Classifying the Political Leaning of News Articles and Users from User Votes

Daniel Xiaodan Zhou, Paul Resnick, Qiaozhu Mei

School of Information, University of Michigan
105 S. State St, Ann Arbor, MI 48109 USA
{mrzhou, presnick, qmei}@umich.edu

Abstract

Social news aggregator services generate readers' subjective reactions to news opinion articles. Can we use those as a resource to classify articles as liberal or conservative, even without knowing the self-identified political leaning of most users? We applied three semi-supervised learning methods that propagate classifications of political news articles and users as conservative or liberal, based on the assumption that liberal users will vote for liberal articles more often, and similarly for conservative users and articles. Starting from a few labeled articles and users, the algorithms propagate political leaning labels to the entire graph. In cross-validation, the best algorithm achieved 99.6% accuracy on held-out users and 96.3% accuracy on held-out articles. Adding social data such as users' friendship or text features such as cosine similarity did not improve accuracy. The propagation algorithms, using the subjective liking data from users, also performed better than an SVM based text classifier, which achieved 92.0% accuracy on articles.

Introduction

In U.S. politics, opinions on a variety of issues involving taxes, the role of government, domestic policy, and international relations are substantially though imperfectly correlated with each other and with party affiliation and with an overall self-identification as liberal or conservative. Thus, classifying people, media outlets, and opinions expressed in individual articles as liberal or conservative conveys meaning to most people.

The liberal vs. conservative classification scheme has its critics (e.g., Klein and Stern 2008). One reason is that the single dimension cannot capture cases such as the libertarians who align with liberals on some issues and conservatives on others. Another is that definitions of conservative and liberal are vague and inconsistently applied. Moreover, many political news and opinion articles express a mixture of conservative and liberal ideology. Thus, not everyone will agree about the correct classification of particular items, or even the correct classification of their own stance.

Despite some fuzzy boundaries, however, the one-dimensional classification scheme persists in our discourse, and many people, articles, and news sources fit clearly into one category or the other. The ability to classify blogs as liberal or conservative enabled Adamic and Glance (2005) to analyze patterns of inter-linking between them. It also served as the basis for investigating people's preferences for difference mixtures of reinforcing and challenging articles in a news aggregator (Munson and Resnick, 2010) and for sorted or annotated displays (Gamon et al 2008; Oh et al 2009; Munson and Resnick 2010).

Here we consider the problem of automatically classifying people and items as liberal or conservative. Most people do not wear "liberal" or "conservative" labels, either in person or on their on-line profiles. And an article's political position may not always be easy to glean from surface analysis of its text. Our inspiration is that a few manually coded labels might be propagated to other people and articles, since liberal people are likely to endorse liberal articles, and similarly for conservative people and articles.

There is a naturally occurring source for the data needed to propagate, a large pool of subjective "votes" for individual articles. Digg.com, a popular social news aggregator, has links to political stories from both blogs and news sites. Users can "digg" stories they like. Individual diggs are visible on the website and accessible through a public API.

Figure 1 illustrates the potential propagation of a few initial labels. Following the current convention in the U.S. media, we color-code conservative as red and liberal as blue. The links in the graph represent diggs, the votes by users for particular articles. The articles dugg by the red user can be colored red. Similarly, the people who dugg the blue articles can be colored blue. In subsequent rounds,

Copyright © 2011, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

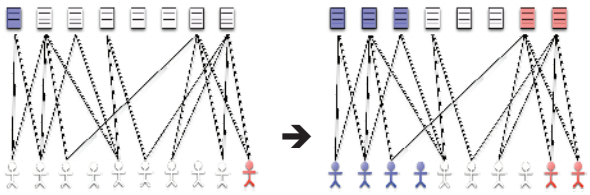

Figure 1. Intuition for the propagation process

those colorings can be propagated still further. Some people or items may lie in the middle, not clearly liberal or conservative. We will think of items that cannot be decisively labeled during the propagation process as "gray." We consider three alternatives for the semi-supervised learning propagation algorithm, as well as a variety of data sources that can affect the structure of the graph and the initial seed labels, and then evaluate classification accuracy.

Related Work

Literature from both political science and computer science has studied the problem of classifying political positions from texts. One line of work used word frequencies, Bayesian statistical models, and topic models (Laver et al 2003, Martin and Vanberg 2008, Lin 2006, Lin et al 2008, Monroe et al 2008, Slaping and Proksch 2008). Another line of work focused on using SVM with optimization of text feature selection (Jiang and Argamon 2008, Oh et al 2008, Yu et al 2008, Hirst et al 2010), as well as complementing with sentiment analysis (Durant and Smith 2006, Mullen and Malouf 2006, Malouf and Mullen 2007).

Rather than using text features, Efron (2004) used co-citations from Google search to classify political blogs. Park et al (2011) used a panel of users and their comments' sentiments to predict the political leaning of articles.

Semi-supervised learning approaches utilize a large amount of unlabeled data in the classification process, and thus achieve good classification performance even if only a small set of labeled examples are available. A particular family of semi-supervised classification algorithms, which we draw on, cast the classification task as a process of label propagation in the graph structure of labeled and unlabeled data (Zhu et al. 2003, Zhou et al. 2004).

The closest study to ours is Lin and Cohen (2008), who used a semi-supervised learning algorithm called "multirank" to classify political blogs using the HTML links between blog stories. We use different algorithms and find that, once we propagate via diggs, adding propagation via blog-to-blog HTML links decreases accuracy.

Semi-Supervised Learning Algorithms

Problem Formulation

From Digg, we have a set of stories V_{story}, users V_{user}, and the diggs from users to stories as approval votes, denoted as $E_{digg}=\{(i,j)\}$, where $i \in V_{user}$ and $j \in V_{story}$. We also have 1) other types of nodes, V_{extra}, such as the domain names of blogs, and 2) other types of links, E_{extra}, which will be discussed in the Datasets section. Let $V=V_{user} \cup V_{story} \cup V_{extra}$, $E=E_{digg} \cup E_{extra}$, and then we can construct a graph $G = <V, E>$. Edges $e \in E$ are undirected because the color label of either node of e should propagate to the other.

From G, we construct the symmetric affinity matrix W, where $W_{ij}=1$ if $(i,j) \in E$, or 0 otherwise. Define D as the diagonal matrix of degrees of the nodes. That is, $D_{ii}=\sum_j W_{ij}$.

Denote the initially labeled nodes as L ($L \subset V$). Let $C=\{red, blue, gray\}$ be the set of category labels. For each $i \in L$, we have an initial label $c_i \in C$. Let $T=V-L$ be the set of unlabeled nodes (i.e., nodes that need to be classified). The labels will be divided into a training set $L_{training}$ and a testing set $L_{testing}$, where $L = L_{training} \cup L_{testing}$. The goal of our algorithms is to use the initial labels from $L_{training}$ and assign a label $c'_j \in C$ as the classification output to each node $j \in T \cup L_{testing}$.

We also introduce a $|V| \times 2$ matrix Y, where the 2 column vectors are indexed as *R* for *red* and *B* for *blue*. $Y_{iR}=1$ if $i \in L_{training}$ and $c_i=red$; $Y_{iB}=1$ if $i \in L_{training}$ and $c_i=blue$; 0 for the other elements.

Note that any *gray* labels in L are simply ignored for the purposes of training. Intuitively, we do not treat *gray* as an independent classification category but rather as nodes for which the classification is mixed or uncertain. Thus, we do not propagate *gray* classifications. However, our algorithms could still output *gray* for borderline items.

Random Walk with Restart (RWR)

Our first semi-supervised algorithm, shown in the box, is based on the popular "random walk with restart" model (Grinstead and Snell 1997). After convergence, F* is the stationary distribution specifying the probability that a random walk with restart at Y' will be at each of the nodes.

In our case, the restart matrix Y' has two columns, corresponding to two separate random walks, leading to two stationary distributions. The first walk, with the *R* column of Y', restarts from only the *red* labeled nodes, and the F^*_{iR} scores indicate the stationary probabilities restarting only from the *red* nodes. Similarly, the second walk, with the *B* columns of Y', yields the F^*_{iB} scores. Intuitively, $F^*_{iR} > F^*_{iB}$ means a walk starting from the *red* nodes is more likely to reach node i than a walk starting from the *blue* nodes, and thus we should label i as *red*. When F^*_{iR} is close to F^*_{iB}, there is not a clear classification and we label them *gray*. Two threshold parameters θ_R and θ_B control how big the ratio of F^*_R and F^*_B has to be in order to make a *red/blue* classification. Increasing the thresholds leads to output of more *gray* labels for borderline nodes.

Local Consistency Global Consistency (LCGC)

Our second semi-supervised learning algorithm, also shown in the algorithm box, follows the "local consistency

global consistency" classification algorithm proposed in Zhou et al (2004). Note that the iteration process differs from RWR only in normalization factors for S and Y.

The intuition behind the algorithm is to optimize for two conditions: a) the labels assignment should not change too much between nearby nodes ("local consistency"), and b) the initial labels assignment should not change too much after propagation ("global consistency"). The parameter α controls the trade-off between the two objectives.

Absorbing Random Walk (ARW)

The "absorbing random walk" model is different from the original random walk model in that it has "absorbing states" without outgoing links (Grinstead and Snell 1997). Let $A \subset V$ be a set of absorbing states. After convergence, each node $i \in V$ has a probability score $P(a|i)$ for each absorbing state $a \in A$, indicating the probability that i would eventually be absorbed into a. We have $\sum_{a \in A} P(a|i) = 1$, and $P(a|a)=1$. The use of absorbing random walks in classification is closely related to Zhu et al (2003).

In our case, we didn't use the labeled dataset L as the absorbing states, because the initial labels are not 100% accurate and should be allowed to change color during propagation. Therefore, we added two new absorbing states a_{red} and a_{blue} to V, and added directed edges $\{(i, a_{red})\}$ and $\{(j, a_{blue})\}$ if $i,j \in L$ and c_i=*red*, c_j=*blue*.

In step 1, $k \in [0, \infty)$ is the weight of the newly added edges $\{(i, a_{red})\}$ and $\{(j, a_{blue})\}$. In step 2, weights on edges (including the new edges) are normalized to create probability distributions over transitions from each node. The normalized matrix decomposes into Q, which gives probabilities of transitions to edges in the original graph, and Y', which gives probabilities of transitions to the absorbing states a_{red} and a_{blue}. After the random walk converges, F^*_{iR} and F^*_{iB} are the probabilities for each $i \in V$ eventually getting absorbed in a_{red} and a_{blue} respectively. In steps 4 and 5, we classify the nodes using the "class mass normalization" method suggested by Zhu et al (2003).

Intuitively, this algorithm classifies $i \in V$ as *red* if it has a much higher probability to be absorbed in a_{red}, and the same for *blue*.

Algorithm 1: RWR

Input: W, D, Y
Algorithm:

1. Construct the transition matrix $S=D^{-1}W$
2. Construct $Y'=(D_Y^{-1}Y^T)^T$, where D_Y is a diagonal matrix with $D_{Y\text{-}ii}=\sum_j Y_{ji}$.
3. Iterate $F(t+1)=(1-\alpha)SF(t)+\alpha Y'$, where F(t) is a $|V|\times 2$ matrix, $F(0)=Y'$, and α is a tunable teleport factor. Let F* denote the limit of the sequence {F(t)}

Classification: Label $i \in V$ as:

a. *red* if $F^*_{iR}/F^*_{iB} > \theta_R$
b. *blue* if $F^*_{iB}/F^*_{iR} > \theta_B$
c. *gray* otherwise

(θ_R and θ_B are tunable threshold parameters.)

Algorithm 2: LCGC

Input: W, D, Y
Algorithm:

1. Construct the matrix $S=D^{-1/2}WD^{-1/2}$
2. Iterate $F(t+1) = (1-\alpha)SF(t)+\alpha Y$, where α is a tunable parameter in (0, 1), and $F(0)=Y$. Let F* denote the limit of the sequence {F(t)}

Classification: *the same as in RWR*

Algorithm 3: ARW

Input: W, D, Y
Algorithm:

1. Construct $W'=\begin{vmatrix} W & kY \\ 0 & I \end{vmatrix}$, where $k=(1-\alpha)/\alpha$, $\alpha \in (0,1]$ is a tunable parameter; I is a 2×2 identify matrix.
2. Construct $S'=D'^{-1}W'$, where D' is a diagonal matrix with $D'_{ii}=\sum_j W'_{ij}$. S' has form $\begin{vmatrix} Q & Y' \\ 0 & I \end{vmatrix}$.
3. Iterate $F(t+1)=Y'+QF(t)$, where $F(0)=Y'$. Let F* denote the limit of the sequence {F(t)}.
4. Let $D_P=\begin{vmatrix} P(a_{red}) & 0 \\ 0 & P(a_{blue}) \end{vmatrix}$, where $P(a_{red})=\sum_{i \in V}(F^*_{iR}/|V|)$, $P(a_{blue})=\sum_{i \in V}(F^*_{iB}/|V|)$. $P(a_{red})+P(a_{blue})=1$.
5. Calculate $F^{*\prime}=(D_P^{-1}F^{*T})^T$

Classification: *the same as in RWR using F*'*

Datasets

Graph Structure

Diggs as votes from user nodes to story nodes (V_{story}, V_{user}, E_{digg}). This is the primary dataset for our study. With the Digg API, we harvested 480,932 stories and their 6,298,104 diggs from 2 categories, 'political news' and 'political opinions', from 2009-5-25 to 2010-8-11. That is an average of 1,083 and 14,185 new stories and diggs each day, with an average of 13 diggs per story. Our study only used the $|V_{story}|$=84,433 "popular" stories that received more than 10 diggs, and the $|V_{user}|$=74,844 "frequent" users who submitted more than 5 diggs. Those "frequent" users made a total of $|E_{digg}|$=5,216,273 diggs to the "popular" stories. The median degree for items (number of frequent users in the dataset who dugg the popular item) was 22. Note that each user could submit an unlimited number of diggs, but only one per story. The largest connected component covers 99.98% of the nodes.

Domain source links (V_{source}, E_{source}). For each story in V_{story}, we have its source domain name. For example, stories posted on HuffingtonPost.com would all share the same domain name. We hypothesize that stories with the

same domain name would share the same political leaning. This is clearly true for political blogs like HuffingtonPost.com, but not necessarily true for NYTimes.com or Blogspot.com. We created domain source nodes V_{source} and edges $E_{source}=\{(i,s): i \in V_{story}$ was posted on source domain $s \in V_{source}\}$.

Links from blogs to stories ($V_{link\text{-}to}$, $E_{link\text{-}to}$). Munson et al (2008) created a dataset consisting of political blogs and their HTML links to other stories. We hypothesize that stories linked to by the same blog would have the same political leaning as the blog. We created $V_{link\text{-}to}$ for the 396 political blogs that linked to any stories in V_{story}, and undirected edges $E_{link\text{-}to}=\{(i,j): i \in V_{link\text{-}to}$ HTML links to $j \in V_{story}\}$. $|E_{link\text{-}to}|=17,372$, which is 21% of $|V_{story}|$.

User friendship links (E_{user}). Digg.com allows users to mark other users as friends, by mutual consent. We hypothesize that users who are friends on Digg.com will tend to share the same political leaning. Using the Digg API, we harvested a total of 2,247,591 user-user friendship links, among which 755,303 are between $u \in V_{user}$. We created $E_{user}=\{(i,j): i,j \in V_{user}$ are friends on Digg.com$\}$.

Story similarity links (E_{story}). Using the Digg API, we obtained the title and a short text snippet for each story, which was used to calculate text similarity between each story pair. We hypothesize that stories that have similar text would also share similar political leaning. We used Apache Lucene to calculate text similarity. We considered only terms that appeared in at least 5 stories but not more than 40% of the total stories. For each story, we selected its 20 terms with the highest tf*idf scores and used them to calculate a cosine similarity with each other story. If i was one of the 10 stories with highest similarity to j, and also j was one of the 10 most similar to i, an undirected edge (i, j) was added to E_{story}. $|E_{story}|=107,961$, so each story had an average of 1.3 text similarity links.

Labels

Users identified in a news article ($L_{user\text{-}reported}$). A news article[1] reported a group of conservative Digg users who created a Yahoo group called the "Digg Patriots" and self-organized themselves to deliberately bury liberal stories on Digg. The article identified 106 conservative users of the group. It also identified 44 liberal users on Digg who were their primary targets (i.e., stories suggested to Digg by those 44 liberal users were voted down). Digg.com has purged some members of the "Digg Patriots" from the system to prevent manipulation. But we still have 104 labeled Digg users, 68 *red* and 36 *blue*. We denote these labeled users as $L_{user\text{-}reported}$. These users dugg 69,785 (83%) of the stories in V_{story}.

Labeled blogs (L_{blogs}). From five sites that classify blogs[2], we compiled 1,635 blogs tagged as conservative or liberal. Of these, only 240 (15%) had any stories in V_{story}. Those blogs form a subset of V_{source} that we call L_{blogs}. 25,643 (30%) of the stories in V_{story} were from one of these labeled blogs.

More labeled blogs ($L_{link\text{-}to}$). For the 396 political blogs in $V_{link\text{-}to}$, Munson et al (2008) also labeled them as liberal or conservative. We used them as $L_{link\text{-}to}$, which overlapped but was distinct from the L_{blogs} above.

Manually coded stories from Amazon Mechanical Turk (L_{mturk}). We randomly selected 1000 stories from V_{story} and posted them on AMT. For each story, we accepted 6 ratings (3 from self-identified liberals and 3 from self-identified conservatives), at the cost of 3 cents per rating and a $1-$2 weekly bonus to the most productive turkers. We collected the ratings from 2010-7-8 to 2010-8-22, with 50-500 incoming ratings per week. 41 turkers coded at least one story, and 13 (8 liberals and 5 conservatives) coded more than 50 stories.

For quality control purposes, we required the turkers to pass a qualification test with 9 correct answers out of 10 questions: 5 questions on basic political knowledge (e.g., who was the Republican candidate in the 2008 presidential election?) and 5 questions on the real coding tasks to test their understanding of the coding guideline. They also had to meet the following criteria: a) located in the US, b) > 90% acceptance rate on other AMT tasks, and c) complete our survey on their political leaning

We also randomly inserted verification questions (e.g., "1+4=?") into 100 stories, and got correct answers from all turkers who encountered them. The turkers spent an average of 63 seconds on each story. The Fleiss inter-rater reliability score was 0.53, a "moderate agreement" (Landis and Koch 1977).

The limited inter-rater reliability suggests that there is not universal agreement about the liberal-conservative categories and that they apply more clearly to some stories than others. We considered those stories where all 6 turkers agreed to be clear examples. There were 73 red and 234 blue stories in our labeled dataset L_{mturk}. In the Extensions section we return to consideration of stories where raters were not unanimous.

Manually coded users ($L_{user\text{-}coded}$). We selected 220 Digg users from V_{user} who had made more than 15 comments, with more than half of their most recent comments on political stories. Then, we took a snapshot of their 15 most recent comments, and hired 2 undergraduate students to code them into *red*, *blue* and *gray* based on the political leaning inferred from the 15 comments.

Before the coding process, we trained the coders to follow coding guidelines. For quality control purposes, we inserted six known Digg users from $L_{user\text{-}reported}$ into the 220 user pool, and both coders correctly classified them. When both coders felt confident enough to assign a *red* or *blue* label, their agreement was 94.5% and their Cohen's kappa score was 0.89. We took the 69 red users and 62 blue users that both coders agreed upon as clear examples, and formed dataset $L_{user\text{-}coded}$.

[1] http://goo.gl/ipQn

[2] They are: blogcatalog.com, blogarama.com, httpetalkinghead.com, blogs.botw.org, and wonkosphere.com

Evaluation

We organized our evaluation process into 4 steps. Due to limited space, we only document the detailed optimization process for the RWR algorithm, and simply report the results for the other 2 algorithms in the first 3 steps. In the last step, we compared the three algorithms. For all 4 steps, we used 10-fold cross validation, repeatedly holding out one tenth of the nodes with known labels for testing.

The primary measurement was "accuracy", averaged across the 10 folds of cross validation. Let $O_{testing} \subset O$ be the output for $i \in L_{testing}$. Let $O^*_{testing}$ be the subset of $O_{testing}$ that are correctly classified.

$$\text{Accuracy} = \frac{|O^*_{testing}|}{|O_{testing}|}$$

Step 1: Optimizing Parameters

We were not confident on the usefulness of the extra structural datasets beyond the diggs, nor of two of the label datasets, L_{blogs} and $L_{link\text{-}to}$. Thus, to tune the algorithm parameters, we used the limited network G'=<V',E'>, where V'= $V_{story} \cup V_{user}$, E'= E_{digg}, and labeled data L'=$L_{user\text{-}reported} \cup L_{user\text{-}coded} \cup L_{mturk}$. On average, for nodes in a cross-validation test set, the shortest path from a node in the training set is 2.55, suggesting that a small but non-trivial amount of propagation will be necessary to propagate labels to nodes in the test set.

First, we optimized the teleport factor α for RWR, fixing θ_R and θ_B at 1.0. Fig. 2(a) shows that accuracy was not sensitive to α in the range 0.1 to 0.7. The optimal was α=0.3, yielding accuracy 94.8%. For the LCGC and algorithms, the optimized α was 0.3 and 0.1 respectively. We used these values for the rest of the evaluation.

Since our labeled datasets include only definitively labeled items (*red*s and *blue*s, but no *gray*s), overall accuracy will be optimized only when all items are assigned a *red* or *blue* label definitively. Not surprisingly, then, holding θ_R=1.0 the optimal value for θ_B was also 1.0, and vice versa. Thus, we assign red labels whenever $F^*_R > F^*_B$ and blue labels when $F^*_B > F^*_R$.

However, θ_R and θ_B could be used to trade off precision and recall rather than simply optimizing for overall accuracy. Precision and recall for *red* are defined as follows (for *blue* they are defined analogously):

$$\text{Precision}_R = \frac{|O^*_R|}{|O_R|}, \text{Recall}_R = \frac{|O^*_R|}{|L_{testing,R}|}$$

In the formula, O_R is a subset of $O_{testing}$ that are *red*. O^*_R is the subset of $O^*_{testing}$ that are correctly classified as *red*. $L_{testing,R}$ is the set of *red* nodes in the initial testing set.

The results are shown in Table 1. At $\theta_R=\theta_B$=1.0, *red* had higher recall and *blue* had higher precision. That means our algorithm tended to over-classify nodes as *red*.

Table 1. High recall for red; high precision for blue

	Precision	Recall
Red	88.0%	99.7%
Blue	99.6%	92.1%

To trade off precision and recall, we could adjust the θ_R and θ_B threshold parameters. In general, as we increase θ_R, fewer things are classified as *red* and more are left as *gray*, increasing precision but decreasing recall for red items, and similarly for θ_B. We have similar results for LCGC and ARW algorithms, too. We will return to the θ parameters in the Extensions section, where items that Turkers did not agree on are labeled *gray* for the purpose of evaluation.

Step 2: Evaluating Source and Link-to Relations from Labeled Blogs

In this step, we evaluated the usefulness of the two labeled sets of blogs, L_{blogs} and $L_{link\text{-}to}$. We added nodes for the blogs in L_{blogs} and edges from them to stories that appeared in those blogs, and added nodes for the blogs in $L_{link\text{-}to}$ and edges for their HTML links to stories. Table 2 shows the effects on accuracy. We get similar results using LCGC and ARW algorithms. Since the labels (and nodes and edges) associated with L_{blogs} were useful, we included them in the baseline for assessments in step 3. But we excluded $L_{link\text{-}to}$ and the associated $V_{link\text{-}to}$ and $E_{link\text{-}to}$ because they did not increase accuracy. The optimized labeled dataset was then $L^* = L_{user\text{-}reported} \cup L_{user\text{-}coded} \cup L_{mturk} \cup L_{blogs}$.

Table 2. Blog sources are useful; not blog links

		Add $L_{link\text{-}to}$?	
		No	Yes
Add L_{blogs}?	No	94.8%	92.1%
	Yes	**95.4%**	92.9%

Step 3: Evaluating Structural Datasets

In this step, we evaluated the usefulness of the three extra structural datasets V_{source}, E_{source}, E_{user}, and E_{story} that added additional nodes and links without adding any additional labels. Prior to this step, we used E=E_{digg}, and the weight w_{ij} for each (i,j)∈E was set to 1. Since we have more than 5 million links in E_{digg} and the number of extra links, $|E_{domain} \cup E_{user} \cup E_{story}|$, is only 10% of $|E_{digg}|$, adding the extra links to E with the same weight as E_{digg} would not have much effect. Therefore, we also optimized the weight for the extra links by varying their values from 1 up to 100.

First, we added V_{source} and E_{source} to G. Note that L_{blogs} was already included in G, so the only additional source nodes added were those not associated with known labeled blogs. Fig.2(b) shows that their addition, with weight 1, decreased accuracy from 95.4% to below 95%. Increasing the weight of edges e∈E_{source}, including the edges from the labeled blogs, increased accuracy, up to an optimal weight of 50, which yielded accuracy of 96.6% .

Second, we added E_{user} to the original G (without V_{source} and E_{source}). As shown in Fig.2(c), adding the friendship links reduced accuracy. Even though accuracy peaked at weight=10, it was still lower than the previous optimum.

Finally, we added E_{story} to G. As shown in Fig.2(d), accuracy dropped steadily as the weight of e∈E_{story}

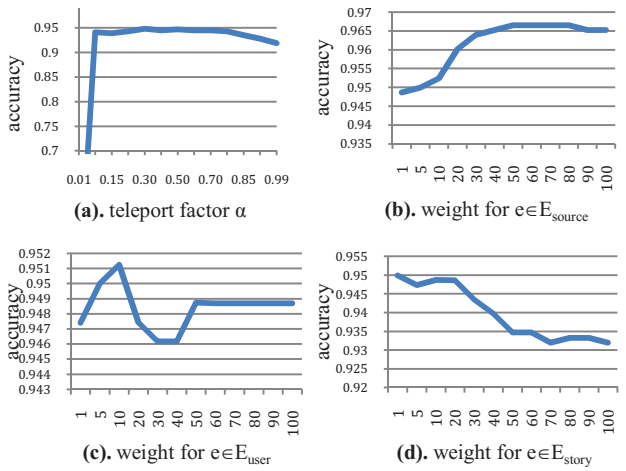

Figure 2. (a) optimal α = 0.3 (b) V_{source}/E_{source} helpful, with optimal weight = 50 (c) E_{user} not helpful (d) E_{story} not helpful.

increased, and it was always lower than the previous optimum of 95.4% without E_{story}.

Thus, the optimal graph structure G*=<V*,E*> has V*= $V_{story} \cup V_{user} \cup V_{source}$, and E*= $E_{digg} \cup E_{source}$ with the weight of $e \in E_{source}$ equal to 50. We found similar results for LCGC and ARW algorithms too, where E_{source} improved accuracy, but E_{user} and E_{story} did not.

Step 4: Semi-Supervised Algorithms Comparison

Next, we compared the performance of the three semi-supervised learning algorithms, using the optimized parameters from step 1 and the optimized L^* and G^* from steps 2 and 3. In addition to the overall accuracy, we also show accuracy for stories and users separately in Table 3.

Table 3. Algorithms comparison

	Accuracy (overall)	Accuracy (stories)	Accuracy (users)
RWR	96.6%	95.4%	99.5%
LCGC	96.9%	95.6%	100%
ARW	97.3%	96.3%	99.6%

Discussion

Overall, the results are quite promising, suggesting that a relatively small number of seed people and stories that are clearly liberal and conservative, together with a large number of people to item votes, can be used to classify, with high precision and recall, the other people and items that are clearly liberal or conservative. The three algorithms all had very high accuracy, with the absorbing random walk performing the best of the three.

Precision was higher for *blue* classifications, but true *red*s had higher recall. We suspect that this is because Digg is quite skewed in favor of liberal stories, but the color distribution in our training data is more balanced.

The liberal/conservative labels of source blogs, together with links from source blogs to the items that appeared in those blogs, proved to be useful inputs to the propagation algorithms. A closer look at the results of the optimized RWR algorithm reveals that 98.5% of stories in L_{blogs} had the same political leaning as their source blogs. We suspect that congruence, together with reasonably high quality classifications of the blogs, made those labels and links useful for classification.

Propagating liberal/conservative labels from blogs through their HTML links to stories, however, decreased overall classification accuracy. Although Adamic and Glance (2005) found that HTML links between blogs that have different ideologies are rare, such links are frequent enough to make propagating labels over those links misleading. According to the RWR algorithm's classification, only 76.5% of HTML links led from blogs to stories that had the same political leaning.

Adding nodes for website domains (including the non-labeled ones) with links to stories that appeared on those domains was helpful for the classification. This implies that most Digg stories posted on the same website indeed share the same political leaning. Classification improved most when the links from sites to stories were given weight equal to fifty individual diggs. We suspect that if we had a dataset with many more diggs per story, the optimal weight for these site-story links might be even higher.

Adding links between declared "friends" decreased classification accuracy. The classification results from our algorithm suggest that only 63.3% of the 755,303 friendship pairs share the same political leaning. One possible explanation is that since Digg is not for political news only, friendship on Digg is not necessarily based on political preference but perhaps on other non-political factors such as the same hobbies or locations.

Adding links between stories with textual similarities also decreased classification accuracy. One plausible explanation is that stories that have similar topics and content features do not necessarily share the same opinions. For example, "thumbs up to healthcare reform" has similar text to "thumbs down to healthcare reform", but clearly they have the opposite political leanings. Another reason was that we didn't have the full text of the stories, which prohibited further optimization on the text similarity links. It could be, however, that more sophisticated text comparisons, perhaps based on topic modeling and sentiment analysis, would provide better inputs to our graph propagation algorithms.

Comparison to SVM

We compared the semi-supervised learning algorithms to the supervised learning algorithm, SVM, which was the most frequently used approach in prior studies on political leaning classification. Our semi-supervised learning algorithms outperformed the SVM algorithm.

We tried three versions of the SVM classifier. All text features were generated using Apache Lucene, and we used the SVM-light[3] application to run the algorithm.

[3] http://svmlight.joachims.org/

In the first version, we simply generated the uni-gram text features from each story's title and text snippet, and ran SVM. In the second version, we followed the optimization process in (Oh et al, 2009): for the text features, we used the combination of uni-gram, bi-gram and tri-gram, and then used χ^2 feature selection that selected 5,440 out of 3,079,759 features with $p<0.1$.

The first two versions only worked for stories that have text features. In the third version, we first ran SVD on W to generate 6 major components for both stories and users, and then used them as 6 features the classifier.

The result is in Table 4. SVM with feature selection worked quite well, achieving 92% accuracy for stories. However, it has two limitations. First, after feature selection, SVM was not able to classify 9.8% of the stories that didn't have any selected features. To be able to classify those stories, we would have had to add more features, possibly noisy ones, which would have driven down accuracy. Second, a text classifier was not able to classify the users (except for using features from SVD) that didn't have any text features.

Table 4. Result of SVM

	Accuracy (overall)	Accuracy (classified stories)	Accuracy (users)
All features	N/A	76.2%	N/A
χ^2 feature selection	N/A	**92.0%**	N/A
SVD features	81.4%	87.6%	76.0%

Extensions

Online updating. Running the algorithm with full iteration would be too time-consuming in an online setting each time a new digg arrived. For the RWR algorithm, we developed a simple online algorithm to classify stories and users using 1-level propagation of previously-computed RWR scores $F^*_i=(F^*_{iR}, F^*_{iB})$ through the complete set of diggs, including those that had newly arrived:

$$F_j = \sum_i \frac{F^*_i}{\text{degree}(i)}, \text{where } (i,j) \in E$$

Using the two automatically collected datasets $L_{training}= L_{blogs} \cup L_{user\text{-}reported}$ to generate the pre-computed F^*_i scores with RWR, we evaluated the 1-level propagation algorithm using the two manually coded datasets for testing, $L_{testing}=L_{mturk} \cup L_{user\text{-}coded}$. Compared to the first row of Table 5, accuracy of 1-level propagation was only a little lower. We conclude that it would be reasonable to use 1-level propagation online and periodically re-compute the stationary F^*_i scores. We leave it to future work to develop similar approximations of the LCGC and ARW algorithms.

Table 5. Result of 1-level propagation on , with RWR

	Accuracy (overall)	Accuracy (stories)	Accuracy (users)
$L_{training}=L_{domain} \cup L_{user\text{-}reported}$ $L_{testing}=L_{mturk} \cup L_{user\text{-}coded}$	95.2%	94.1%	97.7%
1-level propagation	93.8%	92.5%	96.9%

Using *gray* labels in testing set. So far, we have showed the semi-supervised learning algorithms achieved high accuracy on clearly labeled *red* and *blue* items. Some stories and users, however, do not fit cleanly into either category. In some contexts, either *red* or *blue* labels for ambiguous items would be acceptable. In others, however, it would be better to mark such ambiguous items as *gray*, and classifying them as either *red* or *blue* would be considered erroneous. In that case, excluding *gray* items from the calculation of error rates, as we have done, would lead to overestimates of the precision of the classifications.

We have conducted some preliminary analysis of how the algorithms would perform if classifications of *gray* items as *red* or *blue* counted as errors. From the 1000 Mechanical Turk stories, we defined the rest of the 653 stories (excluding 40 broken link stories) that were not in L_{mturk}, those without unanimous ratings from turkers, as *gray*. Adding the new *gray* labels to the testing set, we got the optimal threshold parameters as $\theta_R=1.6$ and $\theta_B=1.45$ for RWR, which switched some of the *red* and *blue* classifications to *gray*. Accuracy overall dropped to 72.4%. Accuracy for the clearly labeled *red* and *blue* items dropped to 89.9% with the new threshold parameters.

For comparison, we used two binary SVMs (one classifies *red* vs. *not-red*, the other classifies *blue* vs. *not-blue*) to classify *red* (as *red* and *not-blue*), *blue* (as *blue* and *not-red*), and *gray* (otherwise) using the new testing data. Accuracy was 85.7%, higher than RWR. Note that SVM could not classify 13% of the stories that did not have any selected features, and we simply labeled them as *gray*. This helped SVM because there are many *gray* labels in the testing set.

With a slightly different definition of true *red*, *blue*, and *gray*, the results turned out differently. For Mechanical Turk stories, we defined *red* as any story having >2/3 *red* ratings from the turkers, *blue* as having >2/3 *blue* ratings, and *gray* for the rest, which resulted in 490 *blue*, 203 *red*, and 267 *gray*. Using these new data in the testing set for RWR, we got the optimal $\theta_R=1.15$, $\theta_B=1.1$, and overall accuracy 74.8%. For clearly labeled *red* and *blue* items, we still have 95.6% accuracy using the new threshold parameters. The SVM algorithm got accuracy 73.9%, now slightly lower than RWR.

Conclusion and Future Work

To conclude, in the paper, we discussed 3 semi-supervised learning algorithms to propagate political leaning of known articles and users to the target nodes. The best algorithm achieved 97.3% accuracy on users and stories that people agreed were clearly liberal or conservative, noticeably better than the most commonly used SVM algorithm.

The biggest challenge for future research is to improve the algorithm's ability to separate clearly liberal and conservative items from those that do not fit neatly into

either category. This will require further methodological innovation as well in developing evaluation schemes for situations where the ground-truth is not crisply defined.

Another interesting direction for future work is to try to understand and characterize the properties of datasets for which the different propagation algorithms will perform better or worse, possibly with an axiomatic approach.

We note that the propagation algorithms gained accuracy with the addition of datasets such as domain source links, where the linked items tend to have high correlation in their labels, but lost accuracy with the addition of datasets such as friendship links and HTML links where the correlation was lower. We therefore discarded those datasets. Clearly, this is not optimal, since even a positive correlation much less than 1 in principle provides some information. Future research should find ways to make use of these noisy datasets rather than discarding them entirely.

Another limitation of the propagation algorithms is that they require interactions between stories and users (i.e., diggs). For unpopular articles not covered in social news sites such as Digg, our algorithm won't be able to classify them. However, the advantage of these algorithms is that they do not require much training data. This is complementary to SVM, which requires lots of training data, but does not require user-story votes. Therefore, one idea is to use the propagation algorithms to generate many labeled data with high accuracy, and then feed this data to train SVM, and then use the well-trained SVM model to classify any textual items.

Acknowledgement

We thank Malvika Deshmukh for developing an early version of the LCGC algorithm, and Emily Rosengren and Erica Willar for classifying Digg users. Participants at a NIPS workshop and a University of Michigan AI Lab seminar provided useful feedback. This work was supported by the National Science Foundation under awards IIS-0916099 and award IIS-1054199-.

References

Adamic, L., and Glance, N. 2005. The Political Blogo-sphere and the 2004 US Election: Divided They Blog, In *Proc. of the 3rd Intl. workshop on Link discovery*, pp.36-43.

Grinstead, C. M. and Snell, J.L. 1997. *Introduction to Probability (2nd edition).* American Mathematical Society.

Durant, K. T., and Smith, M. D. 2006. Mining sentiment classification from political web logs. In *Proc. of WebKDD'06.*

Efron, M. 2004. The liberal media and right-wing conspiracies: using cocitation information to estimate political orientation in web documents. In *Proceedings of CIKM 2004.*

Gamon, M., Basu, S., Belenko, D., Fisher, D., Hurst, M., & Konig, A. C. 2008. BLEWS: Using Blogs to Provide Context for News Articles. In *Proceedings of ICWSM-08.*

Hirst, G., Riabinin, Y., & Graham, J. 2010. Party status as a confound in the automatic classification of political speech by ideology . In *Proceedings of JADT 2010.*

Jiang, M., and Argamon, S. 2008. Political leaning categorization by exploring subjectivities in political blogs. In *Proceedings of DMIN 2008.*

Klein, D. B., and Stern, C. 2008. Liberal Versus Conservative Stinks. *Society, 45*(6), 488-495.

Landis, J. R., and Koch, G. G. 1977. The measurement of observer agreement for categorical data. *Biometrics,* 33(1).

Laver, M., Benoit, K., & Garry, J. 2003. Extracting policy positions from political texts using words as data. *American Political Science Review,* 97(02), 311-331.

Lin W. H. 2006. Identifying perspectives at the document and sentence levels using statistical models. In *Proc. of NAACL '06.*

Lin, F., and Cohen, W. W. 2008. The multirank bootstrap algorithm: Semi-supervised political blog classification and ranking using semi-supervised link classification. In *Proceedings of ICWSM-08.*

Lin, W. H., Xing, E., & Hauptmann, A. 2008. A joint topic and perspective model for ideological discourse. *Machine Learning and Knowledge Discovery in Databases*, 17-32.

Malouf, R., & Mullen, T. 2007. Graph-based user classification for informal online political discourse. In *Proc. of the 1st Workshop on Info. Credibility on the Web.*

Martin, L. W., and Vanberg, G. 2008. A robust transformation procedure for interpreting political text. *Political Analysis*, 16 (1), 93.

Monroe, B. L., Colaresi, M. P., & Quinn, K. M. 2008. Fightin'Words: Lexical Feature Selection and Evaluation for Identifying the Content of Political Conflict. *Political Analysis*, 16(4), 372.

Mullen, T., & Malouf, R. 2006. A preliminary investigation into sentiment analysis of informal political discourse. In *Proceedings of the AAAI symposium on computational approaches to analyzing weblogs.*

Munson, S., Zhou, D. X., & Resnick, P. 2009. Sidelines: An Algorithm for Increasing Diversity in News and Opinion Aggregators. In *Proceedings of ICWSM'09.*

Munson, S., & Resnick, P. 2010. Presenting Diverse Political Opinions: How and How Much. In *Proc. of CHI'10.*

Oh, A., Lee, H., & Kim, Y. 2009. User Evaluation of a System for Classifying and Displaying Political Viewpoints of Weblogs. In *Proceedings of ICWSM-09.*

Park, S., Ko, M., Kim, J., Liu, Y., & Song, J. 2011. The Politics of Comments: Predicting Political Orientation of News Stories with Commenters' Sentiment Patterns. *In Procs. of CSCW 2011.*

Slapin, J. B., and Proksch, S. O. 2008. A scaling model for estimating time-series party positions from texts. *American Journal of Political Science*, 52 (3), 705-722.

Yu, B., Kaufmann, S., & Diermeier, D. 2008. Classifying party affiliation from political speech. *Journal of Information Technology & Politics*, 5(1), 33-48.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., & Scholkopf, B. 2004. Learning with local and global consistency. In *Proceedings of NIPS 2004*.

Zhu, X., Ghahramani, Z., & Lafferty, J. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of ICML-03*.