Relationship Profiling over Social Networks: Reverse Smoothness from Similarity to Closeness [Supplementary Materials]

Carl Yang Kevin Chang

University of Illinois, Urbana Champaign
201 N Goodwin Ave, Urbana, Illinois 61801, USA
{jiyang3, kcchang}@illinois.edu

Appendix A: Detailed Motivations and Concepts

In real-world networks, while links should bear different relationships, they are not explicitly labeled. We argue that being connected in a network does not mean being equally close in reality, and being close does not mean being equally close in every perspective. Since important relationships in social networks are usually discriminatively related with some particular attributes captured by the networks, we propose to leverage user attributes to decipher the hidden relationship semantics of uniform links.

Challenges. The challenges of ARP lie in the effective modeling of homophily– to be systematic and complete as well as robust.. The former is difficult due to the lack of a principled way to infer relationships from attributes, and the later is hard because of incomplete and ambiguous information in real social networks.

Principle: Reverse Smoothness. We notice that there is a systematic connection between attributes and relationships as we desire, which has been explored by the principled framework of graph-based semi-supervised learning (GSSL) [11, 12]. Specifically, GSSL models two proximities on the graph: *closeness* and *similarity*. Consider GSSL in the social network setting. For each user attribute $\mathcal{A}$, an *affinity graph* $\mathcal{R}$ is used to encode user closeness in terms of the corresponding relationship. Then the value of every user on $\mathcal{A}$ can be learned based on $\mathcal{R}$.

As an example, consider v_1, v_5 and v_6 in Figure 1(a) of the main paper. GSSL assumes that closeness in $\mathcal{R}$ is already given. Therefore, if w_{16} is larger than w_{15}, the unknown education attribute of v_1 will be more likely to be predicted as a_6 (UIUC) than a_5 (Stanford), due to the following principle of GSSL.

Principle 1. *(Smoothness Principle) If two nodes v_i and v_j are close on the affinity graph R, their attributes a_i and a_j should be similar [11, 12].*

The focus of GSSL is thus on attribute inference, which goes from closeness to similarity on the graph.

Interestingly, the focus of ARP is the opposite of GSSL, *i.e.*, from similarity to closeness. In ARP, *e.g.*, we only know there is an edge e_{13} between v_1 and v_3. We are interested in the closeness on e_{13} in terms of schoolmate and colleague.

Inspired by GSSL, we intuitively reverse the smoothness principle into the following, which allows us to learn $\mathcal{R}$ by systematically enforcing closeness based on similarity, leading to a novel and unique solution to the ARP problem.

Principle 2. *(Reverse Smoothness Principle) If two users v_i and v_j share similar attributes on $\mathcal{A}$, they should be close on the social affinity graph in terms of $\mathcal{R}$.*

Based on this principle, it is intuitive to implement homophily by probabilistically estimating the closeness on every link in terms of each relationship $\mathcal{R}$ based on the similarity of its related attributes $\mathcal{A}$. The resulting social affinity graphs naturally encode the systematic and complete relationship semantics in the network.

Approach: Holistic Closeness Modeling. Our situation in the real-world graph setting is more complex than that of GSSL. While GSSL can enumerate all pair-wise closeness on each edge and enforce similarity accordingly, the opposite is hard to do in social networks with missing and ambiguous information.

Firstly, attributes are incomplete. Consider v_1 and v_9 in Figure 1 of the main paper. Since the education attribute a_1 and a_9 are missing, we have no idea how similar they are, and thus how close e_{19} should imply in terms of schoolmate. Moreover, even if attributes are complete, closeness cannot be simply enforced on every edge, because similarity can be ambiguous. This is due to the direction of inference, *i.e.*, friends of relationship $\mathcal{R}$ must share the same related attribute $\mathcal{A}$,

Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited

while similar in $\mathcal{A}$ does not necessarily mean close in $\mathcal{R}$. *E.g.*, consider v_2 and v_3 in Figure 1 of the main paper. If v_2 and v_3 are schoolmates, they must share the same education attribute such as UIUC. However, sharing the same education attribute does not necessarily imply the relationship of schoolmates. In fact, they may be colleagues, because they also share the same employer attribute of Google, or both. If we simply enforce closeness on e_{23}, the results will be ambiguous.

To further leverage our reverse smoothness principle and robustly learn the social affinity graph $\mathcal{S}$, we propose to put smoothness constraints and closeness measures onto the whole graph, rather than limiting them to direct edges. Specifically, we define a *path* to be a sequence of non-repeating edges connecting two nodes and use *reachability* to measure closeness as a sum of all weighted paths between two nodes. Then we constrain closeness measured by reachability according to attribute similarity. The intuition is that, the more similar attributes v_i and v_j share, the closer they should be on the graph, and therefore the more paths of shorter lengths and larger weights should connect them.

Continue our example in Figure 1 of the main paper. Inferring r_{19} in terms of schoolmate is challenging due to missing *education* attributes of v_1 and v_9. However, similarity between v_7 and v_8 can be used to estimate the closeness on path $v_7 - v_1 - v_9 - v_8$, which indirectly estimates the closeness on e_{19}. As a result, e_{19} is likely to carry relationship schoolmate, basically because v_1 and v_9 share many friends from UIUC such as v_7 and v_8.

Moreover, continue the discussion from Figure 1 of the main paper about v_2 and v_3, where e_{23} is ambiguous. If we combine closeness measured by multiple paths containing e_{23}, we will end up with a higher probability of e_{23} to be formed due to Google rather than UIUC, mainly because of the short path $v_4 - v_3 - v_2 - v_5$ containing e_{23} between v_4 and v_5 with Google.

By constraining closeness measured with reachability on paths, we effectively utilize the constraints between each pair of nodes v_i and v_j with meaningful attributes onto all edges along the paths connecting v_i and v_j, much beyond their direct edges, if any. Among those edges, many are likely to connect nodes without meaningful values of particular attributes, but in this way, they can still get properly constrained and thus well estimated. Moreover, since each edge can be a component of multiple constrained paths, multiple signals from nearby nodes are combined to disambiguate the semantics of that single edge, yielding much more robust results.

Appendix B: Details of our Algorithms

Efficiency. In practice, $\mathcal{A}$ is usually very sparse, because for every specific attribute, there are numerous distinct values and many users do not have any meaningful values. Therefore, most $f(a_i, a_j)$ computed on $\mathcal{A}$ will be 0 or very small as can be ignored, which means we only need to consider a very small number of pairs of nodes compared to $|\mathcal{V}|^2$.

We develop the Sensor Propagating with Path Indexing (SPPI) algorithm for optimizing our model. SPPI is based on the idea of label propagation on graphs [5]. But instead of labels, we generate a sensor at a starting node s and propagate it to every possible direction at each step, and index the edges it goes through. We call it a sensor because each time it touches a new node t, it detects whether a legal path from s to t has been detected. If yes, we can retrieve the exact edges on that path by looking up the indexes. Another important task of the sensor is to act as a marker, so at each step of propagation, we only consider a small subset of $\mathcal{V}$ with the sensors on.

We formally present the procedures of SPPI in Algorithm 1. To implement our path indexer D, we borrow the idea of *path descriptor* from [7]. A path descriptor $d(\cdot)$ is a pair of bit vectors for each path l as $d(l) = \{V, E\}$. V is a bit vector of $|\mathcal{V}|$ items with $V_i = 1$ if node $v_i \in l$ and $V_i = 0$ otherwise. E is a bit vector of $|\mathcal{E}|$ items with $E_i = 1$ if edge $e_i \in l$ and $E_i = 0$ otherwise. For v_i and v_j, we use $D(i, j)$ as a list of path descriptors to index all paths connecting them. With the path descriptor d and index matrix D, it is easy to encode, record and retrieve multiple paths connecting any two nodes with time complexity $O(|\mathcal{E}|)$. It is also efficient to check if a certain node or edge is on a path and if two paths are the same by simply doing binary AND and XOR on d.

SPPI is different to exhaustive search in several ways. Firstly, it is based on breath-first search (BFS), so that as shorter paths contribute more in our model, we can tune K to avoid the consideration of longer paths and save computation time. Secondly, it is more efficient than brute force BFS by utilizing the sensor vector and path descriptor lists and checking the legitimacy of propagation directions at each step, avoiding repeated iterations when considering the same nodes in different levels of search. Thirdly, it is specially suited to the optimization of our model, because after each execution, it finds paths from a starting node to all nodes with similar attributes on the graph, while avoiding enumerating paths connecting other nodes.

Correctness. We evaluate the correctness of SPPI by checking the completeness and non-repetitiveness of the algorithm. In *Step 8*, by requiring $v_j \notin l$, we require that the next node to be propagated to is not already in the path being considered, so no loopy paths can be

Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited

generated; by requiring $l + e_{ij} \notin D(I, j)$, we guarantee that each path is generated only once. Moreover, in *Step 10* and *Step 14*, we ensure that all and only nodes with the newly propagated sensors on are considered at each step, so the same nodes are not considered multiple times in different search levels. Finally, in *Step 6*, since we always try to propagate the sensors through every possible edge, we make sure that every simple path is considered.

Since path indexing and legitimacy checking are efficiently $O(1)$ with the path descriptor lists, the overall computational complexity of finding paths is $O(K|\mathcal{V}|^2)$. However, the actual computational time is much shorter than $K|\mathcal{V}|^2$. In each step of BFS, the numbers of considered nodes and neighbors are much less than $|\mathcal{V}|$. The efficiency of finding paths can be further improved by path caching and reusing, to fully utilize the path indexes. Specifically, we try to cache as many legitimate paths as possible after they are indexed. Therefore, the paths of length K can be directly reused when considering paths longer than K. As the number of paths goes exponentially with the length of the paths, it is usually impossible to keep all path indexes in cache and even memory. Motivated by the scale-free property of social networks [2], which leads to the frequent reuse of paths between a small number of high-degree hub nodes, we adopt the Least Recently Used (LRU) algorithm for path caching.

Algorithm 1 SPPI algorithm

procedure SPPI ▷ Input

$\mathcal{G}(\mathcal{V}, \mathcal{E})$: the graph.

K: the maximum length of paths we consider.

I: the source node from which we want to find all paths to other nodes in the graph.

▷ Output

$D(I, j), j = 1 \dots |\mathcal{V}|$: each $D(I, j)$ is a list of path descriptors describing paths between v_I and v_j.

▷ Variables we use

Φ: a bit vector of length $|\mathcal{V}|$, where $\Phi(i)$ marks if there is a sensor on node v_i.

▷ Initialize

$\Phi(I) \leftarrow 1, \forall j \neq I, \Phi(j) \leftarrow 0$

$\forall j, D(I, j) \leftarrow null$

▷ Iteration

for $k = 1 : K$ **do**

  for all i in $1 : |\mathcal{V}|$ with $\Phi(i) == 1$ **do**

    for all j in $1 : |\mathcal{V}|$ with $e_{ij} == 1$ **do**

      for each path l in $D(I, i)$ **do**

        if $v_j \notin l$ && $l + e_{ij} \notin D(I, j)$ **then**

          $D(I, j) = D(I, j) + (l + e_{ij})$

          $\Phi(j) \leftarrow 1$

        end if

      end for

    end for

    $\Phi(i) \leftarrow 0$

  end for

end for

return $D(I, j), j = 1 \dots |\mathcal{V}|$

end procedure

Appendix C: More Related Works

Although we are the first to formally define the problem of ARP, since relationship semantics is critical for various tasks on social networks, algorithms in recent literature have already been intensively solving the related problems to ours. However, while they commonly believe in homophily and connect attributes and relationships with it, they do not correctly interpret the nature of homophily as complete and systematic. According to their main objectives, they can be categorized into two groups. The first group applies homophily to learn attributes through relationships, assuming that relationships on each link are mutually exclusive [1, 3, 8]. While they implicitly learn relationships, they do not compute the complete semantics on each link. The second group utilizes homophily to detect communities, assuming that each community of nodes are connected through the same relationships [4, 6, 9, 10]. They compute the semantics of communities, rather than the systematic semantics on links.

The first group of algorithms can produce systematic but not complete relationship semantics. Since attribute learning aims to infer the missing attributes of every node, systematic relationship semantics can usually be retrieved afterwards by looking at the inferred attributes of nodes on each side of a link. The recent work EdgeExplain [1] is the closest to ours, which optimizes relationships jointly with attributes. However, it assumes that each link should only carry one relationship. The discriminative relational learning [8] exploits community features as latent social dimensions to aid attribute classification. Therefore, each link is only understood through one attribute chosen by the classification method applied on the two linked nodes. The Co-Profiling [3] algorithm attempts to learn both user attributes and circles via searching for the reasons of link formation. Each clink is then understood through one reason within one of the non-overlapping circles it detects. Considering two users that are both colleagues and schoolmates, those algorithms force the result to be either of them, which is partial and does not always reflect the truth. In contrast, ARP will yield two close probabilities w.r.t. the two relationships.

Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited

The second group of algorithms can produce complete but not systematic relationship semantics. As they evolve, many community detection algorithms nowadays attempt to characterize communities through attributes. Examples include generative models like CESNA [9] and Circles [4] and other frameworks like PCL-DC [10] and CODICIL [6]. They all explicitly model the node attributes that cause communities to form and compute a weight matrix characterizing communities w.r.t. attributes. Relationship semantics can then be generated by assuming that links within the same communities carry the same relationships. Therefore, multiple relationships can be associated on each link, if the two linked nodes belong to multiple overlapping communities. However, since they only compute the community semantics, the relationship semantics computed from their results are coarse. To be more specific, there is no way to understand every link, such as those between different communities and outside of any communities. Moreover, they fail to distinguish individual links within the same community. Unlike them, ARP aims to profile relationships in a finer granularity. Rather than relying on the detection of communities, it utilizes the local paths to precisely understand every link as long as a path goes through it.

Appendix D: Case Studies on DBLP

One advantage of ARP over the compared algorithms is that it can estimate the probability of each connection to carry each relationship. On LEN and FEN, we convert the probabilities into binary outputs in order to present quantitive comparisons with the baselines. However, the application of ARP is much broader than binary predictions. We use DBLP to present some insightful results derived from the relationship probabilities, which only ARP can generate.

Consider some interesting novel applications on DBLP. One of them is to find out people's closest co-authors within different research fields. *E.g.*, two authors might study similar problems in data mining, but very different problems in database. Thus, how can we identify people's closest co-authors given a specific field? Another interesting application is to identify the closest pairs of authors within each field of study, *i.e.*, who study the closest problems and collaborate most in a specific field? Considering specific relationships, such problems are novel and naturally different from general graph ranking.

We show that problems like these are direct applications of ARP. By modeling publication venues as user attributes and co-authorship as user connections, ARP accurately computes the closeness among authors *w.r.t.* different fields.

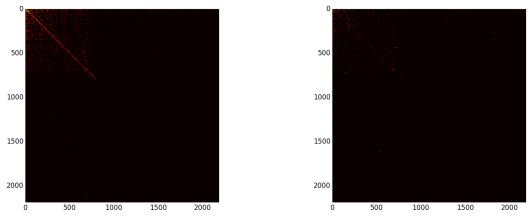
Consider three representative venues that correspond to three different but related fields. Table 1 shows the relationship specific closeness learned by ARP and normalized into multinomial distributions over each pair of authors. While relationships can be multiple and vary across connections, ARP completely retrieves them in all aspects. As we identify the authors, we observe that the relationships between some specific pairs of authors are quite interesting, *e.g.*, Jiawei Han and Xiaolei Li. We see that they are very close in VLDB, a bit close in KDD, but far away in ICML. To interpret the results, we look into the data and find that among the three conferences considered, the two authors have direct collaborations only in VLDB, which explains for the corresponding strong closeness. Nevertheless, they both directly work with authors like Dong Xin and Hong Cheng, who have collaborations in KDD. While the truth is that some of the papers that Jiawei Han and Xiaolei Li co-authored in VLDB are quite related to data mining (KDD), such semantics is hidden but effectively retrieved through paths connecting Dong Xin and Hong Cheng and then annotated to the relationship between Jiawei Han and Xiaolei Li by ARP.

Authors	KDD	VLDB	ICML
Jiawei Han, Philip S. Yu	**0.65**	*0.35*	0
Jiawei Han, Xiaolei Li	*0.04*	**0.96**	0
Jiawei Han, Tianbao Yang	*0.17*	0	**0.83**
Christos Faloutsos, Hanghang Tong	**0.86**	*0.13*	0.01
Divesh Srivastava, H. V. Jagadish	*0.03*	**0.97**	0
Corinna Cortes, Mehryar Mohri	*0.14*	0	**0.86**

Table 1: **Multi-aspect author relationships.**

Authors	(A) Holistic	(B) Single
Jiawei Han, Chi Wang	1.00/1st	1.00/1st
Christos Faloutsos, Hanghang Tong	0.95/2nd	1.00/1st
Hynne Hsu, Mong-Li Lee	0.93/3rd	0.72/10th
Jiawei Han, Philip S. Yu	0.90/4th	0.63/18th
Christos Faloutsos, Lei Li	0.85/5th	0.72/10th

Table 2: **Rank list of closest authors on KDD.**

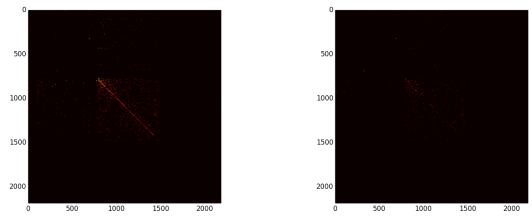


(a) By holistic path model (b) By single edge model

Figure 1: **Collaborations in KDD are highlighted.**

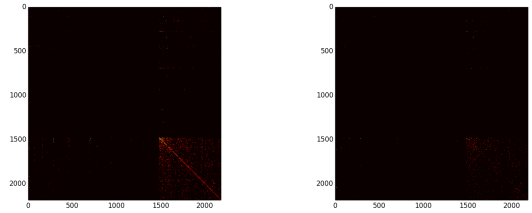
In Table 2, pairs of authors are ranked with their relative closeness in the research field of data mining *w.r.t.* the KDD conference. Column (A) shows the results of holistic modeling, where we set $K = 3$ and $\alpha = 0.8$ to consider indirect collaborations. The results are intuitive because the top ranked pairs of authors

Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited



(a) By holistic path model (b) By single edge model

Figure 2: **Collaborations in ICML are highlighted.**



(a) By holistic path model (b) By single edge model

Figure 3: **Collaborations in VLDB are highlighted.**

are indeed those who collaborate most in the field. To show that the results in Column (A) are non-trivial as cannot be simply computed by counting the number of collaborated papers, we also provide in Column (B) the results without holistic modeling, which are less intuitive. *E.g.*, although Han and Yu work quite closely on data mining, the closeness between them decreases from 0.90 to 0.63 and their rank drops from $4th$ to $18th$, merely because their many indirect collaborations are ignored. The situations are similar for many other pairs such as Faloutsos and Li.

Next, we demonstrate the power of our holistic closeness model by visualizing the computed collaboration matrices in different fields in comparison with those computed by the single edge model. In Figure 1-3, the x and y axis are the same set of researchers. As we can see, the collaboration matrices are naturally sparse. While the three relationships cluster on different locations (thus among different set of authors), some collaborations indeed bare multiple relationships (the same red dots appear in multiple figures). Also, it is interesting to see that in the KDD plot, the red dots scatter quite strictly in the top-left corner, which means that only data mining people collaborate in data mining a lot. However, in both the ICML plot and the VLDB plot, while most red dots scatter in the corresponding locations (the center-center and bottom-right), some obviously appear in the ICML-KDD and VLDB-KDD locations. It clearly indicates that many data mining people also collaborate in machine learning and database. It is true because data mining people need to know both machine learning and database, but not vise versa. Moreover, in both Figure 2 and 3, there is almost no red dot in the VLDB-ICML squares, which means database people seldom collaborate in machine learning conferences and vise versa.

In the collaboration matrices computed by single edge models, the same insightful observations are much harder to make. The red dots are much sparser, especially in the cross-field locations. *E.g.*, in the Figure 2(b), we can hardly observe the collaborations in machine learning between data mining researchers and machine learning researchers, as we can observe clearly in the Figure 2(a).

Due to space limit, for clearer views of the plots and more interesting results, please visit our Github project[1].

References

[1] D. Chakrabarti, S. Funiak, J. Chang, and S. Macskassy, *Joint inference of multiple label types in large networks*, in ICML, 2014, pp. 874–882.

[2] K. Choromański, M. Matuszak, and J. Mi?kisz, *Scale-free graph with preferential attachment and evolving internal vertex structure*, Journal of Statistical Physics, 151 (2013), pp. 1175–1183.

[3] R. Li, C. Wang, and K. C.-C. Chang, *User profiling in an ego network: co-profiling attributes and relationships*, in WWW, 2014, pp. 819–830.

[4] J. J. Mcauley and J. Leskovec, *Learning to discover social circles in ego networks*, in NIPS, 2012, pp. 539–547.

[5] U. N. Raghavan, R. Albert, and S. Kumara, *Near linear time algorithm to detect community structures in large-scale networks*, Physics Review E, 76 (2007), p. 036106.

[6] Y. Ruan, D. Fuhry, and S. Parthasarathy, *Efficient community detection in large networks using content and links*, in WWW, 2013, pp. 1089–1098.

[7] F. Rubin, *Enumerating all simple paths in a graph*, ITCS, 25 (1978), pp. 641–642.

[8] L. Tang and H. Liu, *Relational learning via latent social dimensions*, in KDD, ACM, 2009, pp. 817–826.

[9] J. Yang, J. McAuley, and J. Leskovec, *Community detection in networks with node attributes*, in ICDM, 2013, pp. 1151–1156.

[10] T. Yang, R. Jin, Y. Chi, and S. Zhu, *Combining link and content for community detection: a discriminative approach*, in KDD, 2009, pp. 927–936.

[11] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, *Learning with local and global consistency*, NIPS, 16 (2004), pp. 321–328.

[12] X. Zhu, Z. Ghahramani, J. Lafferty, et al., *Semi-supervised learning using gaussian fields and harmonic functions*, in ICML, vol. 3, 2003, pp. 912–919.

[1] https://github.com/yangji9181/ARP2017

Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited