

社交网络中同一用户的识别

张 征 王宏志 丁小欧 李建中 高 宏

(哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)

摘 要 对不同社交全局网络中同一用户的身份识别进行了相关研究,将社交网络建模为节点带有属性值且含有一个中心节点的网络,即 ego-network,并就社交网络中身份识别的问题设计了相关算法。为挖掘同一个用户的节点对,对用户的属性、好友关系的相似度进行了建模,从而综合评价了不同社交网络中节点间的相似度,即为用户匹配评分,将其作为节点匹配的优先度;然后通过改进后的 RCM 算法得到全局最优的匹配结果;最后剪掉用户匹配评分较低的已匹配用户对以达到更好的效果。基于真实数据集,实验对比了该算法与几种相关算法的表现,并分析了不同参数对实验效果的影响,验证了所提算法的合理性。

关键词 社交网络,用户识别,用户属性,RCM 算法

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.09.012

Identification of Same User in Social Networks

ZHANG Zheng WANG Hong-zhi DING Xiao-ou LI Jian-zhong GAO Hong

(Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

Abstract This paper carried on the related research of the same user identification in different social global networks. The social network was modeled as a network with attribute value and a central node, namely ego-network. And aiming at the identification problem in the social network, this paper designed related algorithm. In order to mine the node pairs of the same user, the user's attributes and the similarity of the friends' relationship are modeled, so as to comprehensively evaluate the similarities among the nodes in different social networks, namely, to get the user match score and to use it in node matching. Then through the improved RCM algorithm, the global optimal matching results are obtained, and finally the matching user pairs with lower user match scores are cut off to achieve better results. Based on real datasets, the performance of the algorithm is compared with several related algorithms. The effect of different parameters on experimental results is also analyzed and the rationality of the proposed algorithm is verified.

Keywords Social networks, User identification, User attributes, RCM algorithm

1 引言

近几年,为了满足人们的各种需求,各种各样的社交平台层出不穷,如可随时随地分享身边新鲜事的微博、分享影评和书评的豆瓣、侧重于社交交友的微信等。社交网络的快速发展,使得人与人之间形成了复杂多维的社交网络集合。用户在社交网络上的各种行为,如购物、交友、搜索信息等,依托于各种软件平台,存在于各种社交网络之中。随着社交网络的发展,人们在社交网络中的行为越来越多样、全面,越来越接近其在真实社交中的行为。

但是,由于各种社交网络之间信息的不流通、封闭性,我们面临的一大问题就是如何整合各种数据,构建出完整、真实的社交网络,从而使每个社交网络的工作者能够得到关于用户

的其他社交网络的信息,为用户提供更好的喜好推荐、好友推荐等服务。本文目的是解决社交网络中同一用户的识别问题。

现实世界中,社交网络是巨大的,每个用户都有其用户属性(如用户昵称、所在地、生日、性别等)以及与其他用户建立的好友关系。本文综合利用了这些信息,将用户抽象为社交网络无向图中的节点,将用户之间的好友关系抽象为该图中的边。多社交网络中同一用户的识别可抽象为如下问题。

如图 1 所示,社交网络表示为一个无向图,记为 $G = (V, E)$,给定两个无向图,分别为 G_0 和 G_1 , V 中的每个节点为一个社交网络中的用户,每个节点 v_i 拥有属性域。节点 G_0 和 G_1 间的属性相似度记为 $P, P = \{p_{ij} \mid v_i \in G_0, v_j \in G_1\}$ 。 E 中的边表示用户之间互为好友关系,且已经知道 G_0 和 G_1 的中心为同一个用户的账户(同一用户的账户用虚线表示)。因

到稿日期:2018-07-10 返修日期:2018-09-30 本文受国家自然科学基金重点项目(U1509216),国家重点研发计划项目(2016YFB1000703),国家自然科学基金面上项目(61472099,61602129)资助。

张 征(1997—),男,主要研究方向为社交数据挖掘;王宏志(1978—),男,博士,教授,CCF 会员,主要研究领域为数据库、大数据,E-mail:wang-zh@hit.edu.cn;丁小欧(1993—),女,博士生,CCF 学生会会员,主要研究方向为数据质量管理、数据清洗等;李建中(1950—),男,博士,教授,CCF 会员,主要研究方向为数据库系统实现技术、数据库库等;高 宏(1966—),女,博士,教授,CCF 会员,主要研究方向为复杂结构数据管理、无线传感器网络等。

此,用户匹配问题就成为了在给定如上信息的基础上,尽可能多且正确地识别出两个 ego-network 节点(ego-network 节点是由唯一的一个中心节点 ego,以及该节点的邻居 alter 组成的,它的边只包括 ego 和 alter 之间,以及 alter 与 alter 之间的边)中的相同用户并将其放入集合 SEED 中的问题。

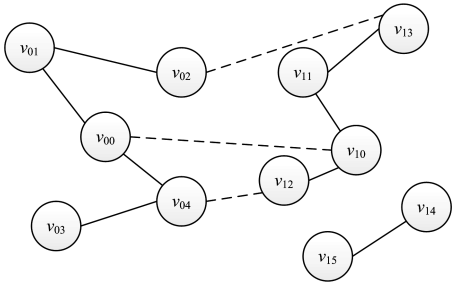


图 1 社交网络抽象图
Fig. 1 Social network abstract illustration

综上,本文提出一种算法,该算法综合利用了用户属性、好友结构关系等信息,优化了匹配过程,尽可能多且正确地得到两个社交网络中为同一个用户的节点对的集合。

本文的主要贡献如下:

- (1)综合利用了属性相似度和好友结构相似度来评价两个社交网络中的节点相似度,结果可信;
- (2)提出了跨多社交网的稳定匹配算法,以挖掘不同社交网络中的同一账户节点;
- (3)针对实际实验中出现的各种问题,如错误匹配对后续匹配节点造成的不良影响、向外迭代次数过多匹配使得得分过低等,提出了一些解决的方法,经真实数据集测试,验证了所提方法的有效性。

2 相关工作

现阶段,大部分的身份识别相关算法主要用于属性字段的相似度计算^[1-2],如 Vosecky 等^[3]提出了一种用于度量名字相似度的 VMN 算法,将每个用户的属性信息表示为一个向量,并计算每个向量维度之间的相似度,得到相似度向量,每个属性域针对其特点采用不同的匹配策略。

但这种算法对数据集的要求过高,且效果不好。下面介绍几种综合利用了属性相似度和好友关系相似度的算法^[4-8]。

Bartunov 等^[4]提出了一种有效结合账户属性信息和链接信息来解决 ego-network 匹配的局部身份识别的 JLA 算法。在 JLA 算法中,图被抽象为 $A=(V,E)$,在该模型上建立点对条件随机场。在该点对条件随机场中,可观测变量 $X=\{x_v=v|v\in V\}$ 。隐藏变量 $Y=\{y_v=\mu(v)|v\in V\}$,其中 $\mu(v)$ 表示 v 匹配的节点,而这些节点取自另一个图的节点集合 B 。任意一个隐藏变量 y_v 和可观测变量 x_v 通过因子 Φ 链接在一起。而边则被建模成因子 Ψ 链接。在这种设定下,一个可观测变量与一个隐藏变量匹配的后验概率公式为 $P(Y|X)\propto Exp(-E(Y|X))$ 。能量泛函 E 由一元泛函 Φ 、二元泛函 Ψ 组成。一元泛函 Φ 对应账户属性相似度,二元泛函 Ψ 对应用户之间的距离。在给定观测变量 X 的情况下,隐藏变量 Y 的最优选择就是最小化能量泛函。

另一个重要的算法——RCM 算法^[5,7]是在参考 JLA 算法后提出的,其创造性地提出了如下几个概念:用户环境评分

是对用户是否在另一个社交网络中含有账号的可能性的评估;用户匹配评分,即结合用户关系相似度和用户属性相似度计算出一个得分,用于从另一个网络中选择出符合程度最高的节点。

本文还利用了一些处理图结构数据的相关算法^[9-10],一个重要的算法是基于结构的聚类算法 SCAN (Structural Clustering Algorithm for Network)^[9],该算法在处理分析现实数据形成的图时取得了很好的效果,可用于得到基于图结构的聚类结构。

3 关键定义

定义 1(节点相似度) 对于同一个节点网中的两个点 a 和 b ,相似度的一个重要衡量指标为节点相似度 N ,即节点间的链接关系相似度。在参考了 SimRank^[11]这一基于图结构的相似度方法之后,本文提出用 a 的 neighbors 和 b 的 neighbors 相似度的平均值来衡量 a 和 b 的相似度,公式如下:

$$N(a,b)=\frac{c}{|I(a)||I(b)|}\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|}N(I_i(a),I_j(b)) \quad (1)$$

其中, c 为阻尼系数。当 $a=b$ 时, $N(a,b)=c$ 。 $I_i(a)$ 表示 a 的第 i 个 neighbor。当 $I(a)$ 或 $I(b)=\emptyset$ 时, $N(a,b)$ 为 0。

节点相似度函数是 Simrank 方法的变种,原 Simrank 计算的是有向图中的节点相似度,因此须考虑 in-neighbor 和 out-neighbor 两种情况。本文中涉及的目标社交网中好友关系是双向的,因此可将好友关系图视为无向图,可将 simrank 方法简化为式(1)中的结果。

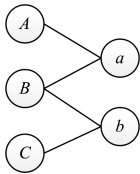


图 2 用户关系示意图
Fig. 2 User relationship diagram

定义 2(用户关系相似度) 对于不同社交网络的两个点 a 和 b ,它们之间的相似度的衡量指标为用户关系相似度 URS(User Relationship Similarity)。计算方法为计算节点与已经识别出的所有节点对(即集合 SEED 中的节点对)之间的用户关系相似度之和,并将其作为两个节点间的用户关系相似度,具体如式(2)所示:

$$URS(a,b)=\sum_{s\in SEED}\frac{N(a,s)+N(b,s)}{2} \quad (2)$$

定义 3(用户匹配评分) 用户匹配评分 UMS (User Match Score)用来评定得到的两个社交网络中的节点是否为同一节点:

$$UMS(a,b)=\frac{URS(a,b)+\lambda*|SEED|*UAS(a,b)}{(1+\lambda)*|SEED|} \quad (3)$$

本文综合了两个社交网络中节点的属性相似度 UAS (User Attribute Similarity)和用户好友关系相似度来综合评价两个社交网络中节点的相似度。其中 a 和 b 分别属于两个不同的社交网络;已经识别出来的种子节点对的个数 $|Seed|$ 用于平衡 URS 和 UAS 的大小; λ 为平衡参数,用来调整 UAS 和 URS 占 UMS 的比重。UAS 则使用 Vosecky 的方法进行计算。

需要注意的一点是,在计算 URS 值时,会利用之前识别成功的节点对,即 $SEED$ 中匹配成功的节点用户对。但为了避免之前的错误匹配对后续节点匹配造成较大的影响,在计算 URS 值时,可使用原认为匹配的节点的用户匹配评分作为权重来减少错误匹配对后续节点匹配的影响,即减小匹配错误的节点对对 URS 值的贡献。

4 算法模型

4.1 初步用户选择、匹配

4.1.1 初始用户选择

好友数量较多的用户在社交网络中的活跃程度较高,其属性相似度、结构相似度等指标较为可信,在另一个社交网络中拥有另一个账号的可能性较高,因此应优先选择这种账户,记为 v_{select} ,为其选择“心仪”的匹配用户。按照这种顺序,可在算法初期以更高的概率得到可靠的匹配。

需要注意的是,如果节点匹配失败,则将其放入匹配失败队列,作为优先度最高的用户,在下次选择开始时优先选择。

4.1.2 可靠用户对的匹配

在选择了合适的账户之后,需要从另一个社交网络中选出最符合的用户。评价是否符合的标准是计算出对应网络中各节点和 v_{select} 的用户匹配评分(UMS)值,选择对应的 UMS 值最大的节点作为 v_{match} 返回。具体的匹配算法如算法 1 所示。

算法 1 匹配算法

```
Input:  $v_{select}; V_0; V_1$ 
Output: matched point
1. if  $v_{select}$  in  $V_0$  then
2.   for each  $v \in V_1$  do
3.     get  $UMS(v, v_{select})$ ;
4.   end for
5.    $v_{match} \leftarrow \max UMS(v, v_{select})$ ;
6.   return  $v_{match}$ ;
7. else
8.   for each  $v \in V_0$  do
9.     get  $UMS(v, v_{select})$ ;
10.  end for
11.  $v_{match} \leftarrow \max UMS(v, v_{select})$ ;
12. return  $v_{match}$ ;
13. end if
```

4.2 稳定匹配

在选择有效的用户和进行可靠的用户匹配后,对两个 ego-network 节纪录片进行整体匹配来得到稳定的匹配结果。算法 2 即为稳定匹配算法。

算法 2 稳定匹配算法

```
Input:  $V_0, V_1$ 
Output:  $SEED$ 
1.  $SEED \leftarrow [(0, 0, 1)]$ ;
2.  $v_{select} \leftarrow NULL$ ;
3.  $v_{match} \leftarrow NULL$ ;
4.  $ArrayUnmatched \leftarrow NULL$ ;
5. while  $V_0 \neq \emptyset$  and  $V_1 \neq \emptyset$  do
6.   if  $v_{select} == NULL$  then
```

```
7.      $v_{select} \leftarrow SELECT(V_0, V_1)$ ;
8.      $v_{match} \leftarrow MATCH(v_{select}, V_0, V_1)$ ;
9.   end if
10.   $v'_{match} \leftarrow MATCH(v_{select}, V_0, V_1)$ ;
11.  if  $v_{select} == v'_{match}$  then
12.    if  $v_{select}$  in  $V_0$  then
13.      if  $UMS(v_{select}, v_{match}) > \alpha$  then
14.        Raise the  $UMS$  of  $V_{pair}$  in the same cluster by  $\rho$ ;
15.      end if
16.       $V_{pair} \leftarrow [v_{select}, v_{match}, UMS(v_{select}, v_{match})]$ ;
17.       $SEED \leftarrow SEED \cup V_{pair}$ ;
18.       $V_0 \leftarrow V_0 - v_{select}$ ;
19.       $V_1 \leftarrow V_1 - v_{match}$ ;
20.    else
21.      Do the same procedure to  $(v_{match}, v_{select})$ ;
22.    end if
23.     $v_{select} \leftarrow NULL$ ;
24.  else
25.     $ArrayUnmatch \leftarrow ArrayUnmatch \cup v_{select}$ ;
26.     $v_{select} \leftarrow v_{match}$ ;
27.  end if
28. end while
```

算法分析:算法 2 借鉴了 RCM 中匹配算法的思想,改进了其不合理的地方,并对其进行改进以应用于本文的实际情况。具体匹配过程如下:

(1)若匹配失败队列不为空,则选择队列中第一个元素作为 v_{select} ,否则计算出待匹配队列的所有元素的好友数量,选择好友数量最多的元素作为 v_{select} 。

(2)通过匹配算法得到 v_{select} 对应的 v_{match} ,再对 v_{match} 求出其匹配的对应该元素 v'_{match} 。

(3)若 v'_{match} 等于 v_{select} ,则说明得到了稳定的匹配,将 $(v_{select}, v_{match}, UMS)$ 放入 $SEED$ 中,作为匹配成功的节点对;否则,认为匹配失败,将 v_{select} 放入匹配失败队列,然后令 v_{match} 为新的 v_{select} ,执行步骤(2)。

(4)若待匹配队列为空,则算法结束,否则跳转至步骤(1)。

需要注意的是,由于社交网络节点众多,不做处理的话,算法在面对大规模社交网络时存在可扩展性的问题,即当向外迭代次数增多时,在后续节点对的 URS 计算过程中所有之前的匹配结果的权重会越来越低,同时得到的节点匹配评分也会逐渐降低,因此须调整权重,即每得到一个匹配结果,就对匹配结果进行可信度分析。如果可信度很高,超过阈值 α ,则反推对这个匹配贡献度高的匹配对,在一定程度上增加了其用户匹配分数,从而提高其匹配结果的可信度并且对之后的匹配过程的影响增大。这可作为一种正反馈及解决向外迭代次数过多、权重越来越低、匹配结果分数过低的情况的方法。这种方法在原 RCM 算法中是没有被考虑到的。

本文采用基于结构的聚类算法——scan^[9]算法来解决上述问题。使用 scan 算法得到节点图的聚类,即得到关系较为密切的节点聚类,当某个匹配结果的可信度超过阈值时,对该节点在同一个聚类中且已被识别出来的节点对的 UMS 值进行放大,如乘以 $(1 + \rho)$ 。

4.3 剪枝过程

在稳定匹配过程中,终止条件是某一个社交网络中的所有节点都得到了相应的匹配,由于数据集中并不是所有节点都在另一个社交网络中有对应的账号,因此最终结果中一定有部分节点的匹配结果是错误的。

算法 2 已经计算了每个匹配的节点对所对应的用户匹配评分,因此可直接使用该指标去除已经匹配的但用户相似度评分过低的结果,并截取 UMS 值较高的结果作为最终的匹配结果。通过实验分析可得,匹配结果为真阳性的用户匹配评分较大,验证了该剪枝方法的可行性。

4.4 算法复杂度分析

记两个社交网络的节点数分别为 $|V_0|$ 和 $|V_1|$ 。身份识别的过程中,主要计算用户匹配评分的计算代价。算法开始时,在第一次计算中由于种子用户的数量为 1 对,因此计算次数为 $|V_0|-1$ 或 $|V_1|-1$;在第二次计算中由于已经识别出 1 对种子用户,因此 UMS 的计算次数为 $|V_0|-2$ 或 $|V_1|-2$;以此类推,最终计算次数为 $(|V_0|-1)(|V_1|-1)$ 。故交叉匹配过程算法的时间复杂度为 $O(n^2)$ 。

5 实验分析

5.1 数据集

本文使用 Korshunov 等^[4]提供的已标注的数据集,该数据集包括 16 个 Facebook-twitter 的 ego-network,一共有 977+398 个节点,其中正确匹配的节点对数量为 141。用户信息包括姓名(用数字表示)、好友关系、用户与用户之间是否为同一个账号以及用户之间的 3 个属性域的相似度。

5.2 实验环境

实验平台:本文所有程序均用 python 语言实现,实验环境为 Intel(R) core i5-8400 @2.80 GHz 六核,系统内存为 8GB,操作系统为 Windows 10。

5.3 评价方案

使用传统的准确率和召回率度量实验结果。准确率公式和召回率公式分别如式(4)和式(5)所示:

$$precision = \frac{tp}{tp+fp}$$
(4)

$$recall = \frac{tp}{tp+fn}$$
(5)

本文还使用式(6)的 F-score 来综合评价算法结果:

$$f_{\beta} = (1+\beta^2) * \frac{precision * recall}{\beta^2 * precision + recall}$$
(6)

当 $\beta=1$ 时,F-score 记为 F_1 -score。

5.4 用于比较的算法

本实验将本文算法与相关工作中的 3 种算法进行对比。对比算法分别是:JLA 算法、RCM 算法、RCM 算法的变种 RCM_{wl} (即去除了好友之间的链接信息的 RCM 算法)。

5.5 实验结果与分析

实验 1 在不同剪枝率的情况下,在数据集上比较准确度、召回率以及 F_1 -score 的变化情况,结果如图 3 所示。在最佳剪枝率的情况下将本文算法与其他几种相关算法的正确率、召回率、 F_1 -score 进行比较,结果如表 1 所列。

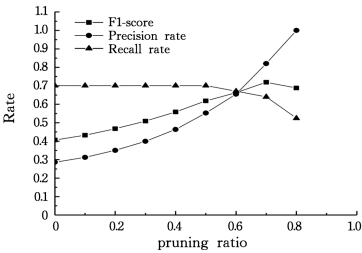


图 3 剪枝率对算法的影响

Fig. 3 Impact of pruning ratio on algorithm

表 1 实验对比结果

Table 1 Experimental comparison results

algorithm	recall	precision	F_1 -score	pruning/%
JLA(T->F)	0.38	1	0.550	—
JLA(F->T)	0.58	1	0.730	—
RCM	0.73	0.85	0.7870	10
RCMwl	0.68	0.81	0.740	65
本文算法	0.64	0.82	0.720	70

从实验 1 可以看出,正确匹配的节点对匹配评分普遍偏高,因此随着剪枝率的提高,匹配评分较低的匹配节点对被去除,这对正确匹配的节点影响很小,故准确率持续升高。但对于召回率,在剪枝率较高的情况下,召回率下降,即小部分正确匹配的节点亦被剪除,并且本文使用的数据集较为稀疏,即在另一个社交网络中拥有对应账号的账号相对较少。

通过对比分析实验结果可得,无论是 Facebook→Twitter,还是 Twitter→Facebook,JLA 算法的准确率都非常高,但这是建立在召回率很低的基础之上的,从 F_1 -score 指标来看本文算法要优于 JLA 算法。

而 RCM^[5]算法在对大部分社交网络进行实验时,没有处理错误匹配对后续节点的影响,且依据得到的匹配节点对的先后次序进行剪枝,因此最初的匹配错误会对后续的匹配造成很大的影响,在对比较大的社交网络匹配时,效果不是很好。

实验 2 通过改变某一参数并固定其他的参数,在数据集上分别进行实验,观察部分参数对实验结果的影响,实验结果如图 4 所示。

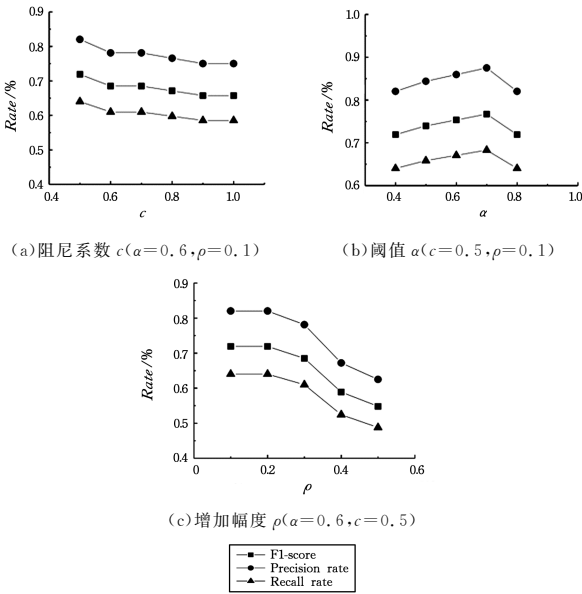


图 4 部分参数对实验结果的影响

Fig. 4 Effects of some parameters on experimental results

从实验 2 可以看出,参数的改变对实验结果有一定的影响(具体的效果依赖于数据集的环境)。

图 4(a)中,阻尼系数 c 影响了 URS 的大小, c 值越高表示 URS 占 UMS 的比例越高。在好友属性较为可靠的情况下,推荐降低 c ,以增加好友属性相似度在实验中的比例。

图 4(b)中,阈值 α 越高,迭代后期的正确匹配对的 UMS 因超过阈值被增加的次数就越少,越可能被剪枝掉,从而使准确率下降。

图 4 (c)中,增加幅度 ρ 越小,正确的匹配在对后续匹配有正面效果时受到的增益小,因此可适当增加 ρ ,不过若 ρ 过大,则会对部分错误匹配有较大的增益,故须进行多次实验,选择适合的 ρ 值。

6 实例分析

为了验证本文算法的有效性 with 正确性,即是否能够从两个社交网络中发掘出同为一个人的账号,本文选取了一个小型的 ego-network 来演示算法过程。图 5 展示了所选择的 ego-network 结构,其中两个社交网中 v_{00} 和 v_{10} 已知为同一用户,节点间的边代表两个用户互为好友关系,节点间的属性相似度如表 2 所列。经 simrank 计算得出的好友关系相似度如表 3 和表 4 所列。

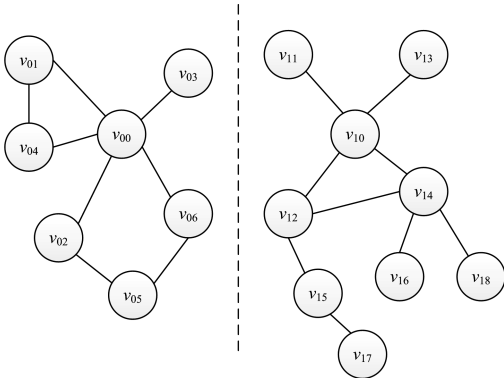


图 5 Ego-network 实例
Fig. 5 Ego-network instance

表 2 各节点的用户属性相似度

Table 2 Dttribute similarity of user nodes

node	v_{00}	v_{01}	v_{02}	v_{03}	v_{04}	v_{05}	v_{06}
v_{10}	1.0	0.8	0.5	0.9	0.6	0.5	0.8
v_{11}	0.8	1.0	0.6	0.7	0.8	0.6	0.3
v_{12}	0.5	0.7	0.8	0.5	0.6	0.7	0.8
v_{13}	0.8	0.6	0.7	0.8	0.7	0.5	0.6
v_{14}	0.8	0.7	0.8	0.5	0.6	0.4	1.0
v_{15}	0.5	0.6	0.7	0.2	0.5	0.8	0.2
v_{16}	0.5	0.8	0.2	0.4	0.6	0.5	0.1
v_{17}	0.5	0.4	0.6	0.8	0.1	0.5	0.1
v_{18}	0.2	0.3	0.4	0.2	0.2	0.5	0.4

表 3 V_0 中用户关系相似度

Table 3 User relationship similarity of V_0

node	v_{00}	v_{01}	v_{02}	v_{03}	v_{04}	v_{05}	v_{06}
v_{00}	1.00	0.71	0.71	0.71	0.63	0.55	0.57
v_{01}	0.71	1.00	0.71	0.71	0.63	0.55	0.57
v_{02}	0.71	0.71	1.00	0.71	0.63	0.55	0.57
v_{03}	0.71	0.71	0.71	1.00	0.63	0.55	0.57
v_{04}	0.63	0.63	0.63	0.63	1.00	0.55	0.53
v_{05}	0.55	0.55	0.55	0.55	0.55	1.00	0.49
v_{06}	0.57	0.57	0.57	0.57	0.53	0.49	1.00

表 4 V_1 中用户关系相似度

Table 4 User relationship similarity of V_1

node	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}
v_{10}	1.00	0.71	0.71	0.71	0.63	0.55	0.49	0.42	0.49
v_{11}	0.71	1.00	0.74	0.71	0.63	0.55	0.49	0.42	0.49
v_{12}	0.71	0.74	1.00	0.74	0.63	0.55	0.49	0.42	0.49
v_{13}	0.71	0.71	0.74	1.00	0.63	0.55	0.49	0.42	0.49
v_{14}	0.63	0.63	0.63	0.63	1.00	0.63	0.49	0.42	0.49
v_{15}	0.55	0.55	0.55	0.55	0.63	1.00	0.49	0.42	0.49
v_{16}	0.49	0.49	0.49	0.49	0.49	0.49	1.00	0.49	0.65
v_{17}	0.42	0.42	0.42	0.42	0.42	0.42	0.49	1.00	0.49
v_{18}	0.49	0.49	0.49	0.49	0.49	0.49	0.65	0.49	1.00

使用 scan 算法得出 V_0 的节点聚类为 $(2,5),(0,1,3,4,6)$ 。除 v_{00} 和 v_{10} 以外,实际 $(v_{06},v_{14}),(v_{01},v_{11})$ 亦为同一用户的账户,不作为已知条件输入。阈值 α 设为 0.8,增加幅度 ρ 设为 10%,平衡参数 λ 设为 1。算法运行过程如下。

(1)第一次匹配

1)初始时,匹配失败队列为空,选择待匹配队列好友数量最多的节点 v_{14} 作为 v_{select} 。

2)通过匹配算法选择 V_0 中与 v_{select} 一起作为节点对的计算得出 UMS 值最大的节点 v_{06} 作为 v_{match} , UMS 值为 0.8,再求出与 v_{match} 相匹配的对应元素 v'_{match} ,即 v_{14} 。

3)由于 v'_{match} 等于 v_{select} ,说明得到了稳定的匹配,将 $(v_{06},v_{14},0.8)$ 放入 $SEED$ 中,作为匹配成功的节点对。

4) V_0, V_1 队列都不为空,继续下一轮匹配。

(2)第二次匹配

1)匹配失败队列为空,选择待匹配队列好友数量最多的节点 v_{12} 作为 v_{select} 。

2)通过匹配算法选择与 v_{select} 的 UMS 值最大的节点 v_{02} 作为 v_{match} , UMS 值为 0.72,再 v_{match} 求出匹配的对应元素 v'_{match} ,即 v_{12} 。

3)由于 v'_{match} 等于 v_{select} ,因此得到了稳定的匹配,将 $(v_{02},v_{12},0.72)$ 放入 $SEED$ 中,作为匹配成功的节点对。

4) V_0, V_1 队列都不为空,继续下一轮匹配。

(3)第三次匹配

1)匹配失败队列为空,选择待匹配队列中好友数量最多的节点 v_{04} 作为 v_{select} 。

2)通过匹配算法选择与 v_{select} 的 UMS 值最大的节点 v_{11} 作为 v_{match} , UMS 值为 0.72,再求出与 v_{match} 相匹配的对应元素 v'_{match} ,即 v_{01} , UMS 值为 0.84。

3)由于 v'_{match} 不等于 v_{select} ,则认为匹配失败,将 v_{04} 放入匹配失败队列,然后令 v_{11} 作为新的 v_{select} 。

4)通过匹配算法选择与 v_{select} 的 UMS 值最大的节点 v_{01} 作为 v_{match} , UMS 值为 0.84,再求出与 v_{match} 相匹配的对应元素 v'_{match} ,即 v_{11} 。

5)由于 v'_{match} 等于 v_{select} ,说明得到了稳定的匹配,将 $(v_{01},v_{11},0.84)$ 放入 $SEED$ 中,作为匹配成功的节点对;同时, UMS 值超过了阈值,故将同一个聚类中已匹配的节点对的 UMS 值增大 10%,则 $UMS(v_{06},v_{14})$ 增大为 0.88。

6) V_0, V_1 队列都不为空,继续下一轮匹配。

(4)第四次匹配

1)匹配失败队列不为空,选择队列最前面的元素 v_{04} 作为 v_{select} 。

2)通过匹配算法选择与 v_{select} 的 UMS 值最大的节点 v_{13} 作为 v_{match} , UMS 值为 0.7,再求出与 v_{match} 相匹配的对应元素 v'_{match} ,即 v_{13} 。

3)由于 v'_{match} 等于 v_{select} ,则说明得到了稳定的匹配,将 $(v_{04}, v_{13}, 0.7)$ 放入 $SEED$ 中,作为匹配成功的节点对。

4) V_0, V_1 队列都不为空,继续下一轮匹配。

(5)第五次匹配

1)匹配失败队列不为空,选择队列最前面的元素 v_{05} 作为 v_{select} 。

2)通过匹配算法选择与 v_{select} 的 UMS 值最大的节点 v_{15} 作为 v_{match} , UMS 值为 0.76,再求出与 v_{match} 相匹配的对应元素 v'_{match} ,即 v_{05} 。

3)由于 v'_{match} 等于 v_{select} ,说明得到了稳定的匹配,将 $(v_{05}, v_{15}, 0.76)$ 放入 $SEED$ 中,作为匹配成功的节点对。

4) V_0, V_1 队列都不为空,继续下一轮匹配。

(6)第六次匹配

1)匹配失败队列不为空,选择队列最前面的元素 v_{03} 作为 v_{select} 。

2)通过匹配算法选择与 v_{select} 的 UMS 值最大的节点 v_{17} 作为 v_{match} , UMS 值为 0.67,再求出与 v_{match} 相匹配的对应元素 v'_{match} ,即 v_{03} 。

3)由于 v'_{match} 等于 v_{select} ,说明得到了稳定的匹配,将 $(v_{03}, v_{17}, 0.7)$ 放入 $SEED$ 中,作为匹配成功的节点对。

4) V_0 队列为空,匹配结束。

(7)结果处理与分析

$SEED$ 中的节点对为 $\{(v_{00}, v_{10}, 1), (v_{06}, v_{14}, 0.88), (v_{02}, v_{12}, 0.72), (v_{01}, v_{11}, 0.84), (v_{04}, v_{13}, 0.7), (v_{05}, v_{15}, 0.76), (v_{03}, v_{17}, 0.7)\}$,剪枝率为 50%,因此最终结果为 $\{(v_{00}, v_{10}, 1), (v_{06}, v_{14}, 0.88), (v_{01}, v_{11}, 0.84), (v_{05}, v_{15}, 0.76)\}$ 。

由匹配过程可知,经常会出现 v'_{match} 不等于 v_{select} 的情况,此时将 v_{select} 放入待匹配队列,再将 v_{match} 作为新的 v_{select} 继续进行迭代运算,直到两方都得到最满意的结果,这样得到的匹配才更加稳定。匹配过程中还会出现 UMS 值较大并超过之前设定的用来判定匹配是否为正确的阈值 α 的情况,这说明之前匹配的节点较为可靠。对节点匹配起到了正面效用,因此提高部分节点的 UMS 值作为正反馈,增加了正确节点对后续匹配的效用,降低了匹配错误节点对后续节点匹配的影响。

结束语 本文将社交网络中的同一用户识别问题分为初步用户选择匹配、稳定匹配两大步骤,并着重给出了衡量用户相似度的算法。本文的主要贡献在于:

(1)综合利用了相关的工作,并针对其不足之处提出了改进方法;

(2)创造性地提出了计算不同社交网络中节点的相似度的方案,并且解决了在大规模社交网络识别中正确节点的权重越来越低的问题,从而减少了错误匹配对后续节点匹配的影响,得到了更好的匹配结果。

今后的研究方向主要在于搜集合适的数据集,从更多维度来衡量用户间的相似度以增加识别结果的正确性。如在计算好友关系时,将好友之间最近互动的亲密度作为好友链接

边上的权重,从而更好地对好友亲密程度进行建模。并且在实际的操作中发现问题,并对算法进一步改进,以提高正确率及匹配速度。如何处理 ego-network 身份匹配失败时的节点,并增加召回率,也是今后工作的一个方向。

参 考 文 献

[1] HASSANZADEH O,PU K Q,et al. Discovering linkage points over web data[C] // Proceedings of the VLDB Endowment. 2013;445-456.

[2] IRANI D,WEBB S,LI K,et al. Large Online Social Footprints—An Emerging Threat[C] // 2009 International Conference on Computational Science and Engineering. 2009;271-276.

[3] VOSECKY,HONG D,SHEN V Y. User identification across social networks[C] // International Conference on Networked Digital Technologies. Ostrava,2009;3660-365.

[4] BARTUNOV S,KORSHUNOV A,PARK S T,et al. Joint link-attribute User Identity Resolution In Online Social Networks [C] // International Conference on Knowledge Discovery and Data Mining, Workshop on Social Network Mining and Analysis. 2012.

[5] MENG B. Research on algorithms for identifying users across multiple online social networks[D]. Dalian:Dalian University of Technology,2015;1-10. (in Chinese)

孟波. 多社交网络用户身份识别算法研究[D]. 大连:大连理工大学,2015;1-10.

[6] YU M H. Entity linking on graph data[C] // Proceedings of the 23rd International Conference on World Wide Web. 2014;21-26.

[7] LIANG W,MENG B,HE X,et al. GCM:A Greedy-Based Cross-Matching Algorithm for Identifying Users Across Multiple Online Social Networks[J]. PAISI,2015,9074;51-70.

[8] ZHOU X,LIANG X,MA Y. Cross-Platform Identification of Anonymous Identical Users in Multiple Social Media Networks [J]. IEEE Transactions on Knowledge and Data Engineering, 2016,28(2):411-424.

[9] XU X,YURUK N,FENG Z,et al. SCAN:A structural clustering algorithm for networks[C] // Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM Press.2007;824-833.

[10] LI Z J,DAI Q Q,LI R H,et al. Social Relationship Mining Algorithm by Multi-Dimensional Graph Structural Clustering[J]. Journal of Software,2018,29(3):839-852. (in Chinese)

李振军,代强强,李荣华,等. 多维图结构聚类的社交关系挖掘算法[J]. 软件学报,2018,29(3):839-852.

[11] JEH G,WIDOM J. SimRank;a measure of structural-context similarity[C] // Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'02. 2002;538-543.

[12] CHEN J M,CHEN J J,LIU J,et al. Clustering algorithms for large-scale social networks based on structural similarity[J]. Journal of Electronics & Information Technology,2015,37(2): 449-454. (in Chinese)

陈季梦,陈佳俊,刘杰,等. 基于结构相似度的大规模社交网络聚类算法[J]. 电子与信息学报,2015,37(2):449-454.