社会网络中序列行为的链接值及事件结果预测

刘秉权^{①*}, 刘峰^①, 王晓龙^{①②}, 孙承杰^①, 刘铭^①^① 哈尔滨工业大学计算机科学与技术学院, 哈尔滨 150001^② 哈尔滨工业大学深圳研究生院, 深圳 518055

* 通信作者. E-mail: liubq@insun.hit.edu.cn

收稿日期: 2015-08-28; 接受日期: 2015-10-20; 网络出版日期: 2015-12-02

国家自然科学基金 (批准号: 61272383, 61300114, 61572151) 资助项目

摘要 随着在线社会网络为其用户提供的功能不断丰富, 用户可以参加多种多样的社交活动, 将这些活动表示为对应种类的事件. 其中一些事件包括多个用户之间的交互行为, 再由这些交互的类型来决定该事件的最终结果. 将每个行为表示为用户之间的链接, 用链接的极性值来表示交互的类型, 对于这些链接值预测的研究称为链接预测. 研究者们对于链接预测问题提出了很多方法, 其中大多是在静态网络环境中孤立地预测每条链接边的值. 他们将这类事件中的用户行为看作“普通的用户行为”, 即这些链接被看成独立且离散的存在, 并未考虑这些链接之间在事件生成中的内在联系. 本文通过将事件中用户行为表示为“序列行为”, 提出了基于序列行为的链接值预测和事件结果预测的方法. 该方法能将这种序列信息建模, 在模型中这种序列信息得以累积, 利用这种累积来预测序列中下一个或多个时刻的链接值. 在完成对整个序列的预测的同时, 通过累积信息可以推断出该事件的结果. 实验结果表明, 我们的方法明显优于静态、孤立的链接预测方法, 而且可以更好地推断出包含该序列行为事件的最终结果.

关键词 序列行为 链接预测 事件结果预测 序列建模 社会计算

1 引言

伴随着互联网技术的高速发展, 社会网络服务 (SNS, social networks services) 也随之出现并得到了快速普及. 目前, 各类在线 SNS 中的注册用户数目十分巨大, 越来越多的研究者和服务商开始关注如何为这些用户提供更好的服务^[1]. 为社会网络中的用户提供丰富多样的用户行为 (user behaviour) 是 SNS 的一个重要功能, 其中作用于用户之间的行为被称为用户之间的交互 (user interaction). 如果将用户看成图中的节点 (node), 这些用户交互行为可以用节点之间的链接 (link) 来表示. 对链接预测方法进行深入细致的研究, 相信会对社会网络的研究和发展具有毋庸置疑的重要意义.

例如 Wikipedia¹⁾, Epinion²⁾, Slashdot³⁾ 等网站, 它们既为用户提供了用户生成内容 (UGC, user generated content) 的发布服务, 也允许用户之间对对方内容进行评论、修正与补充. 一些网站还采取

1) <https://www.wikipedia.org/>2) <http://www.epinion.com/>3) <http://slashdot.org/>

图 1 (网络版彩图) 预测序列行为中的链接值和事件最终结果

Figure 1 (Color online) Predicting sequence behaviour link values and event result

用户自己选举管理员的方式来进行管理, 例如 Wikipedia 的管理员选举 (admin role promotion) [2]. 如果使用图来对社会网络进行建模, 图中的节点表示用户, 以上这些用户行为都可以用节点之间的链接来表示, 用户交互的具体内容可以用链接类型和链接边上的值来表示. 包含这种用户或是实体之间链接的数据集, 一般称为链接数据或是关系数据. 对这些链接数据集进行预测和描述的数据挖掘研究, 统称为链接挖掘 (link mining). 其中用来处理对象间是否存在链接的预测, 以及预测链接类型的研究称为链接预测 (link prediction) [3, 4]. 通过对链接预测相关问题的研究, 研究者可以更好地对 SNS 中的用户交互进行建模和分析, 从而为其用户提供更好的服务.

在实际社会网络环境下, 表示用户行为的链接网络并不是静态存在的, 而是随着用户之间不断发生的各种交互行为动态生成的. 其中如何对这类动态变化进行建模, 并预测将要发生的行为是社会网络研究中的一个难点 [5]. 社会网络中很多用户行为都是针对于某一特定事件, 这类事件一般包括多个用户的一系列行为, 这些行为按序列的方式生成, 在所有行为完成的同时, 事件的最终结果也得以确定, 我们把该类事件中这种连续的行为称作“序列行为”.

以 Wikipedia 的管理员选举的过程为例, 如图 1 所示. 当一个候选人 (candidate) 想要成为管理员时, 需要一系列的维基百科成员 (member) 分别为其投票, 再以最后的投票结果来决定其是否会晋升为管理员. 如图 1 示例, 这种投票包括支持票 (positive) 和反对票 (negative). 用户在进行投票时, 可以看见前面用户的投票结果. 只有在全部投票结束后, 一般以时间截止, 候选人才能够知晓选举的最终结果. 如果将选举看成事件 (event), 每一位用户的投票看作“用户行为”, 选举的成败与否可以看作“事件的最终结果”, 那么一次管理员选举过程可用一个序列行为来表示.

本文主要贡献如下: (1) 针对真实环境下某些事件中用户行为的产生方式, 本文提出了将该类事件用“序列行为”来表示. 从而将静态离散的极性链接网络扩展为包含极性链接按顺序产生的动态事件, 这种表示方法更接近真实链接网络动态生成的过程. (2) 通过对链接网络结构的分析, 本文从链接网络中提取了节点状态特征和用邻居交互特征, 设计并实现了高效提取这些特征的算法. (3) 为了对序列行为进行建模, 本文设计了结构合理的递归神经网络模型, 并提出了序列行为的链接值及事件结果的预测方法. 在来自真实环境 (维基百科) 的链接数据集实验中, 结果表明我们的方法能提高对单个链接值的预测性能, 同时也能更好地预测出事件的最终结果.

本文的余下内容安排如下: 第 2 节是对于链接预测问题的研究现状进行介绍. 第 3 节介绍从链接网络中提取特征的方法. 第 4 节介绍对序列行为的建模, 以及相应的链接预测和事件结果预测方法. 第 5 节是实验部分. 第 6 节为结论.

图 2 预测未来网络中的新链接

Figure 2 Predicting new links in future network

2 链接预测问题研究现状

链接存在预测问题是链接预测较早期就开始研究的问题, 链接预测 (link prediction) 的命名也是由此项任务产生. Getoor 和 Diehl^[4] 给出的经典链接预测定义为“通过两个节点的属性和观测到的链接, 来预测这两个节点之间是否会存在一条链接”. 例如预测社交圈内谁和谁可能会成为朋友^[6], 或是谁会信任谁^[7]. 如图 2 所示, 这类问题的研究内容为根据现有的对象属性和链接网络结构, 来预测链接网络中未来时刻链接的变化.

在研究共同创作 (co-author) 问题时, Nowell 和 Kleinberg^[8] 也给出了类似的链接预测定义: “通过观察网络某一时刻的状态, 来预测社会成员之间将要发生的互动”. 他们尝试了多种作者的相似度计算方法, 并根据相似度的高低来预测两位作者是否会在未来共同创作一篇文章. 类似的研究还有 Richardson 等^[9] 利用信任网络中的路径拓扑结构来预测谁将会信任谁. Dong 等^[10] 列出了多种可用于链接预测的用户相似度计算方法, 并进行了比较. Dong 等^[11] 也比较了多种相似度度量方法在共同创作上的表现. 除此之外, Guo 等^[12] 使用用户相似度和链接产生的时序信息来预测 Enron 邮件发送, 他们将时序信息变换为用户交互的频度, 并假设这种频度会影响用户未来发生交流的可能性.

这类研究的目的在于预测链接网络未来的变化, 该类研究大多基于用户相似度计算, 预测高度相似的用户在未来某一时刻会进行某种行为. 例如共同创作问题经常以 10 年的文章发表情况, 来预测作者在未来 5 年可能共同创作的可能性. 这些方法并不适用于时间上高度联系的事件内序列行为, 因为其并未充分考虑行为发生的序列性, 或是只对链接发生频度进行简单处理. 另外这种方式与推荐方法相似, 即为用户推荐相似的用户, 其方法只能推荐一种类型的行为, 同样不适用于包含多种行为的序列行为.

随着在线社会网络服务的发展, 用户可以使用越来越复杂的功能, 例如从原来的只能表达“粉, 赞”这类正极性链接 (positive links), 增加到可以表达“黑, 踩”等负极性链接. 随着这些功能的出现, 研究者们也开始研究对负极性链接 (negative links) 的预测^[13]. 链接极性值预测的主要研究对象为能提供正、负两种或多种评价、标签等服务的社会网络. 例如 Epinions 这种点评网站, 用户可以表示相信某一用户对某一商品的评价, 或是不相信该用户的评价^[14,15]. 又如 Essembly 提供可标记某一用户为‘Friend’, ‘Ally’ 或是 ‘Nemesis’ 的功能, 同时提供对某一用户观点表示 ‘Agree’ 或 ‘Against’ 的服务^[16]. 下面详细介绍关于这项任务的研究现状.

因为引入了包含负极性值的链接, 以及用户行为的有向性, 使得链接网络结构变得更加复杂, 能提取出的特征也增加了. 例如 Slashdot Zoo 中的用户直接可以标明对方是朋友 (friend) 或是敌人 (foe)^[17], 又例如 Wikipedia 等网站允许用户自身选举时的投票预测. 以上这些关系, 同样可以用图来表示, 如图 3 所示, 节点表示用户, 每个链接表示用户之间发生过的某种行为, 例如评价或态度, 链接

图 3 (网络版彩图) 预测极性网络中的链接值

Figure 3 (Color online) Predicting link values in signed networks

边上的值用来表示具体的相应内容。

基于这些数据, 研究者们设计了很多方法对用户之间的链接值进行预测。比较常用的方法是将对链接值的预测转化为分类问题, 使用训练集样本的特征训练相应的分类器, 再用该分类器来预测测试集中两个节点之间的链接值。Leskovec 等^[18]使用 Logistic 回归模型来预测两个节点的链接值的极性, 他们隐去网络中的一条链接边, 从其余边提取特征, 之后使用这些特征训练分类器并进行预测。Liu 等^[19]使用支持向量机来预测链接的极性值, 并分析了不同特征对于链接极性值的影响。Liu 等^[20]使用深度置信网络来对链接的极性值进行预测, 借助于受限 Boltzmann 机的抽象能力, 他们在多个链接数据集上取得了良好的结果。

这类研究的目的在于预测链接网络中链接的极性值, 即尝试对链接网络用户行为进行分类。现有研究大都隐藏掉部分链接边的值, 使用可见部分链接网络来预测隐藏部分链接边的值。但是这种静态的预测方法, 没有考虑到事件内序列行为的顺序关系。此外因为每条链接都被孤立分类, 也不适合预测一系列行为后事件的最终结果。所以这些方法可能不适用于解决序列行为中的预测问题, 这是因为对序列行为处理时需要考虑其中用户行为产生的顺序信息。

现有的研究大都关注于不确定未来时段的用户行为预测 (链接存在预测), 或是对静态网络中的用户行为进行分类 (链接极性值预测)。很少有研究关注于短时间内某一事件中的序列行为, 以及分析这些行为导致的事件结果。为了能解决本文的研究问题, 需要解决如图 1 示例的序列行为中链接值和事件结果预测, 既需要同时考虑用户之间链接的极性值预测, 也要充分利用这些行为之间的顺序关系。某种程度上可以看作同时实现链接存在预测问题 (图 2) 的求解, 以及链接极性值预测问题 (图 3) 的求解。

3 链接网络中的特征提取方法

本节介绍如何从极性链接网络中抽取用于链接值预测的特征, 包括节点状态特征和邻居交互特征。以链接边为单位来从链接网络中抽取特征, 如图 3 中所示, 较为直观的特征包括边上两个节点的出入度, 较为间接的特征可以从这两个节点对于某一第三节点的链接来抽取。

3.1 链接网络的建立和存储

图的存储一般采用邻接矩阵或是链表的方式。邻接矩阵方法使用 [节点数目 $\times$ 节点数目] 的矩阵来存储网络, 网络中所有位置初始值为 0, 如果两节点之间存在一条边, 则在矩阵中相应行列位置加 1。

邻接矩阵具有边可直接访问, 以及可以使用矩阵运算来进行各类路径的计算等优点. 但它并不适合存储社会网络数据, 这是因为节点数目巨大, 而且矩阵十分稀疏, 这会导致浪费巨大的存储空间.

链表方法使用一个长度为节点数目的数组来存储所有节点链表的首地址, 与该节点有边的节点采用链表方式添加到该节点链表的尾部. 该方法虽说在访问边的速度上较慢, 这是因为需要遍历该节点的链表才能找到与该节点有边的节点, 但在存储空间上是动态分配, 很适合存储社会网络这种稀疏矩阵. 为了优化边的查找速度, 我们采用了一种双向记录的存储方式.

每一个节点的结构体 NodeLinks 存储 4 个链表: 以该节点为起点的正极性链接边 posLinksTo, 以该节点为起点的负极性链接边 negLinksTo, 以该节点为终点的正极性链接边 posLinksFrom, 和以该节点为终点的负极性链接边 negLinksFrom. 再用一个数组 Networks = NodeLinks[n] (n 为节点数目) 来存储所有节点结构体的地址, 在添加每一条从 u 到 v 的链接边 $e(u, v, \text{link_value})$ 时, 根据 link_value (链接值) 来选择放在正极性还是负极性的链表中, 分别添加到 Networks[u] 为起点的链表, 和以 Networks[v] 为终点的链表. 这么存储最大的好处是能在线性时间内抽取两个节点之间对于第 3 个节点的间接关系, 将在算法分析部分作详细说明.

3.2 抽取节点状态特征

首先我们从链接网络中抽取较为直观的节点状态特征, 假设一个节点的状态可以由它所发出和接收的链接边来表示, 那么节点状态特征主要来自于节点自身相应的出入度. 对于每一条从 u 到 v 的链接边 $e(u, v)$, 我们分别抽取 u 和 v 的节点特征. $d_{\text{out}}^+(u)$ 表示节点 u 给出的正极性链接数目 (正出度), $d_{\text{out}}^-(u)$ 表示负出度, $d_{\text{in}}^+(u)$ 表示节点 u 得到的正极性链接数目 (正入度), $d_{\text{in}}^-(u)$ 表示负入度. 对节点 v 也抽取相应的 $d_{\text{out}}^+(v)$, $d_{\text{out}}^-(v)$, $d_{\text{in}}^+(v)$ 和 $d_{\text{in}}^-(v)$. 最终对于任意一条链接边, 它拥有 8 种节点状态特征, 详细计算步骤见算法 1.

算法 1 抽取节点状态特征的算法

Input: link network Networks, edge $e(u, v)$

Output: array of node state features nodeFeatures

- 1: nodeFeatures = float[8]; $d_{\text{out}}^+(u) = 0$; ... , $d_{\text{in}}^-(v) = 0$;
 - 2: $d_{\text{out}}^+(u) = \text{Len}(\text{Networks}[u].\text{posLinksTo})$; $d_{\text{out}}^-(u) = \text{Len}(\text{Networks}[u].\text{negLinksTo})$;
 - 3: $d_{\text{in}}^+(u) = \text{Len}(\text{Networks}[u].\text{posLinksFrom})$; $d_{\text{in}}^-(u) = \text{Len}(\text{Networks}[u].\text{negLinksFrom})$;
 - 4: $d_{\text{out}}^+(v) = \text{Len}(\text{Networks}[v].\text{posLinksTo})$; $d_{\text{out}}^-(v) = \text{Len}(\text{Networks}[v].\text{negLinksTo})$;
 - 5: $d_{\text{in}}^+(v) = \text{Len}(\text{Networks}[v].\text{posLinksFrom})$; $d_{\text{in}}^-(v) = \text{Len}(\text{Networks}[v].\text{negLinksFrom})$;
 - 6: Save above 8 node state features in nodeFeatures[1-8];
-

3.3 抽取邻居交互特征

对于链接边来说, 除了上述能较为直接观察到的节点状态外, 链接网络中还存在着两端节点与第 3 节点之间的交互, 这种交互中最为直接的为两端节点与公共邻居之间的交互. 例如 Jaccard 系数 (Jaccard coefficient) 的计算就是两用户的公共邻居占他们全部邻居的比例. 在不考虑链接方向前提下, 使用 $Ne(u)$ 表示与 u 直接相连的节点集合, $Ne(v)$ 表示与 v 直接相连的节点集合, 他们之间的共同邻居集合为 $CNe(u, v) = Ne(u) \cap Ne(v)$. 首先统计他们的共同邻居数目, 这里有两种统计方式, 按节点数目统计和按链接数目统计. 有两种方式是因为节点之间的交互次数可能不止一次, 为了更全面的研究特征, 本研究同时统计这两种特征. 用 $c_{\text{Ne}}^n(u, v)$ 表示按节点数目的统计结果, 其值等于集合 $CNe(u, v)$ 中节点的数目, 即两节点的公共邻居数目. 用 $c_{\text{Ne}}^e(u, v)$ 表示按链接边数目的统计结果, 其值为 $\sum_w \{w$

与 u 的链接数目 $\times w$ 与 v 的链接数目}, 其中 $w \in \text{CNe}(u, v)$. $c_{\text{Ne}}^e(u, v)$ 的值在网络中的实际意义为不考虑链接方向时, 从节点 u 到 v 所有可能的长度为 2 的路径数目.

接下来统计链接边两端节点与第 3 节点之间的交互信息. 从 $\text{CNe}(u, v)$ 中任选一个节点 w , 如图 3 所示, 链接网络中有两种边的方向, 每条边可以有两种极性值, 那么 w 与 u 可以有 $2 \times 2 = 4$ 种交互关系, 同样 w 与 v 也有这样 4 种交互关系. 所以 w 与 u, v 共有 $4 \times 4 = 16$ 种关系. 举例说明, 如图 3 中 B 和 E 作为 u 和 v , 那么他们都给了 C 正极性的链接, 如果把 C 看成 w , 这种节点 w 的统计数目记作 $c(u \rightarrow w \rightarrow v)$, 详细可见算法 2 的第 5~15 行. 再如用节点 A 代替 E, 记为 u , 那么 C 应该计数到 $c(u \leftarrow w \leftarrow v)$. 每个节点在每种情况中只计算一次, 但是节点可以出现在多种情况中. 这样共有两种共同邻居加 16 种与邻居交互, 共 18 种邻居交互特征, 详细计算步骤见算法 2.

算法 2 抽取邻居交互特征的算法

Input: link network Networks, edge $e(u, v)$

Output: array of neighbour interaction features neighbourFeatures

```

1:  $c_{\text{Ne}}^n(u, v) = 0$ ;  $c_{\text{Ne}}^e(u, v) = 0$ ; KindsOfNeighbors = List[16];
2: CommonNeighbor =  $\emptyset$ ; KindsOfNeighbors[1 - 16] =  $\emptyset$ ;
3:  $c(u \rightarrow w \rightarrow v) = 0$ ; ... ,  $c(u \leftarrow w \leftarrow v) = 0$ ;
4: for each node  $w$  in Networks[ $u$ ].posLinksTo do
5:   if Networks[ $v$ ].posLinksTo.count( $w$ ) > 0 then
6:      $c_{\text{Ne}}^e(u, v) += \text{Networks}[v].\text{posLinksTo}.\text{count}(w)$ ;
7:     if  $w$  is not in CommonNeighbor then
8:       Add  $w$  to CommonNeighbor;
9:        $c_{\text{Ne}}^n(u, v) += 1$ ;
10:    end if
11:    if  $w$  is not in KindsOfNeighbors[1] then
12:      Add  $w$  to KindsOfNeighbors[1];
13:       $c(u \rightarrow w \rightarrow v) += 1$ ;
14:    end if
15:  end if
16:  Repeat line.5 to line.15 with different conditions to count other 3 different kinds of neighbor interactions as
    if(Networks[ $v$ ].negLinksTo).count( $w$ ) > 0 { replace line.13 as  $c(u \rightarrow w \leftarrow v) += 1$ ; }; then
    if(Networks[ $v$ ].posLinksFrom).count( $w$ ) > 0 { replace line.13 as  $c(u \rightarrow w \rightarrow v) += 1$ ; }; and
    if(Networks[ $v$ ].negLinksFrom).count( $w$ ) > 0 { replace line.13 as  $c(u \rightarrow w \rightarrow v) += 1$ ; };
17: end for
18: Repeat line.4 to line.17 with different conditions to count other 12 ( $3 \times 4$ ) different kinds of neighbor interactions as
    for(each node  $w$  in Networks[ $u$ ].negLinksTo) { count  $c(u \rightarrow w \rightarrow v)$ ,  $c(u \rightarrow w \leftarrow v)$ ,  $c(u \leftarrow w \rightarrow v)$ ,  $c(u \leftarrow w \leftarrow v)$  }; then
    for(each node  $w$  in Networks[ $u$ ].posLinksFrom) { count  $c(u \rightarrow w \rightarrow v)$ ,  $c(u \rightarrow w \leftarrow v)$ ,  $c(u \leftarrow w \rightarrow v)$ ,  $c(u \leftarrow w \leftarrow v)$  }; and
    for(each node  $w$  in Networks[ $u$ ].negLinksFrom) { count  $c(u \rightarrow w \rightarrow v)$ ,  $c(u \rightarrow w \leftarrow v)$ ,  $c(u \leftarrow w \rightarrow v)$ ,  $c(u \leftarrow w \leftarrow v)$  };
19: Save  $c_{\text{Ne}}^n(u, v)$  and  $c_{\text{Ne}}^e(u, v)$  in neighbourFeatures[1 - 2];
20: Save  $c(u \rightarrow w \rightarrow v)$ , ... ,  $c(u \leftarrow w \leftarrow v)$  in neighbourFeatures[3 - 18];

```

3.4 特征提取算法复杂度分析

以上已经详细介绍了如何存储链接网络, 以及如何从链接网络中提取的算法. 假设网络中共有 n 条链接, 有 m 个节点, 那么我们需要 $2 \times n + m$ 的存储空间来实现双向邻接表的存储, 链接网络中基本都是 $m < n$, 所以该存储方法在空间上需要 $O(n)$ 的存储空间. 现在分析算法 1 的时间复杂度, 本文链表使用 Python 的广义表类 List 实现, 可在单位时间内返回一个链表中的长度, 所以算法 1 在计

算 n 条链接边的特征时的总开销为 $8 \times n$, 所以该算法可在 $O(n)$ 时间完成. 这里如果只采用单向链表存储, 即只存储节点为起点的正极性链接边 `posLinksTo` 和该节点为起点的负极性链接边 `negLinksTo`. 那么在计算该节点的正、负入度时, 就需要遍历其他所有节点的链表, 来找到以该节点为终点的边. 此时计算一条边的时间复杂度就变为 $(4 + 4 \times n)$, 抽取 n 条边的时间复杂度就变为了 $O(n^2)$. 可见牺牲存储空间换来了计算上的高效, 在机器内存充足的情况下, 这些空间上的开销还是很划算的.

再来分析算法 2 的时间复杂度, 通过分析, 算法中第 5 ~ 15 行可以看成对一种邻居交互特征的抽取, 每次可以看成是对 v 部分邻居的遍历. 再来分析第 4 ~ 17 行中的 `for` 循环, 是对 u 所有邻居的遍历, 而 `for` 循环内的语句可以看成是对 v 所有邻居的遍历, 那么假设单个节点最多的邻居数目为 l , 在一般情况下 $n \gg l$, 则全部循环可在 $(l \times 4l)$ 的时间内完成. 那么在计算 n 条链接边的特征时的总开销为 $(4l^2 \times n)$, 所以该算法可在 $O(l^2 n)$ 时间完成. 双向存储链接边同样在此算法中起到了减少时间开销的作用, 主要是减少了遍历两个节点所有邻居的时间. 如果不存储以节点为终点的链接, 那么算法 2 因为要循环遍历 u 和 v 的邻居来判断, 时间复杂度就会增加到 $O(n^2)$, 而提取全部边特征的时间就会变为 $O(n^3)$.

真实环境中, 因为节点度数符合指数分布法则, 导致链接网络中每个节点平均链接边数目较小, 而且不会随着网络规模 n 增大而迅速增加. 所以从网络中提取全部边的 26 种特征基本可以在线性时间开销内完成. 实际中, 使用主流 PC 机从包含 10 万节点、50 万链接边的链接网络中提取全部特征可在 10 分钟内完成.

4 序列行为的建模和预测方法

本节主要介绍对序列行为建模的模型, 以及如何利用模型进行相应的预测任务. 还介绍了如何训练该模型, 以及该模型的测试方法.

4.1 递归神经网络

递归神经网络 (RNN, recurrent neural network) 在结构上与前向神经网络 (FNN, forward neural network) 相似, 同样包括输入层、输出层, 以及一个或若干个隐藏层. 不同之处在于, RNN 内包含一个或多个递归连接 (recurrent connection)^[21]. 这些递归连接可以将网络中某一层的上一时刻的状态 (state) 作为该层或其他层该时刻的输入, 递归连接同样拥有权重.

这种递归连接使设计合理的 RNN 拥有对序列数据进行建模的能力, 这种能力适用于连续的信号处理任务. Connor 等^[22] 使用 RNN 来预测 4 个月中每天气温的变化, 取得了很好的结果, 他们还尝试了对这 4 个月的用电量进行了预测. Graves 等^[23] 通过改良 RNN 的训练方法, 使其可以在不需要先进行时间片切分的前提下, 对 TIMIT 语音库中的语句进行连续的单词识别, 并取得了优于隐 Markov 模型的方法. Graves 和 Schmidhuber^[24] 通过改变递归连接的方向, 使其适用于手写体识别的任务.

除了信号处理外, RNN 还广泛地应用于自然语言处理任务中^[25], 其思想主要是将语句看作词的连续信号, 使用 RNN 来对这种连续词组成的句子进行如语义理解和情感分析等任务. Irsoy 和 Cardie 等使用含有多个递归隐藏层的 RBM 来预测新闻语料中句子所表达的观点. Wang 等^[26] 使用含有 LSTM 的 RNN, 来预测 Twitter 中用户语句所表达的情感极性. 这些研究给我们以启示, 如果能使用 RNN 对序列行为进行建模, 那么有可能较好地预测出下一时刻用户行为的类型. 而在完成对整个序列的输入后, 也有可能得到整个序列所蕴含的事件最终结果.

图 4 递归神经网络按时间展开示例

Figure 4 Extension of recurrent neural network by time

如图 4 所示, 为一个三层结构的 RNN, 除了和 FNN 相同的输入层到隐藏层的连接 W_{ih} 和隐藏层到输出层连接 W_{ho} 外, 还拥有隐藏层到隐藏层的递归连接权重 W_{hh} . 为了便于观察, 图中将每一个时刻的 RNN 计算展开来显示. 将第 t 个时刻输入层的输入记为 v_t , 则对应 t 时刻隐藏层状态 h_t 为

$$h_t = \begin{cases} f(W_{ih}^T v_t), & t = 1; \\ f(W_{ih}^T v_t + W_{hh}^T h_{t-1}), & t > 1, \end{cases} \quad (1)$$

其中 f 为隐藏层单元的激活函数, 再计算 t 时刻输出层状态 o_t 与一般的 FNN 相同:

$$o_t = g(W_{ho}^T h_t), \quad (2)$$

其中 g 为输出层单元的激活函数.

对于上述 RNN 有一种较为直观的理解方式, 可以看作在网络中增加了一个与隐藏层维度相同的状态层 state, 它与隐藏层之间存在着全连接. 在 $t = 1$ 的时刻, state 为空, 用输入 v_1 激活隐藏层得到 h_1 后, state = h_1 . 在 $t = 2$ 时刻, 通过式 (1), 使用 v_2 和 state 共同激活隐藏层得到 h_2 后, state = h_2 . 如此递归, 直到整个输出序列 V 完成输入. 这时 state 可以看成是对之前所有隐藏层状态的一个累积状态, 从而能表示输入序列样本依次对于最终结果的影响. 同时 RNN 在每一个时刻都有一个对应输出 o_t , 因为 state 的存在, 除了 t 时刻输入样本 v_t 外, o_t 还会受到前 $t - 1$ 个时刻输入样本的影响. 这就使得 RNN 拥有了能够对整个序列进行建模的能力, 从而对于序列样本的处理能力要高于 FNN.

在训练 RNN 时, 可以使用改进的反向传播方法 (BP, back propagation) [27], 或是时间展开反向传播方法 (BPTT, back propagation through time) [28]. 前者的处理方式与上面所介绍的状态层理解方法类似, 在每一个 t 时刻, 将 RNN 展开为包含输入层和状态层两个输入层的 FNN, 使用 BP 算法根据 t 时刻输出来更新权重. BPTT 则是更进一步的将 RNN 展开, 需要记录下前面所有的隐藏层的状态, 同样是在每一个 t 时刻, 将隐藏层输入展开为 $t - 1$ 时刻的输入层和隐藏层, 再将 $t - 1$ 时刻的隐藏层输入展开为 $t - 2$ 时刻的输入层和隐藏层, 直到 $t = 1$ 时刻停止. 之后再使用 BP 算法从 t 时刻输出来向前更新权重, 为了避免 BP 算法在长距离上失效, 可以限制展开步数, 或是加入长短距离记录 (LSTM, long short term memory) 单元 [29]. 向回展开 n 个时段的 BPTT 算法称为 BPTT _{n} 方法, 当 $n = 1$ 时, BPTT₁ 与展开状态层的 BP 学习方法等效, 这里不再累述 BP 算法的细节了.

图 5 用于序列行为建模的 RNN 模型结构

Figure 5 The RNN structure of modelling sequence behaviour

4.2 对事件中序列行为建模和预测

为了同时预测序列行为中的链接值和事件最后结果, 我们设计了如图 5 所示的 RNN 网络结构. 该神经网络结构包含三层神经元: 输入层、隐藏层和输出层. 其中隐藏层神经元连接到隐藏层神经元的权重 W_{hh} 为递归连接, 输出层的链接单元 (link output) 用来预测代表用户行为的链接极性值, 事件单元 (event output) 用来预测事件的最终结果. 输入层神经元选用线性输入方式, 不对样本进行任何操作直接存入输入层. 为了避免不同维度上的样本特征取值范围差别较大对模型造成的影响, 我们将所有特征样本归一化到 $[0,1]$ 区间的实数值. 隐藏层和输出层都选用 Sigmoid 函数作为激活函数, 即式 (1) 和 (2) 中的 $f(x) = g(x) = 1/(1 + e^{-x})$.

在使用上述 RNN 来预测链接值和事件结果时, 随着序列行为中的每个样本输入到该 RNN 的输入层中, 在第 t 时刻, 隐藏层 h_{t-1} 中包含了前 $t-1$ 个时刻链接行为的累积信息, 配合 t 时刻的输入链接样本来激活隐藏层 h_t . 之后再激活输出层, 就可以从链接单元中得到该样本所表示链接极性类别的概率值. 与此同时, 事件的最终结果所属类别的概率值也可以在事件单元中得到. 其中事件单元的取值会随着接下来时刻输入的样本发生改变, 特别是在序列的早期输入时期, 因为这时事件的最终结果还不确定. 但在序列中最后几个样本输入时, 这种变化变得平缓, 这时事件的结果已经基本确定了.

采用这种多任务学习方法, 即只使用一个 RNN 网络来同时预测序列行为中的链接值和事件结果, 能更好地同时解决这两个问题. 这是因为这两个任务之间存在着较为密切的联系, 事件结果是由一连串不同极性值的链接来决定的, 同时事件结果的成败也在一定程度上能反映出链接值的分布. 例如序列前中期正极性较多的序列中, 事件结果也是正极性的. 而在序列后期, 预测到事件可能为正极性时, 后期的链接值也可能更偏向于正极性. 而且 RNN 能利用这一连串行为的序列信息, 从而更好地将两个任务联合起来.

4.3 训练和测试方法

为了解决图 1 中所示的预测序列中的链接值和事件结果问题, 我们用了如下步骤来处理训练数据和测试数据. 如图 6 所示, 每一个链接序列表示一个事件中的序列行为. 可以看到, 每个序列都由许多包含不同链接值的用户行为组成. 我们取出每个序列中前中期的链接行为来构建链接网络, 并按第 3 节所描述的方法来提取样本特征. 这些样本顺序依旧保持不变, 用来构成训练数据集. 这时后面用于预测的用户链接行为并未出现在链接网络中, 其链接值处于未知状态. 同样抽取其第 3 节介绍的特征, 这些样本顺序依旧保留, 构成测试数据集.

在训练模型时, 我们依次使用每个序列中的训练样本来更新模型. 即先用第一个序列中的训练样

图 6 序列行为的存储和处理

Figure 6 The storage and processing of sequence behaviour

本来更新模型, 之后将 RNN 模型中的隐藏层状态置为空, 再使用第二个序列中的样本来更新模型. 如此循环, 直到最后一个序列被学习. 在更换训练序列时, 必须要重置模型中的隐藏层状态. 这是因为, 如果不进行重置, 模型会认为这两个序列属于一个序列, 从而导致训练的错误.

在完成训练后, 使用模型进行测试时, 需要先使用对应的训练数据来激活隐藏层到正确的状态, 之后再输入对应的测试样本. 即在测试第一个序列时, 先依次使用第一个序列中的前 t 个训练样本来激活模型, 使 RNN 模型中的隐藏层状态激活到 t 时刻的状态, 这时再输入 $t+1$ 时刻测试链接样本, 就能得到正确的预测值. 之后再依次输入 $t+2$ 时刻测试链接样本来预测对应的链接值, 直到该序列测试完成. 同时我们记录事件单元的输出结果, 用其均值计算拥有该序列行为事件的最终结果. 在完成第一个序列的测试后, 同样需要重置模型中隐藏层的状态, 才能开始对下一个序列的预测.

5 实验和分析

本节主要介绍本文的实验部分, 首先介绍实验所采用的数据来源和数据分布特点. 之后对相应模型的参数选择以及和模型的实现方法进行说明, 并介绍了基线方法. 最后是实验结果的介绍和对结果的分析与讨论.

5.1 数据集介绍

本文使用 Wikipedia 管理员选举数据来进行实验, 该数据集可以在斯坦福网络分析计划⁴⁾(SNAP, Stanford network analysis project) 网站下载得到, 该数据集包含了 2003 年至 2013 年的维基百科管理员选举数据. 每次选举可以看作一个事件, 选择的成败结果作为事件的结果. 每次选择都包含若干用户对于候选人的投票, 每次投票可以看作一条表示用户行为的链接, 链接的极性值表示投票是赞成或是反对. 包含这些链接的序列可以看作是事件的序列行为, 对某一时刻链接值的预测即对投票类型的预测, 对事件结果的预测即对选举结果的预测.

为了进一步分析链接网络的结构特点, 我们对整个管理员选举数据集中拥有不同度数节点出现的频率进行了统计, 如图 7 所示. 从图中可以明显看出, 出现频率高 (Y 值较大) 的节点的度数较小 (X 值较小), 由此可知大部分节点拥有较小的度数, 而拥有高度数的节点出现频率很低. 特别是度数达到一定数量的节点出现频率极低, 图中几乎平行于 X 轴的“长尾”也反应了这一点. 为了方便观察, 将

4) <http://snap.stanford.edu/>

图 7 节点度数的分布 (X, Y 轴为线性)

Figure 7 Node degrees distribution (linear)

图 8 节点度数的分布 (X, Y 轴为对数 ($\log_{10}$))Figure 8 Node degrees distribution ($\log_{10}$)

表 1 数据集中包含的样本数目

Table 1 Number of samples in dataset

Sample	#Positive	#Negative	#All	% of positive
All elections	1700	1673	3373	50.40
All votes	114865	48658	163523	70.24
Train votes	92961	36457	129418	71.83
Test votes	21904	12201	34105	64.22

图 7 的 X, Y 都取 10 的对数, 得到图 8. 可以看出, 图 8 中的节点度数分布更趋近于线性, 说明了在该网络中, 节点度数符合指数法则分布 (power-law distribution).

我们使用 4.3 小节中介绍的方法来构建训练集和测试集. 为了充分利用数据集中行为的序列特性, 我们舍弃只包含 5 次投票以下的选举活动. 将每次选举中的投票按时间从早到晚排序, 之后使用每个序列中前 80% 时序的投票作为训练集, 后 20% 时序的投票作为测试集. 得到的样本数目和类型见表 1, 因为没有采取任何样本平衡方法, 所以不同类型样本分布存在不平衡特性. 而且训练集和测试集中投票样本比例存在较大偏差, 这将会对我们方法的表现造成一定影响. 但是为了尽量贴近真实社会网络情况, 我们不会对样本进行任何采样处理.

5.2 实验设置

为了能较好地评价我们方法的性能、评价指标, 我们参考了 Nowak 等^[30] 多分类图像的评价方式. 首先计算整体样本的准确率 (accuracy), 之后分别计算了模型识别正、负极样本的 F 值 (F1). 除此之外, 基于接收器可控特性 (ROC, receiver operating characteristics) 的曲线下面积 (AUC, area under curve) 很适合用于多分类器的评价^[31], 我们也采用了这种评价指标.

在 Wang 等^[32] 关于解决链接预测问题的方法综述介绍中, 支持向量机 (SVM, support vector machine) 是一种常用于预测链接值的模型. 为了更好地对比我们方法的表现, 我们在同样的数据集上实现了 SVM 模型来分别对链接值和事件结果进行了预测. 同时, 我们也训练了前向神经网络 (FNN) 来与我们的递归神经网络 (RNN) 方法进行比较. 因为没有利用序列信息, SVM 和 FNN 可以看作是对链接进行静态、孤立的预测方法. 除此之外, 我们还进行了 FNN 和 RNN 模型对于单任务 (single) 和多任务 (multiple) 学习结果性能的对比. 单任务即只进行序列行为中链接值的预测任务, 或是最终

表 2 参数选择
Table 2 Parameter selection

Model	Hidden	Epoch	Link (%)	Event (%)	Average (%)
RNN	5	20	77.70	89.19	83.44
	10	30	78.04	89.76	83.90
	20	50	77.81	89.32	83.56
FNN	5	20	73.85	80.95	77.40
	10	30	75.61	83.74	79.67
	20	50	75.22	84.78	80.00

事件结果的预测任务中的一个. 而多任务则同时利用两个不同的输出单元同时解决这两个任务.

在方法的具体实现过程中, 我们使用 Python 编程来处理数据集并抽取第 3 节介绍的 26 种特征, 并按 4.3 小节介绍的方法生成对应的测试集和训练集. 在训练模型时, 我们使用 Liblinear^[33] 所提供的 Python 接口来训练 SVM, 使用 PyBrain^[34] 所提供的 Python 接口来训练 FNN 和 RNN, 并进行测试得到相应单元的输出结果. 在预测样本标签时, 我们使用全部测试样本对应任务的输出层单元的平均值来作为判断正、负样本的阈值.

因为模型的参数会在较大程度上影响模型的表现. 在训练 SVM 时, 我们使用了 Liblinear 中的 L2-SVM, 对应参数 “-s 2”, 添加了偏置 “-b 1”, 在惩罚项 C 值的选取上, 我们尝试了 {1,2,4,8,16,32} 中的值, 在 “-c 8” 时取得最好模型表现作为实验结果显示. 在选择 RNN 和 FNN 的参数时, 输入层单元与特征维度相同为 26, 在选择隐藏层单元数目时, 我们尝试 {5,10,20} 作为隐藏层单元数目. 随隐藏层单元数目增加, 为了使模型收敛, 训练次数也相应变化为 {20,30,50}. 输出层为两个单元, 采用多任务训练方式来同时解决链接值预测和事件结果预测.

对应参数的结果如表 2 所示, 表中 “Hidden” 表示隐藏层单元数目, “Epoch” 为训练次数, “Link” 为链接值预测的正确率, “Event” 为事件结果预测的正确率, 为了便于观察, 统计 “Average” 为两项任务正确率的平均值. 表中可以看出, 在合理参数搭配下, RNN 受隐藏层节点数目影响不如 FNN 明显, 在隐藏单元数目为 10 时, RNN 取得最好表现. FNN 在隐藏层节点由 5 增加到 10 时提升明显. 由 10 到 20 时并无太明显提升, 但后者需要数倍的计算开销. 综合考虑, 我们在结果展示时, RNN 和 FNN 都是用 10 个隐藏层节点和训练 30 次的设置.

5.3 结果和分析

实验的结果如表 3, 第 1 列表示所用的方法名称, FNN (single) 表示这两行结果是使用 FNN 模型, 采用单任务学习方法训练. 同样 RNN (multiple) 表示这两行结果是使用 RNN 模型, 采用多任务学习方法训练. 第 2 列表示对应处理的任务, Link 表示对于序列行为中链接值的预测任务, Event 表示对包含该序列行为事件结果的预测任务. 在单任务 (single) 方法中, 每种任务单独学习一个模型. 在多任务 (multiple) 方法中, 同时使用两个任务标签来学习一个模型. 后面 4 列分别为评价指标, 包括全部测试样本的正确率 (accuracy), 正标签样本的 F1 值 (positive F1), 负标签样本的 F1 值 (negative F1), 以及 ROC 曲线下面积 (AUC). 因为样本中只包含两种样本, 所以这里只给出正标签样本的 AUC 值. 因为理论上在二分类问题中, 负标签样本的 AUC 值与正标签样本相同, 实际上因为计算面积方法偏差, 会有少许不同. 因为 SVM 只能进行二分类, 多分类 SVM 也是通过训练多个 SVM 实现的^[35], 无法只使用一个 SVM 同时完成链接值和事件结果的预测, 所以 SVM 只有单任务学习的实验结果.

表 3 实验结果
Table 3 Experiment results

Method	Task	Accuracy (%)	Positive F1	Negative F1	AUC
SVM	Link	72.60	0.7593	0.6819	0.8268
(single)	Event	86.89	0.8863	0.8451	0.9191
FNN	Link	69.48	0.7295	0.6498	0.7957
(single)	Event	81.30	0.8472	0.7592	0.9050
FNN	Link	75.61	0.7944	0.7003	0.8159
(multiple)	Event	83.74	0.8565	0.8124	0.8917
RNN	Link	77.60	0.8188	0.7070	0.8436
(single)	Event	85.28	0.8664	0.8363	0.9420
RNN	Link	78.04	0.8270	0.6997	0.8497
(multiple)	Event	89.76	0.9136	0.8743	0.9604

首先我们来比较单任务和多任务学习的结果, 在表 3 中, 无论是 FNN 还是 RNN 在多任务学习时的性能表现都优于单任务学习. 虽然单任务学习要训练两个模型来对两个任务进行分别解决, 表面上使用了更多的网络结构. 但在两个任务内部存在较大联系时, 多任务学习能更好地对模型进行调整, 从而只使用一个模型来更好地同时解决这两个相关问题. 具体分析, 在 FNN 模型下, 多任务学习较大地提升了链接值预测任务的性能. FNN 在缺少序列信息的情况下, 可以利用事件的结果来更好地调整自身连接的权重, 以帮助链接值的预测. 在 RNN 模型中, 情况正好相反, 因为能充分地利用序列行为信息, 每次单独行为的结果能得以累积, 从而能更好地预测事件最终的结果.

接下来分析一下 SVM 对比 FNN 和 RNN 的结果, 在单任务情况下, SVM 性能明显优于 FNN, 但是整体表现逊于 RNN. 这说明在静态离散的预测链接值时, SVM 的分类能力要强于 FNN. 特别是在预测事件结果时, SVM 的性能略优于 RNN, 说明该模型拥有很强的分类能力. 但是在预测链接值时, 考虑序列信息的 RNN 就明显优于 SVM. 而 FNN 和 RNN 可以进行多任务学习, 通过对相关联任务的同时学习来提升模型性能. 多任务下的 FNN 与 SVM 的性能相当, 这也证明了上一段所说明多任务学习的优点. 多任务学习的 RNN 性能明显优于 SVM, 更加体现了将事件作为序列行为进行建模处理的优点.

再来比较 FNN 和 RNN 方法的结果, 在处理序列行为时, 单任务学习的 RNN 性能就优于多任务学习的 FNN. 这是因为序列行为事件中, 后面用户进行的行为可能会受到前面用户的影响. 比如在管理员选举时, 投票的用户可以看到前面用户所投的是支持票还是反对票. 该用户不可避免的会参考前面用户的行为, 从而影响到自己的投票结果. RNN 的递归连接, 能很好地对前面用户的行为进行累积. 与该用户的样本特征同时作为隐藏层的输入, 从而综合考虑了用户自身状态和前面用户的投票情况, 达到了更好的预测性能. 同时这种用户行为的累积, 能更好地推断出事件的最终结果.

对比正、负两种样本的 F1 值时我们发现, 三种模型对于正极性样本的预测性能高于负极性样本. 这种不平衡与表 1 中所示的样本标签不平衡有关. 选举结果正、负样本基本持平, 而投票正、负样本标签分布差异较大. 这可能会导致模型偏向于将样本预测为标签多的那一类, 即正极性样本. 这是因为哪怕将全部样本预测为标签较多的那一类, 模型也不会有低于 50% 的准确率. 在标签分布极度不平衡时, 这种现象会更加严重. 而且测试集中正极性样本比例较大的低于训练集样本, 又进一步加剧了这种不平衡在测试集中的体现, 可以通过调节阈值来平衡正负样本的 F1 值表现. 这时不用阈值的 AUC

评价指标的优点就体现出来了, 因为它衡量的是模型将某种样本识别正确的能力. AUC 显示 RNN 的方法能更好地应对标签不平衡情况, 将正负样本区分开.

6 结论

本文以社会网络中可以表示为“序列行为”的事件为主要研究对象, 提出了对序列行为的链接值及事件结果的预测方法. 首先通过对链接网络结构的分析, 本文从链接网络中抽取出了能有效表示链接值的特征. 其次通过对序列行为产生过程的理解, 本文设计并实现了对序列行为进行表示和建模的方法. 该方法能很好地利用序列行为的顺序信息, 并在模型中将其累积. 最后基于递归神经网络模型, 利用这些累积信息, 本文提出了链接值和事件结果的预测方法. 该方法能很好地预测下一个或多个时刻序列中的链接极性值. 同时在完成对整个序列链接值的预测后, 可以推断出包含该序列事件的最终结果. 实验结果表明, 相比较于静态、孤立的预测方法, 本文的方法能更准确的预测序列行为中的链接值和事件的结果.

我们接下来的研究将关注于能否将更多的问题采用序列行为的模型来表示, 以更好地解决这些问题. 另一方面也将进一步去研究对于序列行为建模的方法, 以提升预测性能.

致谢 我们特别感谢王宝勋博士和王鑫博士研究生对于本文研究提出的建设性意见.

参考文献

- 1 Willinger W, Rejaie R, Torkjazi M, et al. Research on online social networks: time to face the real challenges. *ACM SIGMETRICS Perform Eval Rev*, 2010, 37: 49–54
- 2 Burke M, Kraut R. Mopping up: modeling wikipedia promotion decisions. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. New York: ACM, 2008. 27–36
- 3 Getoor L. Link mining: a new data mining challenge. *ACM SIGKDD Explor Newslett*, 2003, 5: 84–89
- 4 Getoor L, Diehl C P. Link mining: a survey. *ACM SIGKDD Explor Newslett*, 2005, 7: 3–12
- 5 Demailly C. The Business Impacts of Social Networking. *AT&T White Papers*, 2008. 1–8
- 6 Yin D, Hong L J, Davison B D. Structural link analysis and prediction in microblogs. In: *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. New York: ACM, 2011. 1163–1168
- 7 Artz D, Gil Y. A survey of trust in computer science and the semantic web. *Web Semantics: Science, Services Agents World Wide Web*, 2007, 5: 58–71
- 8 Liben-Nowell D, Kleinberg J. The link-prediction problem for social networks. *J American Society Inf Sci Tech*, 2007, 58: 1019–1031
- 9 Richardson M, Agrawal R, Domingos P. Trust management for the semantic web. In: *Proceedings of the Semantic Web-ISWC*. Berlin: Springer, 2003. 351–368
- 10 Dong Y, Ke Q, Wu B. Link prediction based on node similarity. *Comput Sci*, 2011, 38: 162–164
- 11 Dong L, Li Y, Yin H, et al. The algorithm of link prediction on social network. *Math Problems Eng*, 2013, 1: 1–7
- 12 Guo J, Dai J, Ma X, et al. Time series link prediction for communication social network. *J Frontiers Comput Sci Tech*, 2010, 4: 552–559
- 13 Tang J, Chang S, Aggarwal C, et al. Negative link prediction in social media. In: *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*. New York: ACM, 2015. 87–96
- 14 Guha R, Kumar R, Raghavan P, et al. Propagation of trust and distrust. In: *Proceedings of the 13th International Conference on World Wide Web*. New York: ACM, 2004. 403–412
- 15 Massa P, Avesani P. Controversial users demand local trust metrics: an experimental study on epinions.com community. In: *Proceedings of the National Conference on Artificial Intelligence*. Palo Alto: AAAI Press, 2005. 121–126

- 16 Brzozowski M J, Hogg T, Szabo G. Friends and foes: ideological social networking. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New York: ACM, 2008. 817–820
- 17 Kunegis J, Lommatzsch A, Bauckhage C. The slashdot zoo: mining a social network with negative edges. In: Proceedings of the 18th International Conference on World Wide Web. New York: ACM, 2009. 741–750
- 18 Leskovec J, Huttenlocher D, Kleinberg J. Predicting positive and negative links in online social networks. In: Proceedings of the 19th International Conference on World Wide Web. New York: ACM, 2010. 641–650
- 19 Liu F, Liu B, Wang X, et al. Features for link prediction in social networks: a comprehensive study. In: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics. Piscataway: IEEE, 2012. 1706–1711
- 20 Liu F, Liu B, Sun C, et al. Deep belief network-based approaches for link prediction in signed social networks. *Entropy*, 2015, 17: 2140–2169
- 21 Elman J L. Finding structure in time. *Cognitive Sci*, 1990, 14: 179–211
- 22 Connor J T, Martin R D, Atlas L E. Recurrent neural networks and robust time series prediction. *IEEE Trans Neural Netw*, 1994, 5: 240–254
- 23 Graves A, Fernández S, Gomez F, et al. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning. New York: ACM, 2006. 369–376
- 24 Graves A, Schmidhuber J. Offline handwriting recognition with multidimensional recurrent neural networks. In: Proceedings of Advances in Neural Information Processing Systems. Red Hook: Curran Associates Inc, 2009. 545–552
- 25 De Mulder W, Bethard S, Moens M F. A survey on the application of recurrent neural networks to statistical language modeling. *Comput Speech Language*, 2015, 30: 61–98
- 26 Wang X, Liu Y, Sun C, et al. Predicting polarities of tweets by composing word embeddings with long short-term memory. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. New York: ACM, 2015. 1343–1353
- 27 Williams R J, Zipser D. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput*, 1989, 1: 270–280
- 28 Werbos P J. Backpropagation through time: what it does and how to do it. *Proc IEEE*, 1990, 78: 1550–1560
- 29 Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput*, 1997, 9: 1735–1780
- 30 Nowak S, Lukashovich H, Dunker P, et al. Performance measures for multi-label evaluation: a case study in the area of image classification. In: Proceedings of the International Conference on Multimedia Information Retrieval. New York: ACM, 2010. 35–44
- 31 Fawcett T. An introduction to ROC analysis. *Pattern Recogn Lett*, 2006, 27: 861–874
- 32 Wang P, Xu B W, Wu Y R, et al. Link prediction in social networks: the state-of-the-art. *Sci China Inf Sci*, 2015, 58: 011101
- 33 Fan R E, Chang K W, Hsieh C J, et al. LIBLINEAR: a library for large linear classification. *J Mach Learn Res*, 2008, 9: 1871–1874
- 34 Schaul T, Bayer J, Wierstra D, et al. PyBrain. *J Mach Learn Res*, 2010, 11: 743–746
- 35 Burges C J. A tutorial on support vector machines for pattern recognition. *Data Min Knowl Disc*, 1998, 2: 121–167

Link value and event-result prediction for sequence behavior in social networks

LIU BingQuan^{1*}, LIU Feng¹, WANG XiaoLong^{1,2}, SUN ChengJie¹ & LIU Ming¹

¹ School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China;

² Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, China

*E-mail: liubq@insun.hit.edu.cn

Abstract With the development of on-line social networks, increasing numbers of functions are provided to

social members. These social members can join different kinds of activities that are presented as events. Some events may include many user-interaction behaviors, and the results of such events are determined by the kinds of behaviors. If each behavior is presented as a link, the sign value of the link may represent the kind of behavior. Link prediction is the focus for predicting the value of these links. Many studies have been performed on solving link-prediction problems. However, most of them were performed in static network environments, and each link was predicted independent of the others. Those researchers treated user behavior for such events as “normal behavior,” which were treated individually and discretely. Therefore, the correlation of those links that were created during the event was not considered. In this paper, we model these event behaviors as “sequence behavior”, and proposed sequence-behavior-based link-prediction and event-prediction methods. In our method, we can accumulate the sequence information of behaviors, and can predict the next one or several link values in the sequence. When we finish predicting the entire sequence, the event’s result may be inferred. The experiment results show that our method outperforms the static and individual link-prediction methods. The event’s result may be better predicted at the same time.

Keywords sequence behavior, link prediction, event-result prediction, sequence modeling, social computing

LIU BingQuan was born in 1970. He received the Ph.D. degree in computer application from the HIT (Harbin Institute of Technology) in 2003. Currently, he is an Associate Professor at HIT. His research interests include question answering, natural language processing, and social computing.

LIU Feng was born in 1983. He is a Ph.D. candidate in computer application from the Harbin Institute of Technology. His research interests include link prediction, social computing, and deep learning approaches.

WANG XiaoLong was born in 1955. He received the Ph.D. degree in computer application from HIT (Harbin Institute of Technology) in 1989. Currently, he is a professor at HIT. His research interests include intelligent input method, online finance information platform, question answering, and artificial intelligence.

SUN ChengJie was born in 1980. He received the Ph.D. degree in computer application from HIT (Harbin Institute of Technology) in 2009. Currently, he is an Associate Professor at HIT. His research interests include recommending systems and massive open online courses.