Link-based Active Learning

Mustafa Bilgic
Computer Science Dept.
University of Maryland
College Park, MD 20742 USA
mbilgic@cs.umd.edu

Lise Getoor
Computer Science Dept.
University of Maryland
College Park, MD 20742 USA
getoor@cs.umd.edu

Abstract

Supervised and semi-supervised data mining techniques require labeled data. However, labeling examples is costly for many real-world applications. To address this problem, active learning techniques have been developed to guide the labeling process in an effort to minimize the amount of labeled data without sacrificing much from the quality of the learned models. Yet, most of the active learning methods to date have remained relatively agnostic to the rich structure offered by network data, often ignoring the relationships between the nodes of a network. On the other hand, the relational learning community has shown that the relationships can be very informative for various prediction tasks. In this paper, we propose different ways of adapting existing active learning work to network data while utilizing links to select better examples to label.

1 Introduction

Network data, such as online social networks, email communication networks, world wide web, biological networks, online affiliation networks, citation networks are playing an increasing role in our day-to-day life. With availability of increasing computing power and analytical tools, network analysis has become even more important for various purposes, including search, information extraction, targeted advertisement, and drug discovery.

A class of useful network analysis includes prediction; we often want to predict the most influential nodes, missing relationships, duplicate entries, and values of certain attributes of the network members. A classic prediction framework is based on (semi-)supervised learning where a model is trained on an existing data where the network members are annotated with certain attributes. Even though some networks such as online social networks carry a rich set of information, the attribute that we would like to predict is often missing in the data either because the user did not provide the information or the value of the attribute is simply not known a priori. For biological networks, the label might require laboratory experiments. For documents, we might need a human annotator to read a collection of documents. In all these cases, we need to acquire labels through a costly process, such as surveys, human annotation, laboratory experiments, and we often have limited resources.

The active learning literature addresses the issue of gathering the most informative labels under limited resources [4, 6, 11, 14, 15, 17], where the focus is to achieve the best achievable performance using minimal number of acquired labels. However, most of the existing work focused on data that is assumed to be *independent and identically distributed* (IID), where the objects either do not have explicit relationships with one another, or the relationships have been ignored. There are few exceptions to this [1, 2, 10, 18]; however, none of these work offered a general active learning technique for complex networks.

With recent developments in relational models, it has been shown that relationships between objects carry very useful information for the purpose of classification. For example, the collective classification literature has shown that classification performance can be greatly improved if the relationships

between the objects are utilized [3, 8, 7, 13, 16]. Part of the reason for improvement is the fact that related objects' labels tend to be correlated. Thus, it may be feasible to exploit these relationships for developing better active learning techniques as well.

For example, for acquiring labels for medical diagnosis, it is often very helpful to have familial relationships especially for genetic diseases; when we are interested in diabetes and if we have familial relationships between different people in our dataset, then the decisions as to which people to label/query will probably be different if we did not have the familial relationships. When we are interested in virally spreading diseases, then it is useful to know about co-location relationships. When we are interested in assigning topics to scientific papers, then the decisions as to which ones to manually inspect will most likely be different if we also know the citations between the papers than otherwise.

There is a great wealth of research done for developing active learning techniques for data that is assumed to be IID. Most of the existing research can be broadly categorized as *utility-based* active learning where utilities for the unlabeled data are repeatedly computed and the highest-utility objects are queried for their labels. In this paper, we try to adapt the utility-based techniques to network data, so that the existing methods can be utilized and transferred to the network domain. While doing so, we also try to utilize relationships in the network to select better examples to label. In particular, the relationships may be helpful in the following ways:

- One of the general techniques in active learning is to acquire labels for the items that the current model is most uncertain about. The danger with such a technique is the possibility of choosing outliers: the items on which the model is uncertain simply because they are just odd examples, which probably will not help (if not hurt) learning a model that can generalize to unseen data. One possible place where relationships can prove to be useful is to avoid outliers. That is, if we can find out uncertain "regions" rather than uncertain individual items, then we can potentially choose a representative example from the uncertain region and thus potentially avoid outliers.

- One of the concerns in active learning is to be able to explore the space of objects, rather than focusing on certain items, which can happen if the initially learned model is very biased and new item selection is affected by the bias. By using relationships in the network, we can distribute labels across the network rather than focusing on a single region and thus be able to explore the instance space.

- Another technique used to amplify active learning is to exploit the cluster structure in the data [9], where clusters are often defined by attribute similarity between the objects. In addition to attribute similarity, the link structure in the networks can be exploited to cluster the network, which can then be used to augment active learning.

A typical utility-based active learning technique often involves three sub-tasks i) subsampling the space to construct the pool (the candidate space), ii) computing expected utility for each object in the pool, iii) selecting multiple items at once before the utilities are recomputed. These three tasks have been studied heavily for IID classification. In this paper, we address how to augment these tasks to utilize the links in the network.

2 Our Approach

A typical pool-based active learning works as follows: first the space is subsampled to construct the pool of examples. Then, until the budget is exhausted, the items in the pool are assigned a utility score, some of these items are added to the training data, the model is re-trained, and the steps are repeated. The corresponding algorithm is provided in Algorithm 1.

Multiple tasks in this general algorithm can be augmented to utilize the relationships in the network. We specifically modify pool selection (step 1), utility computation (step 5), and selecting multiple instances at once (step 6). We first explain how to modify the utility computation, and then explain how to modify the pool construction and instance selection.

Algorithm 1: General active learning algorithm.

Input:
$\mathcal{U}$ – set of unlabeled examples, $\mathcal{M}$ – the untrained model
Output:
$\mathcal{L}$ – the training set, $\mathcal{M}$ – model trained on $\mathcal{L}$

Choose $\mathcal{P} \subset \mathcal{U}$ `/* Pool selection */`
Choose initial $\mathcal{L} \subset \mathcal{P}$; Remove $\mathcal{L}$ from $\mathcal{P}$ `/* Initial data to learn the model */`
while *stopping criterion not met*
    Train $\mathcal{M}$ on $\mathcal{L}$
    foreach $X_i \in \mathcal{P}$ compute $utility(X_i)$
    Select a subset $\mathcal{S} \subset \mathcal{P}$ based on the $utility$ scores
    Remove $\mathcal{S}$ from $\mathcal{P}$; Add $\mathcal{S}$ to $\mathcal{L}$

2.1 Utility Computation

In active learning, the $utility$ of an individual item X_i represents how much adding that item is expected to contribute to learning a model that can generalize to unseen items. Active learning methods differ mostly on their choice of utility definitions and computations. Uncertainty sampling [6] defines the utility of an instance X_i to be the uncertainty of the model $\mathcal{M}$ on X_i. For probabilistic models, the uncertainty can be defined in terms of probabilities [6]. Tong and Koller [17] define the utility of X_i to be how much it is expected to reduce the version space. Seung et al. [15] uses a committee of classifiers and defines the utility to be the amount of disagreement between the committees. Saar-Tsechansky and Provost [12] subsample the training data $\mathcal{L}$ to learn multiple classifiers and define the utility of an instance to be the variance of the predictions of the classifiers. Roy and McCallum [11] define the utility to be the reduction in expected future error the instance is expected to provide.

One of the concerns is that an individual item might have a (seemingly) high utility, and yet the instance can be a rare instance or an outlier and thus might not be useful to query; on the contrary, it can even lead to an incorrect model that does not generalize well. In network data, related instances' labels and attributes tend to be correlated. One of the possible ways that relationships can help is to avoid outliers; if an individual item has high utility but its neighbors do not have high utility, then it is very likely that the item is particularly odd. Ideally, an active learning technique needs to query items that both have high utility and lie in the high density regions.

Making use of the correlations through relationships, we would like to find the high utility regions by looking at instances and their neighbors. A simple extension of the utility based approaches is as follows. First compute an individual utility for each object. Then, for each object, take an (weighted) average of its utility value along with its neighbors' utility values. If an object's neighbors have low utility values, then that object is particularly odd, and possibly carries little useful information for other objects in the data. Maybe it is particularly noisy. On the other hand, if an object's own utility and neighbor utility values are high, then it is a possible indication that querying on this region will provide useful information for the model:

$$utility(X_i) = w_0 \times score(X_i) + \sum_{X_j \in \mathcal{N}_i} w_{ij} \times score(X_j)$$

2.2 Pool Subsampling

The set of all unlabeled examples, $\mathcal{U}$, is often fairly sizable and thus it is impractical to compute utility values for all members of $\mathcal{U}$. Often a pool $\mathcal{P} \subset \mathcal{U}$ is sampled and utility values for the members of $\mathcal{P}$ are computed. We are not aware of any work specifically targeted at how to choose $\mathcal{P}$, however, the common practice is to sample $\mathcal{P}$ randomly from $\mathcal{U}$. The advantage of random sampling is that $\mathcal{P}$ will be representative of the whole data $\mathcal{U}$. However, random sampling for network data might not be a good idea, simply because random sampling does not guarantee that members of $\mathcal{P}$ will be connected and thus the information that can be utilized from the relationships might be lost. The question is then how to sample $\mathcal{P} \subset \mathcal{U}$ for network data.

Ideally, we want $\mathcal{P}$ to be both representative of the underlying domain and also maximally connected. Finding a small connected subcomponent that is representative of the underlying label and

Algorithm 2: Three-phase pool construction for network data.

Input:
$\mathcal{U}$ – set of unlabeled examples, p – the desired pool size
Output:
$\mathcal{P}$ – The pool

1 **Exploration phase:** Randomly sample $\mathcal{P} \subset \mathcal{U}$ such that $|\mathcal{P}| = \frac{p}{3}$
2 **Expansion phase:** Choose a random neighbor for each $X_i \in \mathcal{P}$ and add it to $\mathcal{P}$
3 **Connection phase:** **while** $|\mathcal{P}| < p$ choose $X_j \in \mathcal{U} \setminus \mathcal{P}$ such that X_j has the highest number of neighbors in $\mathcal{P}$ and add X_j to $\mathcal{P}$

attribute distribution can be quite challenging. Even though there are existing snowball sampling [13] and subgraph sampling techniques [5], they are not aimed at active learning. In this paper, we propose a three-phase sampling approach for active learning.

The sampling approach that we propose consists of three phases: an exploration phase, an expansion phase, and a connection phase. The purpose of the *exploration* phase is to explore the space as much as possible and get a representative sample; in this phase, we sample $\frac{|\mathcal{P}|}{3}$ items randomly from $\mathcal{U}$. In the *expansion* phase, we want to balance exploration and connection; thus, for each instance that was added in the exploration phase, we add a random neighbor. In the *connection* phase, the purpose is to increase the connectivity as much as possible; thus, we add items that have the highest number connections to the items that have been added in the previous steps. This three-phase algorithm is presented in Algorithm 2.

2.3 Instance Selection

In general active learning, once the utility values for each instance in the pool are computed, multiple instances, say $k > 1$, are queried for their labels, simply because re-training the model and re-computing the scores with the updated model is often an expensive process. Different techniques have been employed to choose which k instances to query. It has been shown that choosing the top k instances is the wrong thing to do [6, 12]; for binary classification, Lewis and Gale [6] choose $\frac{k}{2}$ items that have probabilities close to 0.5 from above and $\frac{k}{2}$ items that have probabilities close to 0.5 from below. Saar-Tsechansky and Provost [12] uses the utility values as weights and samples k items from the pool with probability in proportion to their utility values. Here, we explore how we can use relationships to guide which k items to query.

In a typical network, nodes have varying degrees. Because we compute the utility values as a weighted average of individual utility and neighbor utility values, some nodes' utility values are based on only very few nodes whereas others' are based on many nodes. One heuristic then would be to choose the high utility nodes who also have high degree; by doing so, we are choosing nodes whose final utility values are "supported" by many nodes rather than just a few nodes. In essence, we are sampling from high-utility-high-density regions rather than high-utility-low-density regions/items. We take a simple approach and probabilistically choose k nodes in proportion to their degrees from top $2 * k$ nodes with highest utility values.

3 Experiments

We experimented with a cross-product of utility computation (individual, neighbors), pool sampling (random, three-phase), and multiple instance selection (utility-weighted, degree-weighted) techniques. We used Naive Bayes as the classifier and used probabilistic uncertainty (entropy) as the base utility function. At each step, we acquired $k = 5$ items to label before the model is re-trained the utilities are recomputed. In the utility-weighted instance selection scheme, we chose k items probabilistically from top $2 * k$ highest-utility items, with probabilities being in proportion to their utility values. In the degree-weighted scheme, we chose k items probabilistically from top $2 * k$ highest-utility items, with probabilities being in proportion to their degrees. We experimented on two real-world publication datasets: Cora and CiteSeer, which are available online at `http://www.cs.umd.edu/projects/linqs/projects/lbc/`. The Cora dataset contains a 2708 machine learning papers divided into one of seven classes while the CiteSeer dataset has 3312 documents belonging to one of six classes. The methods that we tried are listed as follows:

Table 1: Different techniques tried.

Score Computation		**Pool Selection**		**Multiple Instance Selection**
Individual	x	Random	x	Utility-weighted
Individual + Neighbor		Three-Phase		Utility + Degree-weighted

We compared the different acquisition techniques under two setups: the inductive case and the transductive case. In the inductive case, we set aside a portion of the data as the test data and used the remaining data as the set of unlabeled examples $\mathcal{U}$. The inductive case makes sense for dynamic networks where we train on a snapshot of the network at time t and use the model for prediction for later time steps. In the transductive case, we used all the data as the unlabeled set $\mathcal{U}$ and the examples that were not selected for training were used for testing. The transductive case corresponds to having all the data at hand and the task is to maximize performance on the given data, subject to the budget for acquiring labels. We first present results for the inductive case.

3.1 Inductive Case

We performed 10-fold cross validation, where a tenth of the data was separated as the test data and the remaining nine-tenth was used as $\mathcal{U}$. The pool $\mathcal{P}$ was selected to be half of the unlabeled set $\mathcal{U}$. An initial random set (1% of $\mathcal{P}$) was selected for training the initial model. Then, at each step, 1% more data was added using different acquisition techniques and the models learned on the resulting training sets were evaluated on the test data. These steps were repeated five times for each fold; thus, the results were averaged over 50 runs. We report the percent error reduction over random instance selection; the results for Cora are presented in Table 2 and for CiteSeer in Table 3.

Table 2: Percent error reduction of the different acquisition techniques over random acquisition for the Cora dataset for the inductive case. Each row represents the average reduction for a budget interval; for example, the 1%-5% row represents the average error reduction when the budget is between 1% and 5% of the pool size. For each row, the best method and the methods that do not differ significantly (as measured by t-test with 95% confidence) from the best method are highlighted.

Utility Computation	Individual	Individual + Neighbor			
Pool Selection	Random	Random		Three-Phase	
Instance Selection	Utility	Utility	Utility + Degree	Utility	Utility + Degree
1%-5%	**4.92%**	**5.25%**	**5.16%**	4.64%	**5.41%**
6%-10%	22.30%	25.70%	28.89%	26.82%	**32.50%**
11%-15%	31.16%	33.55%	36.09%	37.75%	**38.51%**
16%-20%	30.73%	31.94%	31.64%	33.51%	**33.91%**
Overall (1%-20%)	22.28%	24.11%	25.45%	25.68%	**27.58%**

Table 3: Percent error reduction of the different acquisition techniques over random acquisition for the CiteSeer dataset for the inductive case.

Utility Computation	Individual	Individual + Neighbor			
Pool Selection	Random	Random		Three-Phase	
Instance Selection	Utility	Utility	Utility + Degree	Utility	Utility + Degree
1%-5%	8.56%	7.95%	**10.00%**	7.78%	**10.04%**
6%-10%	32.28%	32.07%	33.17%	34.00%	**35.61%**
11%-15%	**29.89%**	28.89%	28.51%	28.71%	28.40%
16%-20%	**18.56%**	18.03%	17.91%	16.69%	16.51%
Overall (1%-20%)	22.32%	21.74%	22.40%	21.80%	**22.64%**

We present the average percent error reduction in four intervals: 1%-5%, 6%-10%, 11%-15%, and 16%-20%. For each row, the best performing method as well as the methods that do not differ statistically significantly from the best method are highlighted in bold. For the Cora dataset, the method that exploits the relationships fully, i.e. the method that i) computes utilities as a combination

of individual utilities and neighbors utilities, ii) samples the pool using the three-phase algorithm, and iii) samples multiple instances by using the utility and degree heuristic, performs the best.

For CiteSeer however, even though exploiting relationships perform significantly better overall, the differences at different percentages are dramatically different; initially, 1%-5% and 6%-10%, exploiting relationships have a clear advantage; however, the method that ignores the relationships have a clear advantage in the second half, 11%-15%, and 16%-20%.

3.2 Transductive Case

In the transductive setup, the set $\mathcal{U}$ initially consists of the whole network. Again, in this case, we select $\mathcal{P}$ to be half of $\mathcal{U}$. In this case, however, the test set consists of all the nodes that are not chosen to be added to the label set $\mathcal{L}$, i.e. the test set is $\mathcal{U} \setminus \mathcal{L}$. We report averages over 10 runs. The results for Cora and CiteSeer are shown in Tables 4 and 5 respectively.

Table 4: Percent error reduction of the different acquisition techniques over random acquisition for the Cora dataset for the transductive case.

Utility Computation	Individual	Individual + Neighbor			
Pool Selection	Random	Random		Three-Phase	
Instance Selection	Utility	Utility	Utility + Degree	Utility	Utility + Degree
1%-5%	**7.00%**	**6.93%**	**7.82%**	4.82%	**7.68%**
6%-10%	23.77%	29.51%	32.55%	34.22%	**37.00%**
11%-15%	33.96%	36.35%	38.30%	**40.32%**	**40.17%**
16%-20%	34.28%	35.39%	**35.83%**	**35.92%**	34.81%
Overall (1%-20%)	24.75%	27.05%	28.63%	28.82%	**29.92%**

Table 5: Percent error reduction of the different acquisition techniques over random acquisition for the CiteSeer dataset for the transductive case.

Utility Computation	Individual	Individual + Neighbor			
Pool Selection	Random	Random		Three-Phase	
Instance Selection	Utility	Utility	Utility + Degree	Utility	Utility + Degree
1%-5%	9.48%	9.33%	13.60%	11.43%	**15.40%**
6%-10%	31.73%	35.61%	**37.13%**	35.80%	**37.12%**
11%-15%	30.22%	**30.66%**	29.96%	29.00%	28.75%
16%-20%	**22.59%**	21.92%	21.90%	19.88%	18.99%
Overall (1%-20%)	23.51%	24.38%	**25.65%**	24.03%	25.07%

The results for the transductive case are very similar to the ones in the inductive case, however, this time, the network methods have a better advantage than the individual method for the CiteSeer dataset compared to the inductive case. Part of the reason could be that, in the inductive case, we lost relationships when we separated the test set; here, the test set was not separated, and thus the relationships were preserved.

4 Future Work

This is just the first step towards exploiting relationships for active learning. Even though we have proposed successful methods, the richness of the network data, i.e. multiple types of relationships, group memberships, etc. presents many opportunities for new active learning techniques. Moreover, we proposed how to use links for choosing better items to label for IID classification. Developing general purpose active learners for relational models is a promising future direction.

5 Conclusions

Network data is now ubiquitous and it lends itself to many opportunities to exploit the relationships for many prediction tasks. The active learning literature has remained relatively agnostic to rela-

tionships in the data. Our initial work strives to exploit relationships in the data to guide active learning for better utility computation, pool selection, and instance selection. It also addresses challenges involved in developing active learning techniques specifically targeted to relational models, and hopefully raises interest and provides insights into how relationships can be exploited to develop better active learning techniques.

Acknowledgments

This work was supported by ARO Grant No. W911NF-08-1-0466 and NSF Grant No. 0746930.

References

[1] Brigham Anderson and Andrew Moore. Active learning for hidden markov models: objective functions and algorithms. In *International Conference on Machine Learning (ICML)*, pages 9–16, 2005.

[2] Mustafa Bilgic and Lise Getoor. Effective label acquisition for collective classification. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 43–51, 2008.

[3] Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *International Conference on Management of Data (SIGMOD)*, pages 307–318, 1998.

[4] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[5] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 631–636, 2006.

[6] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Conference on Research and Development in Information Retrieval (SIGIR)*, pages 3–12, 1994.

[7] Qing Lu and Lise Getoor. Link based classification. In *International Conference on Machine Learning*, pages 496–503, 2003.

[8] Jennifer Neville and David Jensen. Iterative classification in relational data. In *SRL Workshop in AAAI*, 2000.

[9] Hieu T. Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *International Conference on Machine Learning (ICML)*, 2004.

[10] Matthew Rattigan, Marc Maier, and David Jensen. Exploiting network structure for active inference in collective classification. In *ICDM Workshop on Mining Graphs and Complex Structures*, pages 429–434, 2007.

[11] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *International Conference on Machine Learning*, pages 441–448, 2001.

[12] Maytal Saar-Tsechansky and Foster Provost. Active sampling for class probability estimation and ranking. *Machine Learning*, 54(2):153–178, 2004.

[13] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3), 2008.

[14] Burr Settles. Active learning literature survey. Computer Science Technical Report 1648, University of Wisconsin–Madison, 2009.

[15] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *ACM Annual Workshop on Computational Learning Theory*, pages 287–294, 1992.

[16] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 485–492, 2002.

[17] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2002.

[18] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003.