Journal of Machine Learning Research 12 (2011) 2489-2492 Submitted 4/11; Revised 7/11; Published 8/11

LPmade: Link Prediction Made Easy

Ryan N. Lichtenwalter RLICHTEN@ND.EDU
Nitesh V. Chawla NCHAWLA@ND.EDU
Department of Computer Science
University of Notre Dame
Notre Dame, IN 46556, USA

Editor: Geoff Holmes

Abstract

LPmade is a complete cross-platform software solution for multi-core link prediction and related tasks and analysis. Its first principal contribution is a scalable network library supporting high-performance implementations of the most commonly employed unsupervised link prediction methods. Link prediction in longitudinal data requires a sophisticated and disciplined procedure for correct results and fair evaluation, so the second principle contribution of LPmade is a sophisticated GNU `make` architecture that completely automates link prediction, prediction evaluation, and network analysis. Finally, LPmade streamlines and automates the procedure for creating multivariate supervised link prediction models with a version of WEKA modified to operate effectively on extremely large data sets. With mere minutes of manual work, one may start with a raw stream of records representing a network and progress through hundreds of steps to generate plots, gigabytes or terabytes of output, and actionable or publishable results.

Keywords: link prediction, network analysis, multicore, GNU make, PropFlow, HPLP

1. Introduction

Link prediction is succinctly stated as the problem of identifying yet-unobserved links in a network. This task is of increasing interest in both research and corporate contexts. Virtually every major conference and journal in data mining or machine learning now has a significant network science component, and these often include treatments of link prediction. Link prediction is of great use in domains ranging from biology to corporate recruiting, but it is a difficult problem for which to develop models because of extreme class imbalance, the longitudinal nature of the data, the difficulties inherent in effective evaluation, and other issues raised by Lichtenwalter et al. (2010). Further, even for standard prediction algorithms, researchers must often write new code or cobble together existing code fragments. The work flow to achieve predictions and fair evaluation is time-consuming, challenging, and error-prone. LPmade is the first library to focus on link prediction specifically, incorporating general and extensible forms of the predictors introduced by Liben-Nowell and Kleinberg (2007). It also streamlines and parameterizes the complex link prediction work flow so that researchers can start with source data and achieve predictions in minimal time.

There is no shortage of graph libraries: the Boost Graph Library, SNAP, igraph, JGraphT, GraphCrunch, GOBLIN, and many others. Some offer extreme generality, some offer extreme efficiency, some offer modeling utilities, and some have a dizzying array of algorithms. LPmade is not just yet another graph library. Its software components are, by necessity, designed for high performance, and it offers a wide array of graph analysis algorithms, but it is first and foremost an extensive toolkit for performing link prediction to achieve both research and application goals. Unlike other options, LPmade provides an organized collection of link prediction algorithms in a build framework that is accessible to researchers across many disciplines. The software is available at `http://mloss.org/software/view/307/`.

2. The Software Package

The purpose of LPmade is to provide a workbench on which others may conduct link prediction research and applications. For link prediction tasks in many large networks even a restricted set of predictions may involve millions, billions, or even trillions of lines of output. Each unsupervised link prediction method, the supervised classification framework from Lichtenwalter et al. (2010), and all the evaluation tools are optimized for just such quantities of data. Nonetheless, the entire process of starting from raw source data and ending with predictions, evaluations, and plots involves an extensive series of steps that may each take a long time. The software includes a carefully constructed dependency tracking system that minimizes overhead and simplifies the management of correct procedures. Both the build system and the link prediction library are modular and extensible. Researchers can incorporate their own prediction methods into the library and the automation framework just by writing a C++ class and changing a `make` variable.

2.1 Network Library

The LPmade network library is written entirely in scalable, high-performance C/C++ that minimizes memory consumption with a compact adjacency list format based on a vector-of-vectors to represent edges and a translation vector to associate external vertex names to internal identifiers. The library includes clearly written yet optimized versions of the most common asymptotically optimal network analysis algorithms for sampling, finding connected components, computing centrality measures, and calculating useful statistics.

LPmade specializes in link prediction by including commonly used unsupervised link prediction methods: Adamic/Adar, common neighbors, Jaccard's coefficient, Katz, preferential attachment, PropFlow, rooted PageRank, SimRank, and weighted rooted PageRank. The library also has some simpler methods useful in producing feature vectors for supervised learners: clustering coefficient, geodesic distance, degree, PageRank, volume or gregariousness, mutuality, path count, and shortest path count. These methods may be selectively incorporated as features into the supervised framework by Lichtenwalter et al. (2010).

Several graph libraries such as the Boost Graph Library are brilliantly designed for maximum generality and flexibility with template parameters and complex inheritance models. One minor drawback to such libraries is that the code is complex to read and modify. The code base for this library takes a narrower approach by offering fewer mechanisms for generality, but as a result it has a much shallower learning curve.

2.2 GNU `make` Script and Supporting Tools

Although it can be used and extended as such, LPmade is not just a library of C++ code for network analysis and link prediction. It is additionally an extensive set of scripts designed for sophisticated automation and dependency resolution. These scripts are all incorporated into a set of 2 co-dependent Makefiles: task-specific and common. Each new raw data set requires its own task-

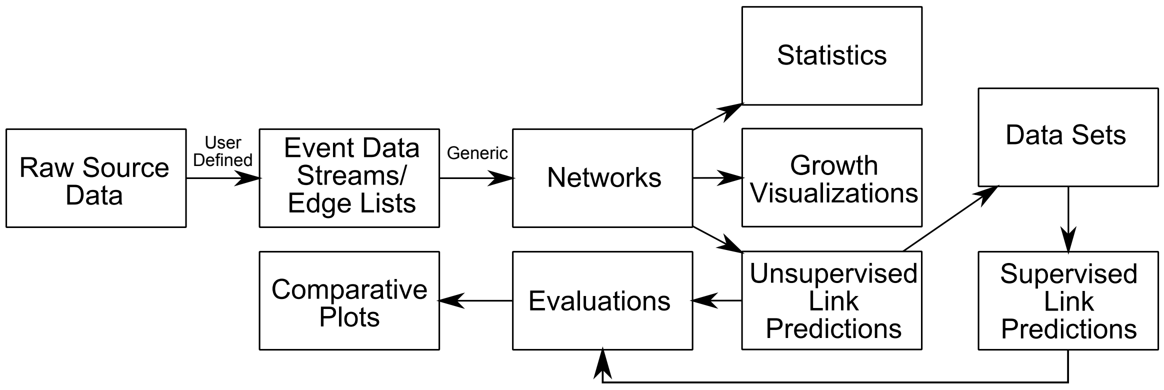

Figure 1: A simplified depiction of some of the build paths in the automation script. Only the first transition is user-defined. Each step involves multiple invocations of many programs to properly assemble data and perform fair evaluation.

specific Makefile, which generally requires less than 20 lines of user code. This Makefile is where users specify the manner in which raw source data is converted to the initial data stream required by subsequent steps in the pipeline. It is also where rules from the common Makefile can be overridden for task-specific reasons. The common Makefile, Makefile.common, includes all the general rules that apply to any network analysis or link prediction task once the task-specific Makefile is written to enable proper handling of raw input. The common Makefile script is designed with advanced template features that allow `make` to modify original Makefile rules in accordance with user requirements. Logical tasks are aggressively provided with their own rules so that the multi-core features of GNU `make` are of optimal benefit. In general, users need not be familiar with writing Makefiles. The important options for the behavior of the automatic build system are presented at the top of the common Makefile along with documentation. For instance, to predict within the 2nd and 3rd degree neighborhoods, set `NEIGHBORHOOD := 2 3`.

Figure 1 illustrates some simplified build paths, and the sample calls below demonstrate several targets with their corresponding actions:

```
make -j 28 sm  # using 28 cores, build a data stream from source, generate required networks, run predictors, and perform evaluations
make -j 8 stats  # using 8 cores, compute several network statistics on the complete network represented by the entire data set
make classify  # construct data sets then use parameters specified in Makefile to train, test, and evaluate
make -j 6 growth  # using 6 cores, generate growth information and plots to describe network saturation
```

Parallelism in these cases is all coarse-grained. Each rule in the Makefile script with no outstanding prerequisites is handled by a separate process to make use of additional cores.

For many large networks, link prediction and supporting analysis yields very large output files. When this prolific output is further combined into data sets, both the I/O capacity and bandwidth requirements may be problematic. To combat this, most steps in the work flow create, accept, and output `gzip`-compressed results. Especially on multi-core systems, this results in a hefty decrease in I/O capacity and bandwidth requirements with a minimal impact on performance. In most cases, the output from `gunzip` is produced faster than the consuming process can accept it. Where necessary, named pipes are used to ameliorate potentially large temporary storage requirements.

2.3 WEKA Modifications

LPmade includes a modified version of WEKA 3.5.8 (Witten and Frank, 2005). It is not meant for direct user invocation. Instead the build system uses WEKA classifier implementations to construct supervised models for link prediction. Unmodified, WEKA has several limitations that make even its command-line mode problematic for operation on enormous link prediction testing sets. These include processing overhead for unwanted computations, Java string overflow and potential thrashing from in-memory result concatenation, and inability to handle compressed C4.5 format input. Alternatives such as MOA solve some but not all of these problems, and WEKA internal classes such as AbstractOutput are unavailable at the command line. We have chosen to modify the WEKA command-line evaluation path to compute only the necessary information and to output directly to standard output for LPmade scripted downstream processing. We have integrated support for processing `gzip`-compressed C4.5 input and use this support in the build system to take advantage of significant space savings on disk.

3. Documentation and Requirements

LPmade comes with `man` pages and a PDF user manual that describes all aspects of the software, most notably describing the setup process, how to use or extend the raw network library, and how to leverage the existing build system to complete many complex steps with short commands. The network library includes an easily extended testing architecture for testing and verification of individual binaries.

The C++ library is written in platform-independent C++ code using only STL extensions. The library may thus be built on any architecture and any operating system that provides a C++ compiler. An included set of high-speed evaluation tools is written in C99 and builds on any system with such a compiler. The bundled distribution of WEKA is cross-platform but requires version 1.5 or higher of the JRE. The automated build system requires GNU `make`. The common Makefile additionally employs many standard tools such as `cut`, `paste`, `sed`, `awk`, `perl`, `sort`, `gzip`, and bundled `gnuplot` 4.4.3.

Acknowledgments

Research was sponsored in part by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053 and in part by the National Science Foundation Grant BCS-0826958.

References

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Inf. Science and Tech.*, 58(7):1019–1031, 2007.

Ryan N. Lichtenwalter, Jake T. Lussier, and Nitesh V. Chawla. New perspectives and methods in link prediction. In *Proc. of the 16th ACM SIGKDD Intl. Conf. on Knowledge Disc. and Data Mining*, pages 243–252, 2010.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, California, USA, second edition, 2005.