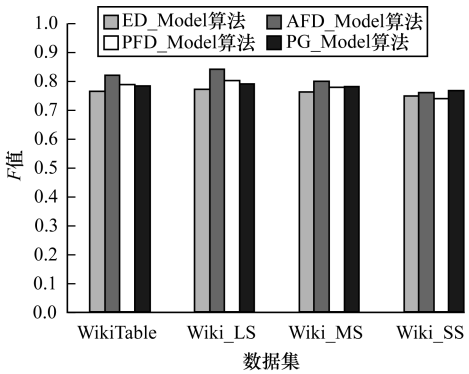

图 4 不同数据集上各算法 F 值的对比

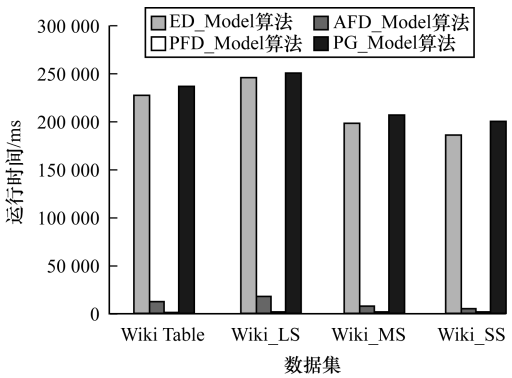

图 5 不同数据集上各算法运行时间的对比

单实体列发现的实验结果表明:

1)在精确度方面,AFD_Model 算法优于其他 3 种算法。ED_Model 算法和 PG_Model 算法都要求 Web 表格的表头在 Probase 库中存在概念属性关系,Web 表格表头的质量和知识库的覆盖程度都会影响 ED_Model 算法和 PG_Model 算法的准确度,而 AFD_Model 算法不依赖任何表头信息和知识库。由于 AFD_Model 算法考虑到了 Web 表格的特点,具有一定的噪声过滤能力,因此实体检测的精确度也高于 PFD_Model 算法。

2)在召回率方面,AFD_Model 算法也高于 ED_Model 算法、PG_Model 算法和 PFD_Model 算法。因为 AFD_Model 算法不要求 Web 表格必须存在表头,不要求表中的实体列与非实体列存在属性关系,

也不要求这种概念-属性关系在 Probase 库中存在,同时具有一定的噪声过滤能力,所以算法的适应性更强。

3)由于精确度和召回率上的良好表现,AFD_Model 算法的 F 值也高于其他 3 种算法。

4)在运行时间方面,ED_Model 算法和 PG_Model 算法的时间花费明显大于 AFD_Model 算法和 PFD_Model 算法,因为 ED_Model 算法和 PG_Model 算法都需要利用 Probase 库将表格的表头或者语义恢复出来的表头概念属性关系确定下来,进而确定实体列,而 AFD_Model 算法和 PFD_Model 算法的时间复杂度仅与表格的大小有关。AFD_Model 算法的时间花费高于 PFD 算法,因为 AFD 算法增加了不一致性数据对函数依赖成立的支持度计算,提高了实体列检测的精确度和召回率。

5)AFD_Model 算法在 L 数据集上无论精确度、召回率还是 F 值均优于在 S 数据集上的实验结果。因为数据量小的 Web 表格受噪声数据的影响较大,而数据量大的 Web 表格受噪声数据的影响较小。

3.3　多实体列 Web 表格的实体列发现

因为 ED_Model 算法只针对单实体列表格,所以多实体列发现的实验结果只能和 PG_Model 算法进行对比,而 PG_Model 算法依赖表头信息。因此,对于数据集中没有表头的表格,人工添加合理的表头作为 PG_Model 算法的实验数据。下面是多实体列发现的评价方法。

假设一个表格 T 包含 m 个实体列,实体列集合记为 $W=\{e_i \mid i=1,2,\cdots,m\}$,通过算法输出 t 个实体列,记实体列集合 $F=\{e_i \mid i=1,2,\cdots,t\}$,那么针对该表格,准确率、召回率的计算如式(12)、式(13)所示。

$$presion(T)=|W\cap F|/t \tag{12}$$

$$recall(T)=|W\cap F|/m \tag{13}$$

假设 N 为实验数据集中 Web 表格总数,实验结果的精确度、召回率计算如式(14)、式(15)所示。

$$precision=\frac{\sum_{i\in(1,N)} precision(T_i)}{N} \tag{14}$$

$$recall=\frac{\sum_{i\in(1,N)} recall(T_i)}{N} \tag{15}$$

分别统计 AFD_Model 算法、PFD_Model 算法和 PG_Model 算法在 WiKi_LM 数据集、Web_LM 数据集、WiKi_MM 数据集、Web_MM 数据集、WiKi_SM 数据集和 Web_SM 数据集上的精确率、召回率和 F 值,如图 6 ~ 图 8 所示。

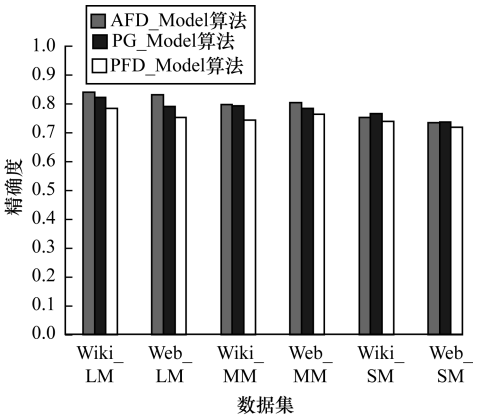

图 6　不同数据集上各算法精确度的对比

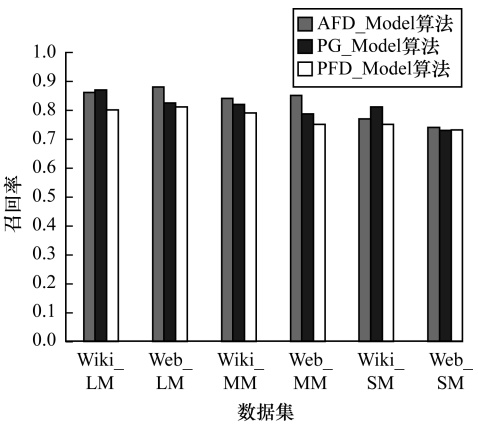

图 7　不同数据集上各算法召回率的对比

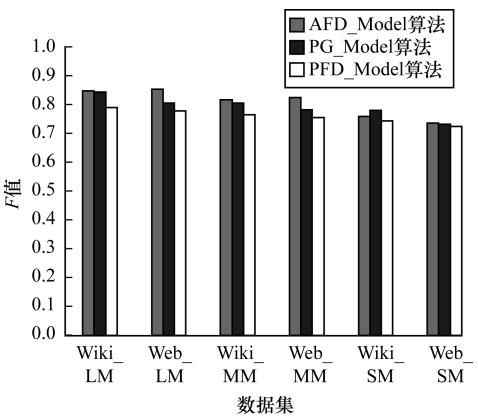

图 8　不同数据集上各算法 F 值的对比

多实体列发现的实验结果表明:

1)无论精度、召回率、还是 F 值,AFD_Model 算法都比 PFD_Model 算法表现优秀,这是因为 AFD_Model 算法在计算属性间的近似函数依赖时,考虑了噪声数据的影响;在大部分数据集上 AFD_Model 算法比 PG_Model 算法表现优秀,因为 Web 表格表头的质量和知识库的覆盖程度都会影响 PG_Model 算法的精确度和召回率;PG_Model 算法在 Wiki Table 数据集上均优于 Web Table 数据集的实验结果,这是因为

Wiki Table 数据集表头的质量比 Web Table 数据集的质量高。

2)AFD_Model 算法和 PFD_Mode 算法在大表上的表现均好过小表,表明本文提出的近似函数依赖和文献[10]提出的概率函数依赖均可能对大数据量的表格更适用。

3)相比传统的实体列检测方法,APD_Model 算法能实现无表头 Web 表格的多实体列检测,且有不错的表现。

4 结束语

本文提出适用于 Web 表格的近似函数依赖及其评分方法,并将其应用于 Web 表格的实体列发现。与现有的实体列检测方法相比,提出的基于近似函数依赖和规范化的实体列发现算法,可以在更多场景下发现实体列。该方法不仅适用于单实体列的 Web 表格,还可用于多实体列的表格;不仅适用于有表头的 Web 表格,而且适用于没有表头或者利用语义恢复技术也无法恢复出完整表头的 Web 表格。下阶段将研究如何进一步提高函数依赖检测的准确度。

参考文献

[1] CAFARELLA M J, HALEVY A, WANG Z D, et al. WebTables: Exploring the Power of Tables on the Web[C]//Proceedings of the 34th International Conference on Very Large Data Bases. New York, USA: ACM Press, 2008: 538-549.

[2] VENETIS P, HALEVY A, MADHAVAN J, et al. Recovering Semantics of Tables on the Web[C]//Proceedings of the 37th International Conference on Very Large Data Bases. New York, USA: ACM Press, 2011: 1601-1612.

[3] BALAKRISHNAN S, HALEVY A, HARB B, et al. Applying WebTable in Practice[EB/OL]. (2015-12-06). http://webdatacommons.org/webtables/.

[4] LIMAYE G, SUNITA S, CHAKRABARTI S. Annotating and Searching Web Tables Using Entities Types and Relationships[J]. Proceedings of the VLDB Endowment, 2010, 3(3): 1338-1347.

[5] DENG Dong, JIANG Yu, LI Guoliang, et al. Scalable Column Concept Determination for Web Tables Using Large Knowledge Bases[J]. Proceedings of the VLDB Endowment, 2013, 6(13): 1606-1617.

[6] WANG Jingjing, WANG Haixun, WANG Zhongyuan, et al. Understanding Tables on the Web[C]//Proceedings of the 31st International Conference on Conceptual Modeling. New York, USA: ACM Press, 2012: 141-155.

[7] LEE T, WANG Zhongyuan, WANG Haixun, et al. Attribute Extraction and Scoring: A Probabilistic Approach[C]//Proceedings of the 29th International Conference on Data Engineering. Washington D. C., USA: IEEE Press, 2013: 194-205.

[8] 任向冉. 网络表格的实体列发现与标识[D]. 北京: 北京交通大学, 2015.

[9] 黎章海, 潘久辉. 基于函数依赖的导出关系候选码计算[J]. 计算机工程, 2016, 42(5): 60-65.

[10] WANG D Z, DONG Luna, SARMA A D, et al. Functional Dependency Generation and Applications in Pay-as-You-Go Data Integration Systems[C]//Proceedings of the 12th International Workshop on the Web and Databases. Berlin, Germany: Springer, 2009: 1654-1655.

[11] WANG J N, LI G L, FENG J H. Fast-Join: An Efficient Method for Fuzzy Token Matching Based String Similarity Join[C]//Proceedings of the 27th International Conference on Data Engineering. Washington D. C., USA: IEEE Press, 2011: 458-469.

[12] 萨师煊, 王 珊. 数据库系统概论[M]. 北京: 高等教育出版社, 2002.

[13] LAUTERT L R, SCHEIDT M M, DORNELES C F. Web Table Taxonomy and Formalization[J]. ACM SIGMOD Record, 2013, 42(3): 28-33.

[14] VERGA P, NEELAKANTAN A, MCCALLUM A. Generalizing to Unseen Entities and Entity Pairs with Row-less Universal Schema[EB/OL]. (2016-06-18). https://arxiv.org/abs/1606.05804.

[15] XIA Zhihua, WANG Xinhui, SUN Xingming, et al. A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data[J]. IEEE Transactions on Parallel & Distributed Systems, 2016, 27(2): 340-352.

编辑 顾逸斐

(上接第 164 页)

[12] TURNEY P D. Thumbs Up or Thumbs Down?: Semantic Orientation Applied to Unsupervised Classification of Reviews[C]//Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. New York, USA: ACM Press, 2002: 417-424.

[13] CHEN Yuefeng, MIAO Duoqian, LI Wen, et al. Semantic Orientation Computing Based on Concepts[J]. CAAI Transactions on Intelligent Systems, 2012, 6(6): 489-494.

[14] BRANDES U. On Variants of Shortest-path Betweenness Centrality and Their Generic Computation[J]. Social Networks, 2008, 30(2): 136-145.

[15] WANG Suge, LI Deyu, WEI Yingjie, et al. A Synonyms Based Word Sentiment Orientation Discriminating[J]. Journal of Chinese Information Processing, 2009, 23(5): 68-74.

编辑 顾逸斐