
Collaborative and Structural Recommendation of Friends using Weblog-based Social Network Analysis

William H. Hsu, Andrew L. King, Martin S. R. Paradesi, Tejaswi Pydimarri, Tim Weninger

Department of Computing and Information Sciences, Kansas State University
234 Nichols Hall, Manhattan, KS 66506

[bhsu | aking | pmsr | tejaswi | weninger}@ksu.edu](mailto:{bhsu|aking|pmsr|tejaswi|weninger}@ksu.edu)

<http://www.kddresearch.org>

Abstract

In this paper, we address the problem of link recommendation in weblogs and similar social networks. First, we present an approach based on collaborative recommendation using the link structure of a social network and content-based recommendation using mutual declared interests. Next, we describe the application of this approach to a small representative subset of a large real-world social network: the user/community network of the blog service *LiveJournal*. We then discuss the ground features available in *LiveJournal*'s public user information pages and describe some graph algorithms for analysis of the social network. These are used to identify candidates, provide ground truth for recommendations, and construct features for learning the concept of a recommended link. Finally, we compare the performance of this machine learning approach to that of the rudimentary recommender system provided by *LiveJournal*.

Keywords: social networks, collaborative recommendation, *LiveJournal*, machine learning, data mining, graph algorithms

1 INTRODUCTION

This paper presents a recommender system for links in a social network. Such links have different meanings depending on the start and end points: between users of a weblog service denote friendship or trust; between users and communities, they denote subscribership and requested privileges such as posting access; between communities and users they denote accepted members and moderator privileges. Specific recommendation targets for weblogs include links (new friends and subscriberships), security levels (link strength), requests for reciprocal links (trust, membership and moderatorship applications), and definition of security levels (filters). There are analogous applications of this functionality in

social networks such as citation and collaboration networks.

We investigate the problem of link recommendation in such weblog-based social networks and describe an annotated graph-based representation for such networks. Our approach uses graph feature analysis to recommend links (u, v) given structural features of individual vertices and joint features of the start and end points of a candidate link, such as distance between them. We present a hybrid system that combines analysis of link structure with analysis of content, such as shared interests. This framework supports more in-depth analyses of structure combined with content, for normalization, feature construction, learning of constraints, clustering of a user's friends and communities. Such capabilities in turn support more sophisticated recommendations such as the security level of new and existing friendships.

In this paper, we describe how this hybrid approach was used to develop *LJMiner*, a recommender system for the popular weblog service *LiveJournal*. *LJMiner* differentiates friends from non-friends in a connected group of users with greater accuracy than the recommender system actually used by *LiveJournal* – namely, ranking users and communities by decreasing count of mutual interests. This task is similar to the friend recommendation task given candidates within a specified radius, so the result is a strong positive indication that *LJMiner* can generate better recommendations than interest-based or simple graph-based recommendation in a fielded application.

2 BACKGROUND

2.1 Social Networks in Weblogs

Social network services such as Denga's *LiveJournal*, Google's *Orkut*, *Friendster*, and *The Facebook* allow users to list interests and link to friends, sometimes annotating these links by designating trust levels or qualitative ratings for selected friends. Among the most popular of these is the weblog service *LiveJournal*, a highly customizable and flexible personal publishing tool

used by several million users. In this work, we focus on *LiveJournal* and derivative services such as *GreatestJournal*, *DeadJournal*, and *JournalFen* based on the same open-source server code. At the time of this writing, there are over 8.5 million *LiveJournal* accounts, of which over 2.5 million are active; these are either user accounts, associated with one or a small number of individuals, or communities (each a forum for multiple users similar to *MSN Communities* or *Yahoo! Groups*). The embeddability, syndication, OpenID integration, and metadata features of *LiveJournal* make it a rich source of structured data about these users and communities, and the interrelationships among them.

Friendship in *LiveJournal* is an asymmetric relation between two accounts and which can be represented as an edge in a directed graph. Either the start vertex u or the end vertex v may denote either a user account or a community account, though community-to-community links are not used. Table 1 lists the categories of links and specific link types. Community-to-user links are of three independent types: “member”, “posting access”, and “maintainer” (post and membership moderation). Of these relations, only membership is requested by users or invited by maintainers; the rest are privileges granted by maintainers.

Table 1. Types of links in the blog service *LiveJournal*.

Start	End	Link Denotes
User	User	Trust or friendship
User	Community	Readership or subscribership
Community	User	Membership, posting access, maintainer
Community	Community	Obsolete

Thus, a reciprocal link between a user and a community means that the user subscribes to the community and is an accepted member of the community. Subscriptions are listed in the “Friends: Communities” section of the user’s page and in a list titled “Watched By” in the community’s page. Links from user u to v are listed in the “Friends” list of u and in an optionally displayed “Friends Of” list of v . This list can be partitioned into reciprocal and non-reciprocal sublists for a user u :

Mutual Friends: $\{ v \mid (v, u) \in E \wedge (u, v) \in E \}$

Also Friend Of: $\{ v \mid (v, u) \in E \wedge (u, v) \notin E \}$

The social network for the *LiveJournal* user base consists of many connected components. There are a few source vertices corresponding to users that link to friends but have no reciprocated friendships. Many of these are aggregator accounts created for reading RSS or other users’ blog entries. Additionally, there are sink vertices corresponding to users or communities watched by others, but who have named no friends. Some of these are

channels for announcement or dissemination of creative work.

2.2 Collaborative, Structural, and Content-Based Link Recommendation

We now discuss the link recommendation problem, the available data, and some previous approaches. One social function of many weblog services is to introduce people to new friends and communities and to provide content aggregators and communication media among people who know each other. The basis for these introductions is often the list of interests reported by a user or community maintainer.

LiveJournal collects all of the abovementioned information on the social network structure, along with user interests, self-reported personal information, and descriptive statistics about posting history in a user information page for each account. We seek to mine this data in order to provide improved link recommendations. Our hypothesis is that recommendations based only on shared interests can be greatly improved using information about the graph structure. For instance, local structural features such as whether a link already exists from the candidate friend to the recommender system user, how many mutual friends of the user and candidate there are, and the degree of user and candidate all provide some supporting evidence for a link recommendation. Additionally, search-based graph analysis can yield information about the shortest alternate path in friendships from the user to the candidate, and vice versa.

The long-term goal of this research is to explore ways in which contextual information can be combined with graph structure or descriptive graph features to obtain an enriched model for making this weblog-based link recommendations. Examples of this information include user interests, preferences and constraints (e.g., desired ranges or limits for number of friends). Mechanisms for combining structural and contextual information include filtering candidate sets by graph proximity, counting number of mutual friends sharing certain interests, normalizing weights of shared interests based on dynamic itemset frequency within a certain graph radius.

Initially, we consider a predominantly collaborative and structural approach to recommendation: we hypothesize that users are likely to prefer links similar to extant ones and therefore generate candidates in this paper from within a specified radius in the social network. This is a form of collaboration in that the paths are formed by other users’ choices of friends. Statistics such as the indegree of a vertex, denoting length of the users “Friends Of” list, are similarly collaborative in nature. We also use counts of mutual interests and mutual friends (structural recommendation). In the next section, we discuss the acquisition of data and experiment design for this recommendation problem.

2.3 Methodologies for link mining

Getoor and Diehl [GD05] recently surveyed techniques for link mining, focusing on statistical relational learning approaches and emphasizing graphical models representations of link structure. Ketkar *et al.* [KHC05] compare data mining techniques over graph-based representations of links to first-order and relational representations and learning techniques that are based upon inductive logic programming (ILP).

Sarkar and Moore [SM05] extend the analysis of social networks into the temporal dimension by modeling change in link structure across discrete time steps, using latent space models and multidimensional scaling. One of the challenges in collecting time series data from *LiveJournal* is the slow rate of data acquisition, just as spatial annotation data (such as that found in LJ maps and the “plot your friends on a map meme”) is relatively incomplete.

2.4 Other applications using graph mining

Popescul and Ungar [PU03] learn a kind of entity-relational model from data in order to predict links. Hill [Hi03] and Bhattacharya and Getoor [BG04] similarly use statistical relational learning from data in order to resolve identity uncertainty, particularly coreferences and other redundancies (also called deduplication). Resig *et al.* [RDHT04] use a large (200000-user) crawl of *LiveJournal* to annotate a social network of instant messaging users, and explore the approach of predicting online times as a function of friends graph degree.

There have been numerous recent applications of social network mining based on the text and headers of e-mail. One notable research project by McCallum *et al.* [MCW05] uses the Enron e-mail corpus and infers roles and topic categories based on link analysis. A primary goal of this work is to extend the graph mining approach beyond link **prediction** and **recommendation** towards link **explanation** and **annotation**.

It may be much more useful to explain why a group of friends in a blog service created accounts *en masse* or added one another as friends than to recommend relationship sets that are already extant or structured according to a preexistent social group. For example, high school classmates often create accounts and encourage their peers to join the same service. In a few cases, this is encouraged or facilitated by a teacher, for a class project. Solving the problem of link prediction is not particularly useful in this case, because the user decisions have already been made or strongly constrained; however, it may be very useful to link other classmates **not** working on the same project to the same relationship set (perhaps they were encouraged to join the blog service by students who continued to use it after the class project).

Large groups such as web comic subscriberships, community co-members, etc. are also somewhat identifiable, and relating members of a blog service to one

another through relationship sets is a typical entity-relational data modeling operation that can be made more robust and efficient through graph feature extraction.

3 EXPERIMENT DESIGN

3.1 *LJCrawler*

To acquire the graph structure and attributes describe in the previous section, we developed an HTTP-based spider called *LJCrawler* to harvest user information from *LiveJournal*. This multithreaded program collects an average of 5 records per second, traversing the social network depth-first and archiving the results in a master index file. Because *LiveJournal*’s functionality for looking up users by user number is only available to administrators, we decided to compile a list of seeds for a disjoint-set representation of the disconnected social network. For purposes of this experiment, however, starting from just one seed (the first author’s *LiveJournal* ID) and restricting the crawl to one connected component was sufficient.

Using *LJCrawler*, we compiled an adjacency list and the following ground features for each user:

- Account type (user, community)
- Paid status (free, paid, permanent)
- Dates of creation and last update
- Interest list

3.2 Feature Analyzers

We define a single example to be a candidate edge (u, v) in the underlying directed graph of the social network, along with a set of descriptive features calculated from the annotated graph recorded by *LJCrawler*:

Graph features:

1. Indegree of u : popularity of the user
2. Indegree of v : popularity of the candidate
3. Outdegree of u : number of other friends besides the candidate; saturation of friends list
4. Outdegree of v : number of existing friends of the candidate besides the user; correlates loosely with likelihood of a reciprocal link
5. Number of mutual friends w such that $u \rightarrow w \wedge w \rightarrow v$
6. “Forward deleted distance“: minimum alternative distance from u to v in the graph without the edge (u, v)
7. Backward distance from v to u in the graph

The degree attributes can be enumerated in time linear in the number of users, as can the mutual friends count for each pair of users. Deleted distance requires one iteration over w for shortest path algorithm (re-relaxation). In a graph (V, E) , backward distance requires $\Theta(|V|^3)$ using the

brute-force dynamic programming implementation produced for this experiment, $\Theta(|E| \lg |E|)$ using a simple heap, and $\Theta(|V| \lg |V| + |E|)$ using Fibonacci heaps. [CLRS02]

Interest-based features:

- 8 Number of mutual interests between u and v
- 9 Number of interests listed by u
- 10 Number of interests listed by v
- 11 Ratio of the number of mutual interests to the number listed by u
- 12 Ratio of the number of mutual interests to the number listed by v

Using a straightforward string pair enumeration and comparison algorithm, the mutual interest counts are stored in matrix of $|V|^2$ elements, each requiring constant time to check (given a maximum of 150 interests).

Other features: Additional planned features for continuing experiments include dates (update frequencies when taken differentially), user options such as maximum friends count, and content descriptors of *LiveJournal* entries and comments (average post length, word frequency, etc.).

3.3 Graph Search Algorithms for Computing Features

Computing the minimum forward and backward distances can be done more efficiently by using breadth-first search. A straightforward implementation that we are currently using performs $\Theta(|V|)$ BFS passes to accumulate the edge set of the entire *LiveJournal* friends graph, which contains multiple connected components. This requires $\Theta(|V|(|V| + |E|))$ time. Currently, a Java implementation of this algorithm requires about 70 seconds to process a 3000-node graph. Since $|E|$ is about 20 times $|V|$ on average (that is, the average outdegree or “Friend Of” cardinality is 20), an algorithm that is more efficient in practice is possible.

We note that the amortized cost of running BFS to precompute all-pairs shortest paths (APSP) with the actual edge deleted (which is necessary to avoid knowing the prediction target in link prediction) is $\Theta(|E|(|V| + |E|))$. This is prohibitively large even for our “mid-sized” subgraphs of 10-50K nodes; when $|V|$ is about 9 million, $|E|$ is about 200 million, enumerating APSP is completely infeasible. However, we do not typically consider all of E , so the bottleneck is typically the first step plus a constant number of calls to BFS, requiring running time in $\Theta(k(|V| + |E|))$.

3.4 Generating Candidates

We considered several alternative ways to generate candidate edges (u, v) :

1. Uniform at random

2. From a query distribution modeled on the frequency of vertices at a given graph distance
3. Exhaustively within a specified radius

The first technique is likely to be unscalable, as the number of candidates is $|V|^2$. The second requires having a representatively large sample of the full *LiveJournal* social network, in order to fit the distribution parameters accurately. The third was the most straightforward to implement. Two calls to the all pairs shortest path algorithm provided cost matrix, and one pass at each radius up to a maximum of 10 yielded the data shown in Table 2. To simplify the initial experiments, we defined the classification problem to be classification of $d(u, v)$ as 1 or 2.

This task is actually useful for social network recommender systems because discrimination of a direct friend from a “friend of a friend” (FOAF) is functionally similar to recommending FOAFs to link to directly. There are more detailed classification targets, such as placement, promotion, and demotion of linked friends within strata of trust (setting, increasing, and decreasing the security level), but choosing a user’s friends to begin with is the more fundamental decision.

Table 2. Number of candidate edges for the 941-node *LiveJournal* graph.

Distance d	Frequency (= d)	Cumulative ($\leq d$)
1	5934	5934
2	45042	50976
3	69013	119989
4	101256	221245
5	87683	308928
6	51040	359968
7	29981	389949
8	13230	403179
9	4808	407987
10	1022	409009

4 RESULTS

4.1 Small Experiment

Using the 941-node annotated graph summarized in Table 2, we generated 50976 candidate edges. Note that all forward distances are greater than 1: when u and v are actually connected, we erase (u, v) and find the length of the shortest alternative path. The complete listing of all twelve features is given in Section 3.

The numerical types of all of the network features – both the ones describing the graph and those measuring and interests and ratios – makes data set amenable to logistic regression.

We defined the concept *IsFriendOf* and trained three types of inducers with:

1. all attributes
2. all graph attributes excluding the forward and backward distances
3. the backward distances alone
4. the backward and forward distances alone
5. interest-related attributes alone.

Table 3. Percent accuracy for predicting all classes using the 941-node graph.

Inducer	All	NoDist	BkDist	Dist	Interest
J48	98.2	94.8	95.8	97.6	88.5
OneR	95.8	92.0	95.8	95.8	88.5
Logistic	91.6	90.9	88.3	88.9	88.4

Table 4. Percent accuracy for predicting edges ($d = 1$, *IsFriendOf* = TRUE) using the 941-node graph.

Inducer	All	NoDist	BkDist	Dist	Interest
J48	89.5	65.7	67.7	83.0	5.4
OneR	67.7	41.1	67.7	67.7	4.5
Logistic	38.3	33.3	0	4.5	4.5

Tables 3 and 4 show the results for three inducers: the *J48* decision tree inducer, the *IR* inducer, and the Logistic regression inducer. All accuracy measures were collected over 10-fold cross-validated runs. The *J48* output with all features achieves a significant boost over the next highest (distance only).

4.2 Data Acquisition and Larger Experiments

The crawler has been improved with several service-specific optimizations for fetching user info pages. Presently these do not use *LiveJournal*'s BML feed of user data, which is incomplete for our purposes (that is, not all ground attributes in our initial relations are provided). At press time, this crawler processes about 20000 user records per hour and thus would require over a week to crawl *LiveJournal*.

The current bottleneck is the $\Theta(|V| (|V| + |E|))$ step described in Section 3.3. This is the dominant term, because the constant k denoting the number of candidate edges is usually much smaller than n , e.g., 100-1000, so that $\Theta(k (|V| + |E|))$ is not only in $\Theta(|V| + |E|)$, but actually just a few hundred times the cost of a single BFS.

4.3 Interpretation

Using mutual interests alone, even with normalization based on the number of interests in u and v , results in very poor prediction accuracy using all inducers with which we experimented. Intermediate results are achieved using mutual friends count and degree (NoDist: 65.7% on predicting edges) and using forward deleted distance and

backward distance (Dist: 67.7%). Using all 12 computed graph and annotation features resulted in the highest prediction accuracy (All: 89.5%).

We note that *LiveJournal* once used a variant of normalized mutual interests to produce a list of potential friends, arranged in decreasing order of match quality. Although this was not the same type of recommender system as *LJMiner* supports, it shows that the state of the art user matching systems have a lot of room for improvement. The results in Table 4 indicate that features produced by *LJMiner*, used with a good inducer, can generate collaborative and structural recommendations.

5 CONTINUING WORK

Scaling up: Our current research focuses on scaling up to tens of thousands and eventually millions of users. Crawling over 9 million records is at least technically feasible, but scaling up the graph analyzers is a challenge that may best be met with heuristic search.

Learning relational models: A promising area of research is the recovery of relational graphical models, including class-level (membership and reference slot) uncertainty. [GFKT02] *LJMiner* has yielded a ready source of semistructured data for both structure learning and distribution learning. Another potentially useful approach is to organize users and communities into clusters using this relational model. We have developed schemas for blog posts (entries, threads, comments) and for users and dynamic groups of users. This is related to previous preliminary work on relational data mining for personalization of web portals, especially computational grid portals. [HBJ03]. Much of the relational metadata in the bioinformatics domain comes from description languages for workflows and workflow components [Hs04]. The next step in our experimental plan is to use schemas such as our detailed ones for blog service users and bioinformatics information and computational grid users [Hs05] to learn a richer predictive model. Finally, modeling relational data as it persists or changes across time is an important challenge.

6 ACKNOWLEDGEMENTS

We thank Todd Easton and Kirsten Hildrum for helpful discussions and Jason Li for assistance with implementations.

7 REFERENCES

[BG04] I. Bhattacharya & L. Getoor. Deduplication and group detection using links. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD) Workshop on Link Analysis and Group Detection (LinkKDD2004)*, Seattle, WA, USA, August 22-25, 2004.

- [CLRS02] T. H. Cormen, C. E. Leiserson, R. L. Rivest, & C. Stein. *Introduction to Algorithms, Second Edition*. Cambridge, MA: MIT Press, 2002.
- [GD05] L. Getoor & C. P. Diehl. Link mining: a survey. *SIGKDD Explorations, Special Issue on Link Mining*, 7(2):3-12.
- [GFKT02] L. Getoor, N. Friedman, D. Koller, & B. Taskar. Learning Probabilistic Models of Link Structure. *Journal of Machine Learning Research*, 2002.
- [HBJ03] W. H. Hsu, P. Boddhireddy, & R. Joehanes. Using probabilistic relational models for collaborative filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Statistical Learning of Relational Models (SRL)*, Acapulco, MEXICO, August, 2003.
- [Hi03] S. Hill. Social network relational vectors for anonymous identity matching. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Statistical Learning of Relational Models (SRL)*, Acapulco, MEXICO, August, 2003.
- [Hs04] W. H. Hsu. Relational graphical models of computational workflows for data mining. In *Proceedings of the International Conference on Semantics of a Networked World: Semantics for Grid Databases (ICSNW-2004)*, p. 309-310, Paris, FRANCE, June, 2004.
- [Hs05] W. H. Hsu. Relational graphical models for collaborative filtering and recommendation of computational workflow components. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Multi-Agent Information Retrieval and Recommender Systems*, Edinburgh, UK, July 31, 2005.
- [KHC05] N. S. Ketkar, L. B. Holder, & D. J. Cook. Comparison of graph-based and logic-based multi-relational data mining. *SIGKDD Explorations, Special Issue on Link Mining*, 7(2):64-71.
- [MCW05] A. McCallum, A. Corrada-Emmanuel, & X. Wang. Topic and role discovery in social networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Edinburgh, UK, August, 2005.
- [MH04] M. Mukherjee & L. B. Holder. Graph-based data mining on social networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD) Workshop on Link Analysis and Group Detection (LinkKDD2004)*, Seattle, WA, USA, August 22-25, 2004.
- [PU03] A. Popescul & L. H. Ungar. Statistical relational learning for link prediction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Statistical Learning of Relational Models (SRL)*, Acapulco, MEXICO, August, 2003.
- [RDHT04] J. Resig, S. Dawara, C. M. Homan, & A. Teredesai. Extracting social networks from instant messaging populations. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD) Workshop on Link Analysis and Group Detection (LinkKDD2004)*, Seattle, WA, USA, August 22-25, 2004.
- [SM05] P. Sarkar & A. Moore. Dynamic social network analysis using latent space models. *SIGKDD Explorations, Special Issue on Link Mining*, 7(2):31-40.