– To appear in the COMMPER workshop at ECML/PKDD, 2012 –

Mining Dynamic Networks: The Importance of Pre-processing on Downstream Analytics

Sofus A. Macskassy

Information Sciences Institute University of Southern California
Marina del Rey
CA 90292, USA sofmac@isi.edu,
WWW home page: http://www.isi.edu/~sofmac

Abstract. Dynamic networks are becoming ubiquitous and the analysis of these is becoming increasingly important to better understand the processes by which they evolve over time. Various recent work in this space have looked at how to detect communities, how to model adding/removing nodes and edges, and how to model how nodes change roles over time. In this paper, we look at a different aspect of dynamic networks: assuming we can identify and track a community over time, can we predict larger evolutionary events such as whether it is about to merge with another community or split or dissolve. Further, can we predict at a node-level whether nodes are about to leave the community. We here provide an initial exploration of how different settings for network extraction from observed time-stamped links impact community detection and performance of machine learning. We show on two data sets that changing these parameters have drastic impact on the consistency and stability of communities found.

Keywords: dynamic network analysis, social network analysis, community detection, machine learning, visual analytics

1 Motivation

The amount of social network data available for study is growing exponentially both in terms of complexity as well as in volume. Of particular interest is that this data is temporal in nature, enabling us unprecedented insight into how these networks evolve over time. One aspect of social networks which has received a lot of attention over the past decades is that of detecting communities and this problem translates directly into the dynamic networks as well where one can now detect and track communities over time in larger dynamic networks. There are various aspects of tracking communities which are of interest, but most work has focused on aspects of how communities evolve (e.g., [1, 7, 8, 5]). Most of the work in this area use community detection algorithms in some form to detect communities (e.g., [5, 11]). However, few have looked at the problem of how one actually generates the network from the dynamic data and what impact this has on performance of downstream analytics. We study in this paper how one can extract a "current" network from dynamic data, how this affects community detection and how this impacts machine learning on two prediction tasks: predicting whether a community is stable or about to merge with another community, and whether a node is likely to stay in a community or is about to leave.

To better understand the interplay between these aspects of network generation and analysis, we break up the problem into three parts:

1. Generating snapshots of what the "current" network looks like. We explore in this paper aggregating edges observed within a period of time and decaying edges from prior snapshots. While this generalizes to realtime updating as edges arrive as the period of time decreases to 0, we here explore larger aggregation periods.
2. Identify and track communities across snapshots. In this case, we use a standard modularity clustering algorithm (see [10]) to identify distinct communities in a snapshot and then track communities across snapshots using heuristics to decide whether two communities are the same based on membership overlap. This is in line with what others have done (see, e.g., [8]).
3. Evaluate how well machine learning can predict changes in nodes and communities as we change the parameters for taking snapshots. In particular, we explore whether we can predict if a community is about to merge or not or whether a node is about to leave a community. As such, this is a relatively simple prediction task, but turns out not to be that easy.

Each of these parts contribute to our understanding of dynamic network analysis, and the main contribution of the paper is that we show how the parameters we chose to take snapshots is quite important to downstream analytics. As we use this kind of paradigm to analyze dynamic networks, we need to take care in how these parameters are chosen.

We empirically show how changing the parameters for generating snapshots have drastic impact on the communities found and their stability over time. While this finding is not surprising, it shows the need to pay careful attention to the parameters and how they must be chosen both for the data set in question and for the goal of the analysis.

The remainder of the paper first describes the three parts of our network analysis (Sections 2-4). We then describe our data sets in Section 5 and describe our study of the impact of changing parameters in Section 6. We provide a short discussion of related work in Section 7 and finish with a discussion of our findings in Section 8.

2 Generating Network Snapshots

Dynamic networks as we define them in this paper are networks where edges are observed over time and these edges induces a network over the source and destination nodes. As such, an edge can be defined as e_{ij}^t, where the edge means that some relationship between node i and node j was observed at time t. This relationship can in general directed or undirected, have some semantic meaning and possibly a strength associated with it as well. In this paper we consider the edge to be generic and undirected though it can have a strength associated with it.

Assuming that most edges come at distinct time steps (e.g., there is only one edge observed at time $t \pm \epsilon$), then there would be no active "network" at time t. Therefore, in order to generate a network at time t, we need to consider observations into the past to create an aggregate network. However, an edge observed ten years ago is, in many cases, not as relevant as an edge observed one minute ago and so there is a question of how to deal with past observations.

The approach we take in this paper is to parameterize the network generation with three parameters: δ, the window of time which we consider to be "current" where all edges are taken as is; η, how much to decay edges observed prior to the current window, and γ, the threshold below which an edge is just not strong enough to be relevant and can therefore be removed. This threshold is important for computational reasons as the

network can otherwise become very dense and will have an impact on how well (and how fast) we can detect communities.

Using these three parameters, we can compute the network at time t. We can conceptually represent the network as an adjacency matrix A where the non-negative value $a_{ij} = a_{ji}$ represents the edge between node i and node j. If this is 0 then there is no relation between the nodes. Otherwise the value represents the strength of the relation. Using our three parameters, δ, α and γ, we compute A^t as follows:

$$A^t = A_0^t + \alpha * A^{(t-\delta)},$$

where $A_0^t = \{e_{ij}^{t'} | (t-\delta) \leq t' \leq t\}$. In other words, A_0^t is the network (adjacency matrix) induced by considering all "current" edges at time t. This is a recursive definition, going back to the start of our observed edges. We generate the final network G^t from A^t by pruning out edges whose value is less than γ:

$$\begin{aligned} G^t &= (V^t, E^t) \\ E^t &= \{a_{ij}^t | a_{ij}^t \geq \gamma\}, a_{ij}^t \in A^t \\ V^t &= \{v_i | i \in E^t\} \end{aligned}$$

To simplify notation and without loss of generality, we normalize time such that $\delta = 1$. Thus, given a set of edges over time, $E = \{e_{ij}^{t_0}, \ldots, e_{kl}^{(t_n)}\}$, we generate a set of snapshots: $G^1, \ldots, G^T$, where $G^1 = G^{t_0+\delta}$ and $G^T = G^{t_n}$.

3 Tracking Communities

Given a series of network snapshots, $G^1, \ldots, G^T$, we can now consider the question of identifying and tracking communities over time. We here assume that the dynamic network consists of multiple communities interacting over time and we want to track how these communities change from one snapshot to the next.

To do so, we must first be able to detect a community in a snapshot and second, we must be able to identify whether this community was present in a previous snapshot. While there are numerous community detection algorithms available (see, e.g., [10, 13, 8, 5, 9, 11]). We will in this paper use modularity clustering [10] to induce a set of disjoint communities $C^t = \{c_1^t, \ldots, c_k^t\}$ from G^t, where $c_i^t = \{v_j | v_j \in V^t\}, c_i^t \subseteq V^t$.

Given that we have $C^1, \ldots, C^T$, derived from $G^1, \ldots, G^T$ respectively, we need to identify whether c_i^t is a continuation of $c_j^{(t-1)}$. In fact, as has been enumerated elsewhere (see, e.g., [8, 5]), communities my take a set of actions between time-steps. We here define four major events which we track:

1. **Continue**: $\geq 50\%$ of $c_i^{(t-1)}$ moves on to c_j^t, and $\geq 65\%$ of c_j^t comes from $c_i^{(t-1)}$.
2. **Merge**: $\geq 50\%$ of $c_i^{(t-1)}$ moves on to c_j^t, but makes up $< 65\%$ of c_j^t.
3. **Split**: Significant portions ($> 30\%$) of $c_i^{(t-1)}$ moves on to multiple new communities (whether they join or create new communities is ignored).
4. **Death**: In this case, none of the above happened.

Using the above heuristics, we assign an action to each community at time t. From these, we categorize actions of nodes as well:

1. **Stay**: The community continues or merges and the node stays with that community.
2. **Leave**: The community continues or merges but the node goes to another community or does not belong to any community.
3. **Other:** The community splits or dies. Ignored in the study below.

4 Predicting Changes

The goal of this work was to develop predictive models to predict the action of a community or node at time t. In this paper, we looked at whether we could predict is a community would continue or merge and whether a node would leave or stay. All other actions were ignored for this exploration.

We framed this problem as a basic classification problem and used standard machine learning techniques to learn predictive models. The key factor here is how we generated attributes and labels. Clearly we can generate the labels (actions) automatically based on the processes described above, leaving us with how to generate attributes and the experimental methodology.

For communities, we computed a large set of metrics for each community at each time step. These included density, the ratio of closed triangles (triads) completely within the community or shared with another community (one or two of the nodes were in another community), the sum of edge weights within the community and going outside the community, and the closeness centrality of the most central people.

For nodes, we computed various structural attributes such as degree, ties within the community and to outside community, number of triads within the community and shared with another community, and closeness and betweenness centralities.

While we explored using trends for communities that lasted more than one timestep, we here only discuss results from using the metrics from a given snapshot.

For the experimental methodology, we varied α and γ and evaluated how well machine learning could predict the actions of communities and nodes in the induced networks. Our evaluation metric was area under the ROC curve. We performed 5x2 cross-validation [2], where we sub-sampled the majority class in the training set to make the two classes even because there was a very large class skew. However, we kept the test set at the large class skew. We test a variety of machine learning algorithms, including decision trees, logistic regression and naive Bayes. We used the Weka [14] machine learning toolkit. For brevity, we will only report results from logistic regression.

5 Data sets

We make use of two well-known and quite different data sets to show how changing parameters can have significant effect on downstream analysis.

The first data set is the **Enron email corpus** [6][1]. This data set contains the full email trace for roughly 150 enron employees spanning a period of three years (May 1999 through June 2002). The corpus contains over $500,000$ emails. For our purpose, we extracted the emails that were from one of the Enron employees to another employee. This left us with $60,409$ emails, containing a total of $139,183$ links as emails could have multiple recipients (in the to, cc or bcc fields). We used a δ of one month, meaning that we extracted one snapshot per month. The weight of a edge e_{ij} in a month is the number of emails between nodes i and j in that month.

[1] Available at `http://www.cs.cmu.edu/~enron/`

The second data set is the **World trade flow** data [4][2]. This data contains trade flow data between all countries from 1962 through 2000. While the data contains detailed statistics, we here use the overall trade information. This data details the full amount of imports and exports between two countries in a given year. Because trades increase over time, we normalized values such that the weight of edge e_{ij} between countries i and j denoted the ratio of all goods exported from i which went to j. Further, because this was a fairly dense network, we only kept the top 5 outgoing nodes from each country. We here used a δ of one year because that was the granularity of the particular data itself. The data contains information of 203 countries, not all of which were active in the beginning. The complete data (from 1962 to 2000) contained a total of $32,348$ links.

6 Study

The core of our study was focused on understanding the impact of changing α and γ both on the dynamics of the communities as well as on the performance of our classifier. We analyzed the two data sets with $\alpha \in \{0.5, 0.75, 0.9, 1.0\}$ and $\gamma \in \{0.05, 5.00\}$.

We first visualized how communities changed over time. Figure 1 shows one pair of visual analytics for the Enron data set. In order to keep the visual simple, we removed many of smaller communities as well as many of the edges showing how many people were moving between communities. However, the pair is still quite striking in that the analysis with lower α and higher γ has significantly more deaths and merges, and significantly fewer splits. This was a trend was saw on both data sets.

Second, we explored how well we could learn whether nodes were leaving or staying and whether communities would continue or merge. These were chosen because those were the most prominent classes. While we tested a variety of classifiers, we here report results from logistic regression.

	$\alpha = 0.50$			$\alpha = 0.75$			$\alpha = 0.90$			$\alpha = 1.00$		
	C	M	AUC	C	M	AUC	C	M	AUC	C	M	AUC
$\gamma = 0.05$	151	50	0.425	174	19	0.688	179	20	0.590	198	15	0.596
$\gamma = 5.00$	122	53	0.597	175	34	0.649	184	24	0.492	194	14	0.642

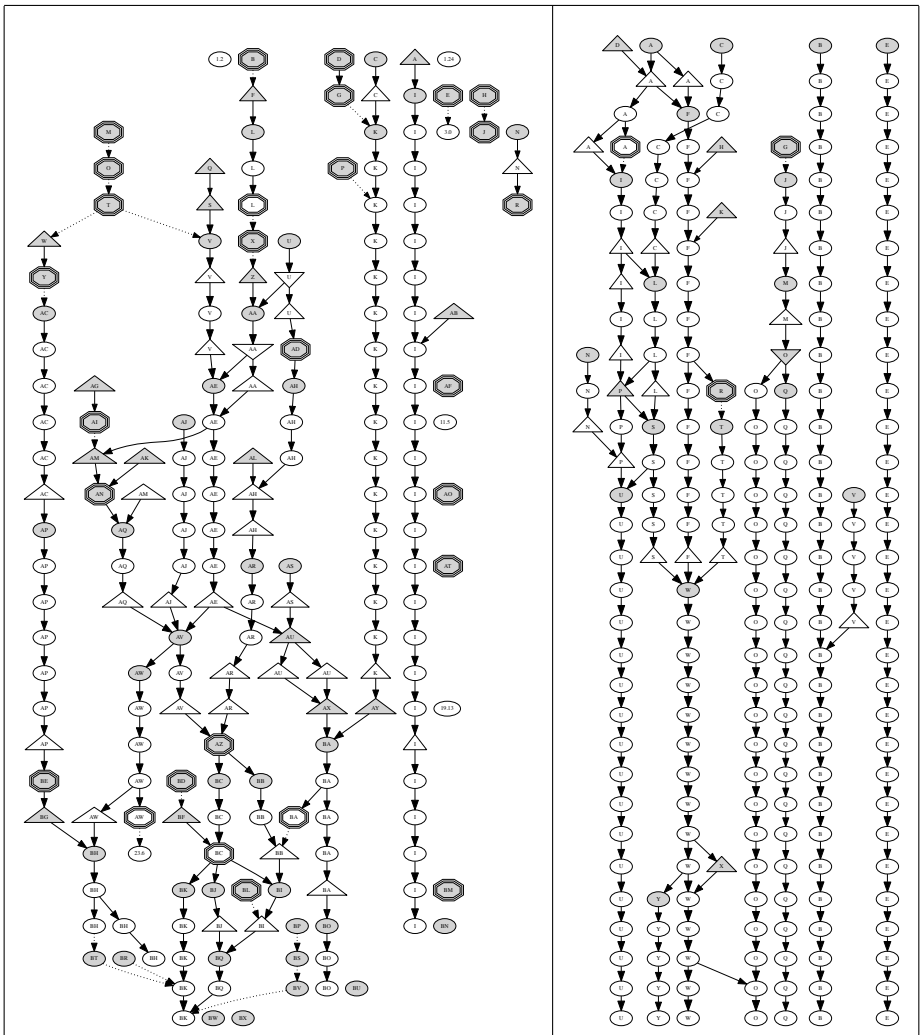
Table 1. Class skew and performance of logistic regression as we varied α and γ to predict community actions (C=continue, M=merge) on the Enron data set. As we can see, there was a strong effect of α on the class skew, but less of an effect with γ.

	$\alpha = 0.50$			$\alpha = 0.75$			$\alpha = 0.90$			$\alpha = 1.00$		
	S	L	AUC	S	L	AUC	S	L	AUC	S	L	AUC
$\gamma = 0.05$	3777	132	0.665	3847	107	0.704	3911	103	0.769	4001	0	1.000
$\gamma = 5.00$	2177	87	0.590	3230	59	0.682	3637	76	0.635	3729	78	0.744

Table 2. Class skew and performance of logistic regression as we varied α and γ to predict node actions (S=stay, L=leave) on the Enron data set. As we can see, both α and γ had a significant effect on class skew.

We first look at the Enron data set, where we varied α and γ to generate the evolving communities. We then used logistic regression to learn a classifier to predict actions for both communities and nodes. Table 1 shows the results of the generated communities

[2] Available at `http://www.nber.org/data/`



(a) Enron $\alpha = 0.5, \gamma = 5.00$ (b) Enron $\alpha = 0.9, \gamma = 0.05$

Fig. 1. Comparison of community evolution on the **Enron data** from 01/2000 through 06/2002 as we vary α and γ. Greyed out shapes mean that this was a new community. Circles means the community continued, triangles mean the community merged at the next step, inverted triangles meant the community split and octagons with thick borders meant the community died at that step. We see that with rapid decay (lower α) and higher γ, communities are much more volatile.

and learning the actions of these. As we can see, there was an effect of α but not much of an effect of γ on the class skew. Logistic regression did perform better than average as is shown by AUC values greater than 0.5 (except for two cases). However, performance did not correlate with α and γ. We see in Table 2 the same study when we learned to predict actions of nodes. In this case we see that both α and γ has significant impact on the class skew. We also see that these correlate well with the classifier performance.

	$\alpha = 0.50$			$\alpha = 0.75$			$\alpha = 0.90$			$\alpha = 1.00$		
	C	M	AUC	C	M	AUC	C	M	AUC	C	M	AUC
$\gamma = 0.05$	172	69	0.681	197	34	0.688	198	27	0.670	201	16	0.639

Table 3. Performance of predicting community actions on the world trade data communities. Again, we see strong impact of α on class skew, but little effect on classifier performance.

	$\alpha = 0.50$			$\alpha = 0.75$			$\alpha = 0.90$			$\alpha = 1.00$		
	S	L	AUC	S	L	AUC	S	L	AUC	S	L	AUC
$\gamma = 0.05$	6008	274	0.650	6233	246	0.691	6080	246	0.711	6484	181	0.660

Table 4. Performance of predicting community actions on the world trade data nodes. As before, we see strong impact of α on class skew, as well as some effect on classifier performance.

We next look at the World Trade Flows data. Tables 3 and 4 show the results of generating our clusters and predicting actions as we vary α and set $\gamma = 0.05$. As before, there is a strong impact on class skew and dynamics both for community and node actions (similar to what we saw in Figure 1). Although there was little change in performance for predicting actions of communities as shown in Table 3, we did see an impact of performance on predicting node activities as shown in Table 4. We are not quite sure why AUC dropped for $\alpha = 1.00$ and this is something that needs to be looked at closer.

7 Related Work

Of most relevance to this work are recent explorations of tracking the evolution of communities [8, 5]. Lin et al. [8] develop a framework for analyzing dynamic social networks, using generative models and stochastic block models. They use this model to track how a set number of communities interact over time, where the communities are identified from the aggregate graph. Greene et al. [5] look at the problem of tracking communities over time, identifying events such as birth, death, split and merge. They use a single similarity score to track communities from step to step and look at the volatility of community events as the threshold is changed.

The bulk of work in dynamic social networks has focused on various aspects of analyzing communities. For example Xu et al., have looked at evolutionary clustering [1] to identify optimal α at each time step to track clusters over time (see, e.g., [15]). This work focused on tracking clusters over time. This problem, and that of detecting communities in dynamic social networks, has also been looked at in various other published works (see, e.g., [13, 3, 9, 11]).

Most related to the prediction of nodes is the literature on predicting churn (nodes leaving the graph). While most work in this space uses non-network predictive models, Richter et al. [12] uses social and community features to identify potential churnes with some success.

8 Conclusion

Analyzing dynamic networks and communities is becoming more prominent both in the literature as well as in industry. However, dynamic networks also add complexity and we need to be mindful of the impact that data pre-processing has on downstream analytics. We have in this paper explored how changing the parameters of generating

snapshots of a dynamic network had a large impact on the observed dynamics of communities and nodes. We additionally saw that this impact on the dynamics also impacted performance of our classifiers when we learned models to predict actions of communities and nodes.

Acknowledgments

This work is based on research sponsored by the Air Force Research Laboratory (AFRL) under agreement number FA8750-12-2-0186. The views and conclusions herein do not represent those of AFRL or the U. S. Government.

References

1. Chakrabarti, D., Kumar, R., Tomkins, A.: Evolutionary clustering. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (2006)
2. Dietterich, T.G.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation 10(7), 1895–1923 (1998)
3. Duan, D., Li, Y., Jin, Y., Lu, Z.: Community mining on dynamic weighted directed graphs. In: Proceedings of the 1st ACM International workshop on Complex networks meet information and knowledge management (2009)
4. Feenstra, R.C., Lipsey, R.E., Deng, H., Ma, A.C., Mo, H.: World trade flows: 1962-2000. Working Paper 11040, National Bureau of Economic Research (January 2005), http://www.nber.org/papers/w11040
5. Greene, D., Doyle, D., Cunningham, P.: Tracking the evolution of communities in dynamic social networks. In: Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM) (2010)
6. Klimt, B., Yang, Y.: Introducing the enron corpus. In: Proceedings of the First Conference on Email and Anti-Spam (CEAS) (2004)
7. Leskovec, J., Backstrom, L., Kumar, R., Tomkins, A.: Microscopic evolution of social networks. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD) (2008)
8. Lin, Y.R., Chi, Y., Zhu, S., Sundaram, H., Tseng, B.L.: Analyzing communities and their evolutions in dynamic social networks. ACM Transactions on Knowledge Discovery from Data (TKDD) 3(2), 1–31 (Apr 2009)
9. McDaid, A., Hurley, N.: Detecting highly overlapping communities with model-based overlapping seed expansion. In: Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM) (2010)
10. Newman, M.: Modularity and community structure in networks. In: Procedings of the National Academy of Sciences. pp. 8577–8582 (2005)
11. Nguyen, N.P., Dinh, T.N., Xuan, Y., Thai, M.T.: Adaptive algorithms for detecting community structure in dynamic social networks. In: The 30th IEEE International Conference on Computer Communications (INFOCOM) (2011)
12. Richter, Y., Yom-Tov, E., Slonim, N.: Predicting customer churn in mobile networks through analysis of social groups. In: Proceedings of the SIAM International Conference on Data Mining. pp. 732–741 (2010)
13. Tantipathananandh, C., Berger-Wolf, T., Kempe, D.: A famework for community identification in dynamic social networks. In: Proceedings of the 13th ACM SIGKDD International conference on Knowledge Discovery and Data mining (2007)
14. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools with Java implementations. Morgan Kaufmann, San Francisco (2000)
15. Xu, K.S., Kliger, M., III, A.O.H.: Tracking communities of spammers by evolutionary clustering. In: Proceedigns of the Workshop on Social Analytics: Learning from Human Interactions at the 27th International Conference on Machine Learning (2010)