Accurate Semi-supervised Classification for Graph Data

Frank Lin
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
frank@cs.cmu.edu

William W. Cohen
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
wcohen@cs.cmu.edu

ABSTRACT

Most machine learning algorithms require labeled instances as training data; however, not all instances are equally easy to obtain labels for. For example, the best-known papers and/or websites would be easier for a domain expert to label. We propose a new PageRank-style method for performing semi-supervised learning by propagating labels from labeled seed instances to unlabeled instances in a graph using only the link structure. We show that on four real-world datasets, the proposed method, using only a small number of seed instances, gives highly accurate classification results and both outperforms simple content-only baselines and is competitive with state-of-the-art fully supervised algorithms that uses both the content and the link structure. In addition, the method is efficient for large datasets and works well given high-PageRank seeds.

1. INTRODUCTION

Traditional machine learning methods require large amounts of labeled training instances, which are sometimes difficult to obtain. Two proposed solutions to this problem are *semi-supervised learning methods [20], which make use of unlabeled instances in addition to labeled instances, and* active learning methods (e.g., [9, 18]), which reduce the number of labels required for learning by intelligently choosing the instances to label.

One important issue that has not been considered in prior work is that in many practical settings, *some instances are easier to label than others. For example, in website classification, a better-known website would probably be easier for a domain expert to label, since the expert would be more likely to be familiar with it, and since the website would be less likely to be difficult-to-evaluate because of having content that is limited (or simply incoherent). Similarly, in classifying technical papers, the more influential papers would probably be easier for a subject expert to label.*

In this paper, we propose a semi-supervised learning method for data in graphs. The problems of measuring authority [16, 12] and detecting communities [8, 4] in graphs have been widely studied, as has the task of collective classification, i.e., classification of data in graphs—e.g. web page classification [6, 5] and scientific paper classification [14, 13]. Experimentally, our method performs very well when trained on a small set of highly authoritative examples (where authority is assessed using PageRank [16] on the instance graph). Indeed, on four datasets from two different domains, the method is competitive with state-of-the-art *fully supervised methods for learning in graphs [14, 13]—a surprising result given that many fewer labels are used by our method, and that the method (as currently implemented) makes no use of the "content" of the instances, only the graph structure. Furthermore, the method is also highly scalable, requiring time linear in the number of edges of the graph.*

Below, we will first present the MultiRank bootstrapping method, which can be summarized as a PageRank-style method for propagating labels from seed instances to other instances in the graph using the link structure. The method assumes the instance graph is homophilous—i.e., that instances belonging to the same class tend to link to each other. Homophily is found in many natural networks—including networks of websites, blogs, and paper citations—and often arises when a single network is jointly constructed by several communities (each associated with a class). In our method, we also associate each *link with a class, from the same set as used for the nodes, and then treat the link as a class-specific "recommendation" for the linked-to node. This differs from most uses of link structure, in which links are considered as* universal "recommendations". Associating each link with a class helps to constrain the propagation of labels.

We then test the MultiRank method on four different datasets—two political blog datasets and two citation datasets—and compare its classification performance against several baseline methods. For three of the four datasets, the instances correspond to documents, and we compare to fully-supervised methods for text classification methods. Two of the four datasets are two-class datasets, and on these we compare to state-of-the-art spectral clustering methods. Finally, on two of the datasets, we compare to a state-of-the-art collective classification method, stacked graphical learning. We also compare our results to prior experiments on the same datasets.

The experiments show that our method outperforms simple content-only baselines, is competitive with or outperforms spectral clustering methods, and shows competitive performance against the supervised collective classification algorithms. We show that the method is efficient for large datasets, and works when trained only on high-PageRank seeds that are arguably easier to label.

2. PROPOSED ALGORITHM

PageRank [16, 10] is a link structure-based ranking algorithm widely used in determining the authority of a website in the World Wide Web or the importance of a node in a directed graph. It is defined as follows: given a direct graph G and a personalization vector $\mathbf{u}$, it returns a ranking vector $\mathbf{r}$ satisfying the following equation:

$$\mathbf{r} = (1-d)\mathbf{u} + dW\mathbf{r} \tag{1}$$

where W is weighted transition matrix of graph G where transition from i to j is given by $W_{ij} = 1/degree(i)$ and d is a constant damping factor. If we assume a uniform personalization vector (which we shall do throughout the rest of this paper), Equation 1 can be simply defined in terms of G ($\mathbf{r} = PageRank(G)$) and the $\mathbf{r_i}$ can be interpreted as the probability of a random walk on G arriving at node i, with teleportation probability $(1-d)$ at any time to any node with a uniform distribution.

2.1 MultiRank

In the context of web surfing, PageRank assumes that, after arriving at a page, to go to his or her next destination, the web surfer clicks on one of the outgoing links randomly and with equal probability. However, in a real-world setting the surfer will discriminate between the links and click on one after determining which link points to a page of his or her preference. For example, a user surfing among a network of political blogs will most likely follow links that point to a blog of his or her political leaning. If we imagine every link is annotated with a political leaning label e.g. liberal or conservative), a liberal user may "walk" the network following only liberal links. This idea naturally gives rise to MultiRank, a natural extension to PageRank defined over a graph containing nodes belonging to different classes or *factions* (through out the rest of the paper, *class* and *faction* will be used synonymously). MultiRank is defined as follows: given a directed graph $G = (V, E)$, and a set of edges E_f belonging to faction f, it returns a ranking vector $\mathbf{r_f}$ of faction f satisfying the following equation:

$$\mathbf{r_f} = (1-d)\mathbf{u} + dW_f\mathbf{r_f} \tag{2}$$

where W_{fij} is W_{ij} if the edge from i to j is in E_f, otherwise zero; and $\mathbf{u}$ is the uniform personalization vector where $u_i = 1/|V|$. This simple modification of the PageRank is very intuitive and $\mathbf{r_f}$ can be seen as the probability of a random walk on G if the we only follow edges belongs to faction f, and this probability can be interpreted as a ranking score on how popular or authoritative a blog is [16] with respect to faction f.

2.2 Bootstrapping

The original PageRank algorithm requires only the simple link structure as the input. However, in order to calculate $\mathbf{r_f}$ for MultiRank, we need E_f – faction labels for the edges, which we do not have. In order to solve this problem, we propose an iterative algorithm to bootstrap the labeling of nodes and edges from a set of initial seeds nodes S:

STEP 0:
Initialize all nodes to have uniform ranking for all factions and mark them as not confident. Initialize all edges to be unlabeled (not belonging to any faction). Use seed labels to label some initial nodes and mark them as confident.

STEP 1:
Label each edge according its target and source nodes. If the target node is marked confident, label the edge with target node's faction label; else if the source node is marked confident, labeled the edge with source node's faction label; if neither of the incident nodes are marked confident, leave the labeling unchanged (the edge could be unlabeled).

STEP 2:
According to edge labeling, run MultiRank on edges belonging to each of the factions and produce a ranking matrix R where $R(i, f)$ is the ranking of node i with respect to faction f. Note that unlabeled edges do not belong to any faction and therefore do not influence the output R.

STEP 3:
Label each node according to its faction rankings and neighbor's faction rankings and mark whether this labeling is confident. We define a score function P for a node i with respect to faction f as follows:

$$P(i,f) = \frac{R(i,f) + \sum_j R(j,f)}{|N|+1} : j \in N \tag{3}$$

where $j \in N$ if and only if $\{i, j\} \in E$. We then label i according to $\max_f P(i, f)$ and mark it confident if $\max_f P(i, f) > \frac{1}{|V|}$. In other words, this scoring function is a "smoothed" ranking over the MultiRank of the node and its succeeding neighbors, and a labeling of the is confident if and only if the score for this node is larger than if the score were distributed uniformly among all nodes.

STEP 4:
If during STEP 1 any previously unlabeled edge was labeled, or if any edge that was previously labeled had been re-labeled with a different label, go to STEP 1, otherwise stop.

Notice that the methods for labeling edges in STEP 1 and for labeling nodes in STEP 3 are very simple. We can imagining replacing them with more sophisticated algorithms or classifiers. If we consider STEP 1 and 3 to be replaceable with any classifier that uses the same input to produce labels and confidences for edges and nodes, we can define the MultiRank bootstrapping procedure formally as shown in Figure 1.

```
Given: A graph $G$, a set of examples $X_V$ corresponding to nodes in $G$, a set of examples $Y_V^0$ corresponding to seed labels, a set of examples $X_E$ corresponding to edges in $G$, a node classifier $C_V$, an edge classifier $C_E$, and MultiRank algorithm that, given edge labels $Y_E$ returns a ranking matrix $R$ where $R(i, f)$ is the ranking of node $i$ with respect to faction $f$
For t=0,1,2,...
    1. Set $Y_E^t \leftarrow C_E(G, Y_V^t)$ if confident
    2. Set node rankings $R \leftarrow MultiRank(G, Y_E^t)$
    3. Set $Y_V^{t+1} \leftarrow C_V(G, R)$
    4. Stop if $Y_E^t = Y_E^{t-1}$
```

Figure 1: The generic MultiRank bootstrapping method

2.3 Similarity to the Yarowsky Algorithm

To perhaps better understand the MultiRank bootstrapping method and also as a point of interest, we draw its similarity to the Yarowsky algorithm [19, 2], a well-known bootstrapping algorithm in computational linguistics. The Yarowsky algorithm can be generically defined as shown in Figure 2.

```
Given: Examples $X$, initial labels $Y^0$, and an inner classifier $C$
For t=0,1,2,...
    1. Train $C^t$ on $Y^t$
    2. Set $Y^{t+1} \leftarrow C^t(X)$ if confident
    3. Stop if $Y^t = Y^{t+1}$
```

Figure 2: The generic Yarowsky algorithm

If we modify the MultiRank bootstrapping method so that we are given Y_E^t instead of Y_V^0 and we move STEP 1 so that it executes right after STEP 3, the two algorithm would indeed seem very much alike. In addition, both algorithms use a confidence threshold to iteratively expand the pool of labeled examples and both algorithms re-label previously labeled examples–even the initial seed. Furthermore, both algorithms rely on strong model assumptions: the Yarowsky algorithm exploits specific properties of human language found in texts (*one sense per discourse* and *one sense per collocation* [19]); and the MultiRank bootstrapping method takes advantage of the link properties found in network data containing distinct *communities* or *factions*.

2.4 Selecting Seeds

The MultiRank bootstrapping method described requires seed instances, and we propose using more popular or authoritative instances. There are two advantages to prefer highly popular or authoritative instances as seeds:

First, popular or authoritative instances are easier to obtain labels for because a) domain experts are more likely to recognize them and label them without detailed assessment of the instances, resulting in less time and human effort spent, and b) popular or authoritative instances are more likely to have already labeled available. For instance, websites such as *www.etalkinghead.com* contain blog directories that are organized according to political leaning. Although the directories contain relatively few blogs (150 liberal, 148 conservative, and 48 libertarian blogs as of the time of this writing), these can be readily used as seed instances.

Second, popular or authoritative instances will likely to have many incoming links (other instances are more likely to link to or cite them) and sometimes outgoing links as well (in the case of blogs, popular blogs are usually well-kept and contain more entries and links). Having many incoming and outgoing links helps to propagate the labels faster and more reliably when using a semi-supervised link-based algorithm such as the one proposed here.

Based on these observations, we propose a simple seeding method to test our classification algorithm: **at-least-n-per-class**. This method requires running the original PageRank algorithm first on the unlabeled graph. Given a number n, we sort the instances by their PageRank score from highest to lowest. Starting at the top of the list and going down, we mark each one as a seed instance until at least n seeds per class have been marked. This method could be seen as a simulating a domain expert labeling a given list of instances according to some popularity or authoritative ranking (either PageRank or other heuristics) and labeling the list until n instances per class have been labeled.

3. DATASETS

To assess the effectiveness of our algorithm, we test it on four different datasets. The first two datasets are from the political blog domain, which we refer to as the Kale dataset and the Adamic dataset, and the second two datasets are from scientific paper citations, which we refer to as the Cora dataset and the CiteSeer dataset. Some statistics of the four datasets can be find in Table 1.

The Kale dataset is constructed in the same way as Kale et al. did in [11], by first finding a set of overlapping blogs between the ICWSM 2007 BuzzMetrics [1] dataset and Lada Adamic's labeled dataset [3], then a graph is generated using links found in the BuzzMetrics dataset posts, and lastly we take the largest (weakly) connected component of the graph. Because of the small size of the dataset, we added 50 additional labeled blogs to it by crawling *www.blogcatalog.com*. So our Kale dataset is slightly larger than the one described in [11] and we end up with a dataset of 404 connected blogs that are labeled either *liberal* or *conservative*. This dataset contains both link structure and content text.

The Adamic dataset is constructed by simply creating a graph from Lada Adamic's political blog dataset [3] and taking the largest (weakly) connected component. This dataset, contains 1222 connected blogs. Every blog within this datasets are labeled either *liberal* or *conservative*. This dataset contains only the link structure but no content text.

Although the two political blog datasets are not entirely independent (as the blogs from the Kale dataset are mostly a subset of the blogs from the Adamic dataset), using the above procedure we effectively create two distinct datasets of different sizes and link structures. In the Kale dataset, the links are gathered around May 2006 from the content of the blog posts; in the Adamic dataset, the links are from two months before the 2004 presidential election and are extracted mostly from the sidebars of the blogs. The links from the Kale dataset can be considered more *temporary*, pertaining to the blogger's interests at the time of the post, while links from the Adamic dataset can be considered more *stationary*, indicating the blogger's long-term interests and recommendations. Finally, it should be pointed out that the labeling of the political blog datasets is not 100% accurate as noted in [3]. Class label distribution for the two political blog datasets can be found in Table 2.

The other two datasets are scientific paper citation datasets. The Cora dataset contains 2708 papers from 7 categories: Neural_Networks, Case_Based, Reinforcement_Learning, Probabilistic_Methods, Genetic_Algorithms, Rule_Learning, and Theory. The CiteSeer dataset contains 3312 papers from 6 categories: HCI, IR, Agents, AI, ML, and DB. The details of their construction can be found in [14]. Since our algorithm takes only the link structure and a small number seed nodes as input, we require the graph to be connected. Again, we extract the largest (weakly) connected component from these datasets and end up with 2485 papers for the Cora dataset and 2110 papers for the CiteSeer dataset. An edge exist in the graph from node a to node b if paper a cites paper b. Class label distribution for the two scientific paper citation datasets can be found in Table 3.

	Nodes	Edges	Density	IntraR
Kale	404	2725	0.01670	0.861
Adamic	1222	19021	0.01274	0.912
Cora	2485	5209	0.00084	0.808
CiteSeer	2110	3757	0.00084	0.741
Cora*	2708	5429	0.00074	
CiteSeer*	3312	4732	0.00043	

Table 1: Some statistics on the four datasets. Density is the ratio of number of edges to the number of nodes squared. IntraR is the intra-faction edge ratio – the number of edges where the source and destination nodes belonging to the same class divided by the total number of edges. Cora* and CiteSeer* are the original datasets before being reduced to a connected component.

	Kale	Adamic
Liberal	198	586
Conserv.	206	636

Table 2: Class distribution for the Kale and Adamic datasets.

4. EXPERIMENTS

We run our algorithm on four datasets and compare its classification accuracy and number of seed labeled instances against a number of baseline and competitive algorithms. In the section below we will describe each algorithm and their settings.

Content-only Baseline

We use a Naïve Bayes classifier with bag-of-words features as the content-only baseline. Since we have no content data for the Adamic dataset, we run this baseline on the Kale, Cora, and CiteSeer datasets. We run 5 experiments per dataset, setting the amount of training data at 10%, 20%, 40%, 60%, and 80% of the full data, and the result of each experiment is averaged over 20 runs of 5-fold cross-validation.

Cora		CiteSeer	
Neural	726	HCI	304
Case	285	IR	532
Reinforce	214	Agents	463
Prob	379	AI	115
Genetic	406	ML	308
Rule	131	DB	388
Theory	344		

Table 3: Class distribution for the Cora and CiteSeer datasets.

Spectral Clustering Methods

Spectral methods have been used widely used to cluster data that can be represented as graph, by using the eigenvectors of the affinity matrix [17, 15, 7]. Spectral methods are simple to implement and can be used for any data where a distance measure can be derived between two data points. As we are interested in the amount of labeled data needed to achieve a certain classification accuracy, we would like to explore clustering methods where no labeled data is required.

We compare our algorithm against two spectral clustering methods: normalized cuts [17] and the spectral clustering algorithm presented by Ng et al. in [15]. In [17], the normalized cut criterion (measuring both the total dissimilarity and total dissimilarity between different groups) is used to partition a graph in two, and this can be done recursively to arrive at k partitions. In [15], the largest k eigenvectors of the graph's Laplacian matrix is used to project the data onto a k-dimensional space, and then k-means is used to cluster the points.

For both of these methods, we derive a undirected (symmetric) affinity matrix from the directed graph as input, where $A(i, j) = 1$ if there is a link from i to j or vice versa, and 0 otherwise. We test these two methods on the simpler, two-class datasets, the Kale and the Adamic datasets. When evaluating the clusters, we choose the assignment of labels to clusters that result in the highest accuracy.

Stacked Graphical Learning

We also compare our algorithm against a state-of-the-art, supervised, collective classification method that use both textual features and the link structure in predicting labels for network data: stacked graphical models [13]. Collective classification methods that utilizes the link structure has been to shown to significantly outperform methods that uses only content features [14, 13].

We rerun the experiments described by Kou and Cohen in [13], using $k = 1$ and 5-fold cross-validation. Although we are unable to achieve the exact same performance on either dataset as reported in the paper, possibly due to slight variation in the base learning algorithm or feature extraction, the accuracy results are either very close to (on the Cora dataset) or even better than (on the CiteSeer dataset) the results reported in [13].

To make a fair comparison, we run every experiment twice: once on the original Cora and CiteSeer datasets (marked with a * in Table 1) and once on the connected component versions that we used for our proposed algorithm. In addition to reporting results from our rerun of the stacked graphical model method, we also report results from the original [13] paper and results on the same dataset from another collective classification paper, *Linked-based Classification* by Lu and Getoor [14]. Again, note that the results we report directly form these papers are from the original Cora and CiteSeer datasets.

MultiRank Bootstrapping

We run our proposed algorithm on each of the four datasets, using the **at-least-n-per-class** seeding method described earlier, varying n. For the Kale dataset ,we set n=1, 2, 5, 10, 20, 40; for the Adamic dataset, we set n=1, 5, 10, 40, 80, 160; for the Cora dataset, we set n=1, 2, 5, 10, 20, 40; for the CiteSeer dataset, we set n=1, 2, 5, 10, 15, 40, 60. The largest n for the seeding method is chosen so that the resulting number of seeds will be less than the one third of the full dataset. The algorithm is only run once for each experiment because the result of MultiRank bootstrapping method is deterministic given that the seeding algorithm is deterministic.

4.1 Results and Discussions

The results of our algorithm compared against various baseline and competitive algorithms are shown in Figure 3 and Figure 4. The numbers shown in the charts are classification accuracies on the unlabeled instances. The algorithms are labeled on the figures as follows: **MRB** is the MultiRank bootstrap algorithm; **Clustering** is both the Spectral Clustering and Normalized Cuts algorithms – they have exact same performance on these datasets; **Kou** is the best result reported in [13]; **Kou-Rerun** is the result from our re-run of **Kou**; **Kou-Rerun-C** is our re-run of **Kou** using the connected component version of the dataset; **Lu** is the best result reported in [14]; **Content-NB** is the content-only Naïve Bayes using bag-of-words features; **Content-Kou** is the content-only baseline reported in [13]; and **Content-Lu** is the content-only baseline reported in [14].

4.2 Political Blog Datasets Results

The two spectral clustering methods have almost the exact same performance and they do extremely well on the Kale dataset, at 0.95 accuracy. However, when applied to the Adamic dataset they fail miserably. At first this seems strange, as the two datasets are quite similar except that one is much bigger. To further explore this phenomenon, we create synthetic datasets by *subsampling* the Adamic dataset: randomly draw n instances from the dataset and take the connected component from the resulting subgraph as input data. We start from $n = 400$ and increase n until the resulting graph is the same as the original dataset and observe the change in accuracy. Each subsampling is done 10 times and the accuracy and the resulting number of nodes and edges in the connected graph are averaged. From the experiment we see a sharp decline in accuracy at $n = 1200$ and we also note the decrease in graph density as the graph becomes larger: it may be that spectral clustering methods require more edges, or blog links, in order to perform well. To test this hypothesis, we randomly augmented the full Adamic dataset with additional edges between members of the same faction, so that the edge density would be the same as it was at above 0.9 accuracy; this is done 10 times and the results averaged. The resulting accuracy (0.909) from the augmented Adamic graph seem to support this hypothesis, but further analysis is required and is beyond the scope of this paper. Spectral clustering methods are often used in datasets where the graph is generated using carefully designed distance metric [20], and such graphs are often dense with weighted edges [17, 15]. Graph data with naturally occurring edges are often weighted much more sparse, which may prove to be more challenging for spectral clustering methods. The results from the experiment on the Ng et al. spectral clustering are shown in Table 4; the normalized cuts, thought not shown, produces nearly identical results.

n	Nodes	Edges	Density	Accuracy
400	262	1404	0.0205	0.953
600	430	2929	0.0158	0.947
800	608	5361	0.0145	0.911
1000	785	8812	0.0143	0.950
1200	957	12313	0.0134	0.869
1400	1140	16821	0.0129	0.695
Full	1222	19021	0.0127	0.520
Full+E	*1222*	*21321*	*0.0143*	*0.909*

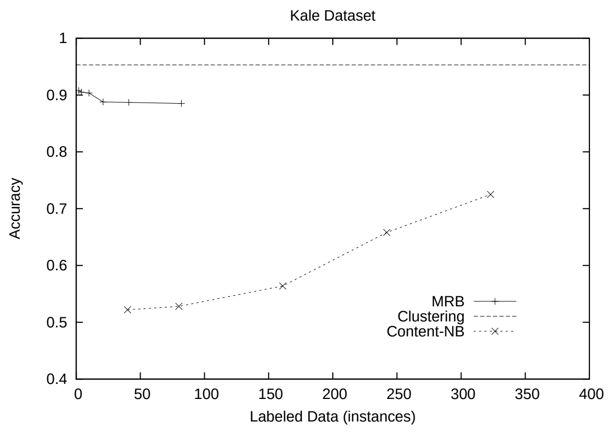
Table 4: Subsampling experiment on the Adamic dataset using the spectral clustering method presetned by Ng et al. Full+E is the full Adamic dataset augmented with additional intra-faction edges.

Our proposed algorithm, on the other hand, does well on both datasets, and surprisingly, it does so with a very few number of seeds. In fact, our proposed algorithm performs just as well with one seed per class – the minimum number of seeds required by the algorithm – as with more than a hundred seeds per class. In comparison, the content-only baseline reaches 0.725 accuracy only after using 80% of the data as labeled training instances.

On the two political blog datasets, our proposed method, using only a small set of labeled instances, is able to exploit the link structure and outperform a simple content-only baseline algorithm and is also shown to be more stable and scalable than the spectral clustering methods.

4.3 Citation Datasets Results

The scientific citation datasets are in many ways a more challenging dataset. Not only there are a larger number of classes and the class distribution is more varied, the graph is less dense and the ratio of intra-faction edges smaller (see Table 1) – properties that our proposed algorithm rely on to propagate labels from seed instances to the rest of the graph. However, the proposed method still outperforms the simple content-only baseline and shows competitive performance compared to the stacked graphical model method and other accuracy results reported in [14] and [13] that uses a good amount of training data containing both textual features and link structures.



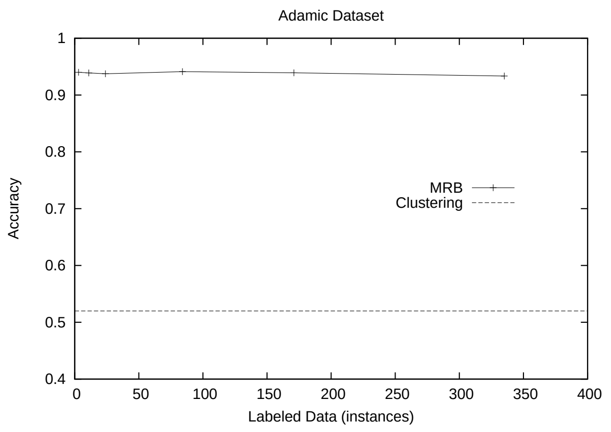
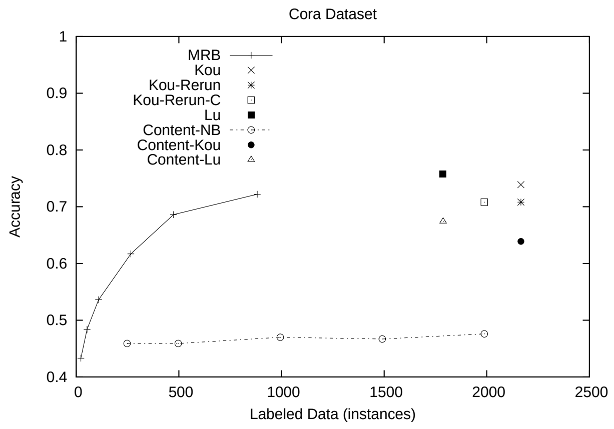


Figure 3: Results on the Kale and Adamic political blog datasets. The x-axis indicates the number of labeled instances used by the algorithm; the y-axis indicates the labeling accuracy.



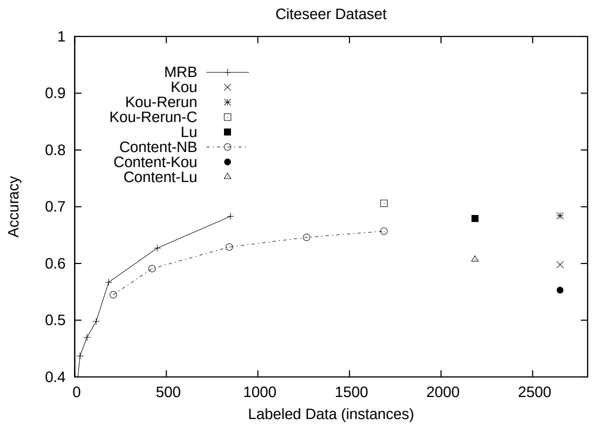


Figure 4: Results on the Cora and CiteSeer scientific paper citation datasets. The x-axis indicates the number of labeled instances used by the algorithm; the y-axis indicates the labeling accuracy.

Note that on both datasets, the accuracy of **Kou-Rerun-C** (the connected component dataset) is a little higher then **Kou-Rerun**. Since stacked graphical model is a collective classification algorithm, naturally it performs better with a more strongly connected dataset. In the same way we also expect the reported results from [14] and [13] to be a bit higher if they had used the connected component version of the dataset instead.

4.4 Scalability

Our proposed algorithm is very efficient in required memory and run-time speed. The memory required is the amount of memory required to store the graph plus $k * 2 * V$, where k is the number of classes and V is the amount of memory required to store ranking scores for each of the nodes in the graph. MultiRank, like PageRank, converges fairly quickly [16], usually within 50 iterations with a reasonable tolerance, and often only requiring one or two iterations. The bootstrapping algorithm is observed on the four datasets and synthetic datasets of similar sizes to converge around 10 iterations and no more than 30 iterations. Figure 5 displays run-time speed of the algorithm on subsamplings of the Adamic dataset, showing the run-time speed is linearly proportional to the number of edges. Each subsampling of the speed test is ran 5 times and the median is shown in the figure. The test was done on a single-processor 1.86 GHz Linux machine with 4 GB of memory. The program was implemented in Java.

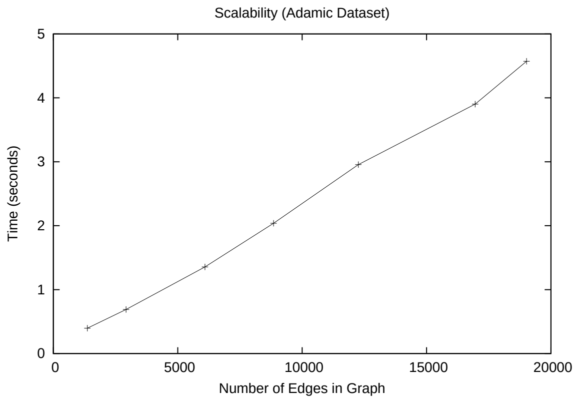


Figure 5: Scalability. The x-axis indicates the number of edges in the graph; and the y-axis indicates the time it took for the algorithm to converge in number of seconds.

5. CONCLUSIONS

We have proposed a semi-supervised learning method for data in graphs. By leveraging the homophily found in graph data, we use MultiRank to propagate faction- or class-specific PageRank, which in turn helps to propagate labels from seed instances when bootstrapped with simple edge and node labeling methods. This MultiRank bootstrapping method works given a small number of popular or authoritative seeds instances, ones that are arguably much easier to obtain labels for.

This MultiRank boostrapping method is tested on four datasets from two different domains and is found to outperform or is competitive with state-of-the-art spectral clustering methods and state-of-the-art, *fully supervised methods for learning in graphs. The method is also very efficient and scales to large datasets with run time linearly to the number of edges in the graph.*

6. FUTURE WORK

As pointed out in section 2.2, the STEP 1 and STEP 3 can easily be replaced with other classifiers that labels edges and nodes. Besides employing more sophisticated labeling algorithm based on ranking and confidence measures, a natural extension to the algorithm would be extending STEP 3 to use content features. Furthermore, bootstrapping the algorithm with a classifier that uses content features take away the one limitation of the proposed algorithm - that the input data need to be a connected graph.

7. REFERENCES

[1] *Nielsen Buzzmetrics, www.nielsenbuzzmetrics.com.*

[2] *S. Abney. Understanding the yarowsky algorithm.* Computational Linguistics*, 30(3):365–395, 2004.*

[3] *L. Adamic and N. Glance. The political blogosphere and the 2004 u.s. election: Divided they blog. In* Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem*, 2005.*

[4] *R. Andersen and K. J. Lang. Communities from seed sets. In* Proceedings of the Fifteenth International World Wide Web Conference (WWW 2006)*, 2006.*

[5] *W. W. Cohen. Improving a page classifier with anchor extraction and link analysis. In* Advances in Neural Information Processing Systems 15*, 2002.*

[6] *M. Craven, D. DiPasquo, D. Freitag, A. K. McCallum, T. M. Mitchell, K. Nigam, and S. Slattery. Learning to construct knowledge bases from the World Wide Web.* Artificial Intelligence*, 118(1/2):69–113, 2000.*

[7] *I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means, spectral clustering and normalized cuts. In* Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.*

[8] *G. W. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization of the web and identification of communities.* IEEE Computer*, 35(3):66–71, 2002.*

[9] *Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm.* Machine Learning*, 28(2-3):133–168, 1997.*

[10] *T. Haveliwala, S. Kamvar, and G. Jeh. An analytical comparison of approaches to personalizing pagerank. Technical report, Stanford University, 2003.*

[11] *A. Kale, A. Karandikar, P. Kolari, A. Java, T. Finin, and A. Joshi. Modeling trust and influence in the blogosphere using link polarity. In* Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007)*, 2007.*

[12] *J. M. Kleinberg. Authoritative sources in a hyperlinked environment.* Journal of the ACM*, 46(5):604–632, 1999.*

[13] *Z. Kou and W. W. Cohen. Stacked graphical models for efficient inference in markov random fields. In* Proceedings of the 2007 SIAM International Conference on Data Mining*, 2007.*

[14] *Q. Lu and L. Getoor. Link-based classification. In* Proceedings of the 20th International Conference on Machine Learning*, 2003.*

[15] *A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In* Advances in Neural Information Processing Systems 14*, 2002.*

[16] *L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.*

[17] *J. Shi and J. Malik. Normalized cuts and image segmentation.* IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.*

[18] *S. Tong and D. Koller. Support vector machine active learning with applications to text classification.* Journal of Machine Learning Research*, 2:45–66, 2001.*

[19] *D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In* Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 1995.*

[20] *X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.*