Content-Blind Learning on Social Networks

Aron Szanto [1]

Abstract

As the locus of social discourse shifts to the Internet, characterizing the dynamics of social media networks is a key problem in domains from politics to security. While learning on networks traditionally involves graph feature extraction, recent work has considered whether graph kernel methods perform better due to their topologically-rich encodings of a network. In this paper, we build a novel dataset of Twitter networks around a wide variety of news articles, extracting data related to the information diffusion emanating from each story. We find that, blind to the content of an article and using only information from the network surrounding it, we can accurately classify both the topic and the political leaning of the article. Most significantly, we demonstrate that graph kernel techniques far outperform baselines set by feature extraction methods, with superior computational efficiency.

1. Introduction

As social media becomes the world's largest source of real-time news and information, understanding the ways in which different types of information diffuse through these systems becomes vitally important. Many social media platforms like Twitter and Facebook are well-represented by networks whose nodes are users and whose edges are aspects of their interactions. While much work has been done on the separate fields of network classification and social media network mining, there is little research at their intersection. Because of the power that social media holds in arenas from politics to security, deciphering the dynamics of information flow through these networks has critical implications for predicting major societal events like presidential election outcomes and terrorist attacks.

We focus on discerning the latent features of a news article, knowing only information about the topology of the social network that forms around it. We term this *content-blind* prediction. Within the content-blind domain, no linguistic, temporal, or user-identifying information is known to the predictor. For example, a model attempting to determine the political leaning of two articles whose networks are shown in Figure 1 would know nothing but shapes of the networks corresponding to the underlying news articles. We posit that models that use such information to predict the underlying features of a social network have several attractive features, including portability across languages, easy application to a wide variety of problems, and robustness to adversarial agents in the network.

[1]Harvard University, Cambridge, MA, USA.

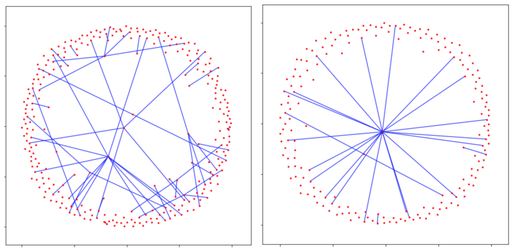
Figure 1. Article Networks: New York Times and Breitbart News

Our contribution is to compare two methodologies for supervised learning on social media networks, bringing both feature-based and graph kernel-based techniques to bear on a variety of topic classification tasks. Building the social networks that form around a variety of news articles, we first use standard network feature predictive models as a baseline performance indicator for the task of gleaning information about the underlying stories. Then we leverage graph kernel learning models to capture information-rich topological encodings of the networks in an attempt to improve on the baseline. In this domain, a model that is better than at-chance guessing for a particular task can be interpreted as successfully using *only features of the information diffusion network* to predict the underlying attributes of the target content.

The remainder of this paper will proceed as follows: Section 2 describes related work in network classification, graph kernels, and learning on social networks. Section 3 contextualizes the content-blind domain and defines our supervised learning task over news articles and Twitter networks. Section 4 describes our data collection and curation methodology. Section 5 details our technical approach in two parts: 5.1 specifies the network feature models we build as a baseline, and 5.2 provides an overview of graph kernel methods before defining the particular algorithms we employ. Section 6 presents our results, comparing the baseline and graph kernel approach along dimensions of computational efficiency and predictive power. Finally, section 7 concludes and discusses extensions of our work to tangential areas of research.

2. Background and Related Work

Much of the research in network classification and graph kernels has been done in the field of computational biology in order to understand protein structures, biological compounds, and gene regulation.

Graph kernel methods involving walks compare two graphs in terms of the paths taken by random walks on those graphs. Borgwardt et al. (2005) make use of random walk graphs kernels to predict protein function. Kashima et al. (2003) use similar methods for two tasks, classifying one set of biological compounds by carcinogenicity and another by mutagenicity.

Another type of graph kernel makes use of subgraph similarity: Kriege & Mutzel (2012) apply this method to a host of chemical compound datasets. A similar graph kernel that counts matching cyclic patterns was used for the classification of molecules by Horváth et al. (2004).

Graph kernels using subtree patterns developed by Ramon & Gärtner (2003) involve counts of matching structures within subtrees for pairs of graphs. These types of graph kernels have also been successfully applied to chemical compound classification tasks (e.g. Shervashidze et al. (2011), who were the first to introduce the Weisfeiler-Lehman graph kernel used in this paper).

The literature on the use of graph kernels in social network classification is sparse. One study in this space develops so-called deep graph kernels, which are traditional graph kernels augmented with neural networks, and applies them to predict Reddit sub-community interactions (Yanardag & Vishwanathan, 2015). Nikolentzos et al. (2017) use a convolutional neural network-based graph kernel that performs well on a synthetic dataset but has more limited success on real-world social network datasets. Though relevant, neither of these studies makes a comparison to a baseline classifier that uses standard features extracted from the networks. Thus, it is unclear if their results demonstrate that graph kernels are effective.

3. Problem

The problem space is defined as follows: each data point x_i is a tuple (a_i, n_i), where a_i is an abstract piece of content with arbitrary underlying features and n_i is a directed graph characterizing its diffusion. In this paper, we refer to a_i as an *article* and define the set of articles as A, while the set of networks is Ω. When an article and a network are linked like this, we say that the network *surrounds* the article.

n_i is defined as (V_i, E_i, L_i), where V_i is the set of entities that interacted with a_i; $(u, v) \in E_i$ iff $u \in V_i, v \in V_i$, and u had an interaction with v that involved a_i; and $L_i : V_i \to \Sigma$ is a labeling function that assigns labels (letters in a finite alphabet Σ) to nodes. Thus, $L_i(v)$ is the label of node v in network n_i. We define the size of n_i as the cardinality of the set of nodes, $|V_i|$. In our paper, a $v \in V_i$ that interacts with a_i is a tweet about a news article a_i, and a directed edge (u, v) denotes that u and v both tweeted about a_i, and v is a retweet of u.

A *tag* is a value associated with an article reflecting some aspect of its content. An article a_i's tag is $y(a_i) \in \lambda$. For example, in one of our models, we define the tag $y(a_i)$ as an indicator for whether article a_i is written about a political topic, and the corresponding $\lambda = \{0, 1\}$.

Thus, the problem of interest is as follows: given a set of N training examples $(y(a_1), n_1), \ldots, (y(a_N), n_N)$, learn a function $F : \Omega \to \lambda$ to predict the tag $y(a_i)$ for article a_i, looking only at the network n_i associated with the article.

In our dataset, one (a, n) pair consists of a news article linked with the Twitter network that surrounds it. Each article is tagged in two ways: whether the article is political or not, and whether (conditional on the article's being political[1]) the article's source is a liberal or conservative media outlet. We discuss the assignment of node labels in Section 5.2.2, as their use is confined to graph kernel methods.

We thus define two concrete tasks related to the formal problem above: given just the social network that surrounds a news article, predict whether the article is political in nature (POL) and whether the article is liberal or conservative (BIAS). We describe the construction of this dataset in Section 4.

4. Data Collection and Curation

We aggregated URLs and other metadata for 10351 news stories in the 7-day period from 11/23/2017 to 11/29/2017[2]. We tagged an article as "political" if and only if the item had metadata that included one of the strings in the set {'clinton', 'elect', 'democrat', 'republican', 'trump', 'senate', 'congress', 'bill', 'politic', 'legis-

[1]This condition is important because we would, for example, expect a sports article from liberal and conservative sources not to differ significantly based on the political leaning of their organizations.

[2]via NewsAPI (https://newsapi.org)

lat', 'gop', 'g.o.p', 'government', 'campaign', 'vot', 'ballot', 'conservative', 'liberal', 'progressive', 'libertarian'}. We gathered all the political articles and for each, tagged them as liberal or conservative by consulting findings from Mitchell et al. (2014), a large-scale Pew Research study that labels a news outlet as liberal if its readers are more liberal than the average internet user and conservative if its readers are more conservative than the average internet user. We omit examples from outlets not expertly categorized in this manner.

Next, for each article, we queried Twitter for tweets that included a link to the article. We built a graph per article by representing tweets about the article as nodes, and drawing edges where one tweet was a retweet of another. We filtered our results to include only those networks with more than 50 nodes, and removed networks whose set of tweets included retweets of a particular tweet but not the tweet itself[3]. Ultimately, we were left with with 2652 tweet networks, each tagged as political or non-political, and 1842 tagged as liberal or conservative. From here, we construct four datasets. For each tag type (POL/BIAS), we create both a class-balanced version and a class-unbalanced version. For the class-balanced version, we use all the examples in the minority class (non-political and conservative, respectively) and randomly sample from the majority class to give a class-balanced dataset. For the class-unbalanced version, we use all examples in both classes. We report the final number of examples in each class for each problem in Tables 1 and 2.

Table 1. Class Sizes for Balanced and Unbalanced POL Datasets

Label	Unbalanced	Balanced
Political	936	936
Not Political	1716	936

Table 2. Class Sizes for Balanced and Unbalanced BIAS Datasets

Label	Unbalanced	Balanced
Liberal	1444	398
Conservative	398	398

5. Model

We detail two methodologies for extracting information from a social network. In the first, we tabulate a feature vector per network corresponding to standard statistics about the nodes and their connections. We tune a variety of models using supervised learning techniques to establish a baseline performance against which to compare more sophisticated algorithms. The second technique uses the Weisfeiler-Lehman (WL) (Shervashidze et al., 2011) graph kernel to compute a high-dimensional comparison between each pair of graphs in the dataset in order to learn a linear classification function in the kernel space.

[3]This is an indication that our results were truncated by the Twitter API for that particular query.

Our analysis considers four datasets, as described above. For each, we randomly assign 90% of the data to a training/validation set and 10% of the data to a test set. We apply the network feature and graph kernel methodologies to each.

5.1. Network Features

For a network n_i, we construct a per-example concatenated vector of graph features that reflect attributes of its shape. These features are of several types and include:

- Basic: Density, Number of Nodes and Edges
- Connectivity: Mean/Max Degree Connectivity
- Degree Centrality: Mean, Max, Std., Skew of In- and Out-Degree Centrality
- Closeness Centrality: Mean/Max
- Cliques: Size and Number of Max Cliques
- Components: Number of Connected Components

For our baseline performance metrics for both the POL and BIAS tasks, we used logistic regression (LR), support vector machines (SVM), random forests (RF), and multilayer perceptrons (MLP) for classification, representing a wide variety of model types and complexities. Hyperparameter tuning for the regularization parameter of SVM and LR, for the SVM kernel choice, and for the several parameters of RF and MLP were done using 10-fold cross validation on the training/validation set. We used the L-BFGS solver with at most 1000 epochs for MLP and the liblinear solver with at most 100 epochs for LR.

5.2. Graph Kernels

5.2.1. Overview

While network feature models attempt to extract relevant statistics based on network attributes, graph kernels are designed to take the full shape of a network into account. Graph kernels are algorithms for computing a topologically-rich similarity metric between networks. Consider the task of determining the similarity between the two small graphs shown in Figure 2. While to the human eye the graphs seem nearly identical, it is a non-trivial problem to compute network similarity at scale. Indeed,

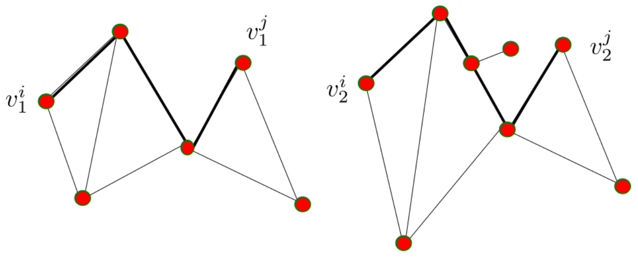

Figure 2. GKs compute a rich comparison between two graphs

it is unclear which standard metrics could be computed over the two graphs in Figure 2 to give insight into their near-isomorphism. Graph kernels move beyond obtaining these traditional features from a graph and instead focus on higher-order similarities like identical subgraphs and similar path patterns. Given a dataset, once all pairs of graphs have been compared using to a graph kernel, the resulting matrix may be fed into a kernelized learning model to compute a linear separation between graphs of different types.

Formally, a graph kernel implicitly represents a network $\mathcal{X} \in \Omega$ as a vector $\phi(\mathcal{X})$ in a Hilbert space $\mathcal{H}$ (Scholkopf & Smola, 2001). A graph kernel $k : \Omega^2 \to \mathbb{R}$ is a positive semidefinite function such that given a mapping $\phi : \Omega \to \mathcal{H}$ and a network pair $(\mathcal{X}, \mathcal{X}')$, the kernel value is equal to the Hilbert-space inner product:

$$k(\mathcal{X}, \mathcal{X}') = \langle \phi(\mathcal{X}), \phi(\mathcal{X}') \rangle_{\mathcal{H}}$$

This inner product represents a similarity metric between two graphs; intuitively, a graph kernel attempts to find shape-preserving representations of two graphs in $\mathcal{H}$ and determine their distance from each other.

To use a graph kernel in a classification model, we arrange our data as a sequence of graphs $G_1, \ldots, G_n$ and construct a matrix K such that $K_{ij} = k(G_i, G_j)$. This matrix is the basis for kernelized learning techniques like SVMs, Gaussian Processes, and Kernel PCA.

Graph kernels generally have one of three flavors, corresponding to their core network algorithm: random walks, shortest paths, and subtrees. We find that average paths on the Twitter networks are very short (the median longest path is 3), so we would expect random walk and shortest path approaches to perform poorly due to their inability to encode information about graphs whose paths are highly invariant. Thus, we choose a state-of-the-art subtree method, the Weisfeiler-Lehman (WL) kernel, which we hypothesize will capture important information about the network structure despite the relatively short average paths thereof.

5.2.2. Weisfeiler-Lehman Kernel

We apply the Weisfeiler-Lehman kernel (Shervashidze et al., 2011) to the domain of social network classification. The WL kernel is a subtree-based approach that measures the similarity of labeled graphs by iteratively comparing common labels, merging labels by edge, then comparing again. It derives its name and underlying technique from the Weisfeiler-Lehman test of isomorphism between graphs, which it applies iteratively to compute a metric for how "close to" (or far from) isomorphism two graphs are.

The computation of the WL kernel begins with a graph $G \in \Omega$ and a choice of a number of iterations p. We proceed by iteration, with each indexed by i. In iteration i, we associate a label $\ell_i(v) \in \Sigma$ and a multiset of strings $M_i(v)$ for each vertex $v \in V$. To begin, we assign $\ell_0(v)$ to $L(v)$ and $M_0(v)$ to the set of the labels of v's neighbors, i.e., $M_0(v) = \{L(v') | v' \in \mathfrak{N}(v)\}$, where $\mathfrak{N}(v) = \{v' | (v, v') \in E\}$.

For iteration i, we set $M_i(v) = \{\ell_{i-1}(v') | v' \in \mathfrak{N}(v)\}$. For each v, we sort and concatenate the strings $M_i(v)$ to obtain $s_i(v)$. Next, we prefix $s_i(v)$ with $\ell_{i-1}(v)$, the labels from the previous iteration. Last, we compress the prefixed $s_i(v)$ by encoding it with a hash $h : \Sigma^* \to \Sigma$, stipulating that $h(s_i(v)) = h(s_i(w)) \iff s_i(v) = s_i(w)$ (i.e., h is a perfect hash function[4]). We set $\ell_i(v) = h(s_i(v))$ for all v.

At each iteration i, the label $\ell_i(v)$ of a node is thus a *distinctive encoding* of a sequence of merges of labels from its neighbors in each iteration. At the end of p iterations, we compute $c_i(G, \sigma_{ij})$, which is a count of the number of times the label σ_{ij} occurs in the labels of G at iteration i. Formally, let the set of labels that occur in the $\ell_i(v)$ associated with G at iteration i be $\Sigma_i = \{\ell_i(v) | v \in V\}$. Assume without loss of generality that $\Sigma_i = \{\sigma_{i0}, \ldots, \sigma_{i|\Sigma_i|}\}$ is sorted. Then we say that the mapping $c_i : \Omega \times \Sigma \to \mathbb{N}$ that represents the number of times that the label σ_{ij} occurs in Σ_i.

Finally, define

$$\phi(G) = (c_0(G, \sigma_{00}), \ldots, c_0(G, \sigma_{0|\Sigma_1|}), \\ \ldots c_p(G, \sigma_{p0}, \ldots c_p(G, \sigma_{p|\Sigma_p|}))$$

That is, the concatenated values of the label counts for each label at each iteration. Then the WL kernel is computed as

$$k(G, G') = \phi(G)^T \phi(G')$$

Figure 3 (Shervashidze et al., 2011) shows the computation of a WL kernel with $p = 1$ for a small graph. Intuitively,

[4]While this is theoretically impossible due to the pigeonhole principle, it is a trivial condition for modern 32- or 64-bit computers to guarantee with near certainty. Indeed, our implementation simply hashes character string labels to integers.

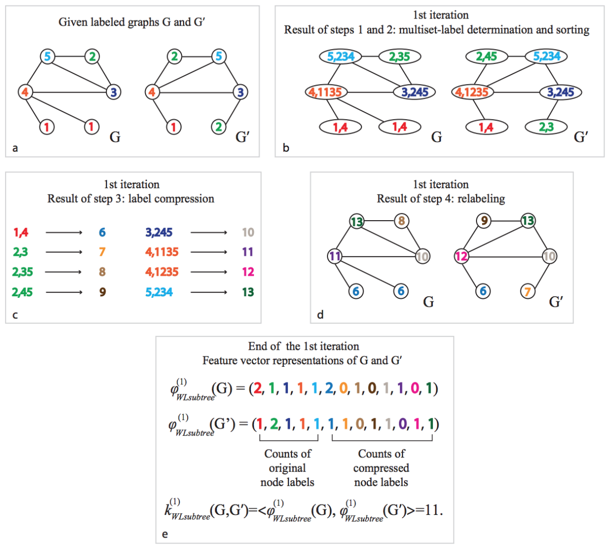
Given labeled graphs G and G′
1st iteration
Result of steps 1 and 2: multiset-label determination and sorting
1st iteration
Result of step 3: label compression
1st iteration
Result of step 4: relabeling
End of the 1st iteration
Feature vector representations of G and G′
Counts of original node labels
Counts of compressed node labels
a
b
c
d
e

Figure 3. WL Kernel Computation for One Iteration

we get higher numbers for this value if the graphs have similar behavior during the merges, i.e. their shapes are similar around a variety of nodes. For p large, a high kernel value means that not only are the graphs similar around pairs of nodes, but that entire regions are similar, since the inherited labels from several hops away are similar. Of course, $k(G, G')$ is maximized when G is isomorphic to G', and thus their labels are identical for arbitrary iterations.

The time complexity to compute $\phi(G)$ as described above is linear in the (maximal) number of edges $m = |E|$. Thus calculating k for all pairs of N graphs for p iterations is an $O(N^2mp)$ algorithm. In our implementation, however, we cache values of $\phi(G)$ for each graph in the dataset before computing the kernel matrix, reducing our run time to $O(Nmp)$, linear in the size of our dataset.

Our implementation is in Python, using the networkx (Hagberg et al., 2008) package, based on (Sugiyama et al., 2017). Once a kernel matrix was computed for a problem, we used an SVM model to learn a linear separation between classes for the task at hand. We use 10-fold cross validation on our training/validation set to determine both the optimal number of WL iterations p and the SVM regularization parameter. We use a node's degree as its initial label (i.e., $\forall_v L(v) = |\mathfrak{N}(v)|$).

Finally, note that the set of nodes are disjoint across all networks, since the tweets in one network explicitly mention the article associated with that network[5]. Thus this labeling does not leak any relevant information about a node's identity, since even if the same user's tweets were represented in two graphs, they would only theoretically share any information if they had exactly the same number of retweets.

[5]It's theoretically possible that one tweet mentions more than one article, but we did not find any such examples in our dataset.

Table 3. Classification Performance. Best performers for each problem and for each metric are **bolded**. Metrics for the WL kernel are underlined.

BIAS-B	acc	pre	rec	F1
LR	0.65	0.67	0.63	0.65
SVM	0.70	0.75	0.64	0.69
RF	0.69	0.72	0.63	0.68
MLP	0.71	0.95	0.46	0.62
WL	**0.93**	**0.91**	**0.98**	**0.94**
BIAS-U	acc	pre	rec	F1
LR	0.80	0.80	0.99	0.88
SVM	0.77	0.77	1.00	0.87
RF	0.82	**0.94**	0.95	0.91
MLP	0.77	0.77	1.00	0.87
WL	**0.90**	0.88	**0.99**	**0.93**
POL-B	acc	pre	rec	F1
LR	0.67	0.71	**0.61**	0.66
SVM	0.68	0.68	0.64	0.66
RF	0.67	0.71	0.60	0.65
MLP	0.67	0.73	0.58	0.64
WL	**0.76**	**0.89**	**0.61**	**0.72**
POL-U	acc	pre	rec	F1
LR	0.74	0.63	0.44	0.52
SVM	0.71	0.65	0.39	0.49
RF	0.74	0.61	0.53	0.57
MLP	0.74	0.59	0.58	0.58
WL	**0.82**	**0.85**	**0.65**	**0.74**

6. Results

We refer to our political and bias tasks with class balancing and no balancing as POL-B, POL-U, BIAS-B, and BIAS-U, respectively. We report accuracy, precision, recall, and the F1 score for the network feature baseline and the WL kernel together in Table 3.

We find that the WL kernel significantly outperforms the baseline accuracy on all tasks and nearly all metrics as well. On the balanced BIAS task in particular, the graph kernel approach demonstrates its effectiveness, predicting the political leaning of an article at 93% accuracy simply by looking at the topology of the network surrounding it. In all, the kernel approach represents a 10-31% improvement on the network feature extraction methods. Notably, our graph kernel model maintains a high F1 score independent of the balance of classes in the dataset. This suggests that the model holds significant predictive power unrelated to idiosyncracies in the class distribution.

The features that were the most predictive for our standard

Table 4. Optimal Hyperparameters: WL-p and SVM-C

Parameter	BIAS-B	BIAS-U	POL-B	POL-U
WL-p	1	1	1	1
SVM-C	0.04	0.04	0.09	0.09

models in the BIAS task were related to connectivity, while those that were the most predictive for the POL task were related to centrality. This indicates that in traditional terms, the political leaning of an article affects the behavior of average users more, and the topic of an article affects the behavior of high-influence users more.

The results of the cross validation for optimizing the number of WL iterations p and SVM regularization parameter C are given in Table 4. The finding that one iteration of WL was sufficient for high predictive power is significant and belies a profound result about classifying Twitter networks. Recall that at iteration $i = 1$, the label for a node v is

$$\ell_1(v) = h(L(v)||s_1(v))$$

where $s_1(v)$ is the sorted and concatenated $M_1(v) = \{L(v')|v' \in \mathfrak{N}(v)\}$. Thus, each label at iteration 1 was passed information from the nodes at most one hop away. If one such label merging operation from neighbors to nodes is sufficient to detect similarity (as our results demonstrate), then the shapes of the local neighborhood structures around single nodes in a social network are highly predictive of the article that the network surrounds. Moreover, the result lends weight to the hypothesis that subtree methods will both perform and scale well when applied to Twitter networks, since they do not require long paths to differentiate between graphs. We anticipate that because average paths are short in these networks, graph kernels based on random walks or shortest paths will tend to underperform subtree kernels. However, future work will have to test this intuition by comparing WL against a variety of other kernel methods.

One of the major criticisms of graph kernels is that they do not scale; the graph isomorphism problem at the heart of many kernel algorithms has no known polynomial time algorithm. Thus, many graph kernels have runtimes that are cubic or worse (Vishwanathan et al., 2010). However, the domain of social networks is a good fit for graph kernels, since the graphs are significantly smaller than those in traditional graph kernel applications like biochemistry. In Table 5 we present runtime results for our algorithms, finding that our linear implementation of the WL kernel is more than twice as fast as the network feature extraction routine.

Our last result regards the convergence of our models. We demonstrate that because of the small amount of data available, the WL kernel model did not reached peak performance on either the POL or the BIAS task. By varying the size of our training sets, we show that performance increases as the model consumes more training data, but that it has not yet converged. However, a similar experiment demonstrates that it is less obvious that the models used to compute the baseline have not converged for both tasks. This finding further confirms the superiority of the WL graph kernel in the Twitter network domain and bodes well for future work involving larger datasets. Figures 4 and 5 demonstrate the disparity between WL kernel and network feature model convergence for the BIAS and POL tasks, respectively.

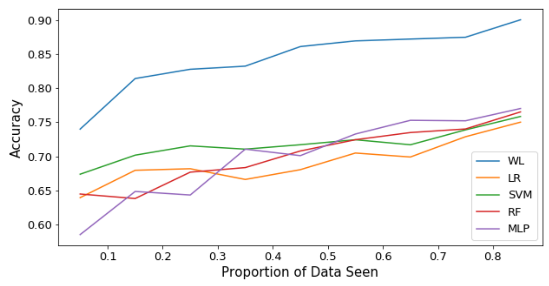

Figure 4. Model Convergence: BIAS

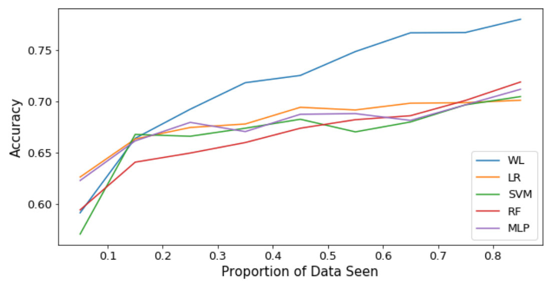

Figure 5. Model Convergence: POL

7. Conclusion

In this paper, we defined the problem of identifying attributes of online articles using content-blind techniques. We constructed a novel real-world dataset composed of news articles and their associated Twitter networks, build-

Table 5. Runtime: Time to Compute Network Features and Graph Kernels (seconds)

	POL	BIAS
Network Features	60.4	50.4
WL Kernel	**29.1**	**22.7**

ing both network feature models as baseline performance indicators and a fast implementation of the Weisfeiler-Lehman graph kernel. Testing on both class-balanced and class-unbalanced datasets, we demonstrated that the WL kernel outperforms the baseline significantly in accuracy by up to 31%, and in runtime efficiency by up to 108%. Finally, we interpreted our results topologically, arguing that analyzing Twitter networks at the node-neighborhood granularity is important for content-blind classification, and demonstrating mathematically that the WL kernel does exactly this.

Our findings demonstrate that there is information about the content of an article encoded in the shape of the network surrounding it that is rich enough to inform prediction in at least two arenas as disparate as topic and political leaning. The implications are numerous: can topology be used to classify networks by the sentiment of their interior article? By truthfulness? While the content-blind domain complicates the article classification task significantly, it has an important advantage in that a successful model is highly robust to adversarial actors. It is relatively easy to reverse-engineer a linguistic model for topic or veracity prediction in order to inject content intended to fool the classifier. However, doing the same in the face of a graph model would require orchestrating a coordinated attack in which a critical mass of network participants must change their behavior to pass off an article as a different type from its true one. In this way, our findings may be used to augment non-content-blind models as a security measure against malicious agents in the network.

Other future work might include testing other types of graph kernels against the baseline set by WL; while the standard kernels based on random walks and shortest paths should be a starting point, more exotic kernels that involve deep neural networks (Nikolentzos et al., 2017; Yanardag & Vishwanathan, 2015) might be explored. Because of our findings regarding model convergence, we believe that the WL kernel may perform even better if given more training data; as such, further experimentation is necessary with larger datasets. Development of further prediction tasks is another important avenue of research– multi-class topic classification problem that extends our POL task is one possibility; another is an exploration into whether networks surrounding articles that were written to deceive are topologically different than those surrounding articles written to inform. Yet another area of future research is to introduce a notion of time; truncating edges based on when they formed can simulate performing the classification tasks in real time, allowing insight into how prediction accuracy improves with the evolution of the network.

References

Borgwardt, Karsten M, Ong, Cheng Soon, Schönauer, Stefan, Vishwanathan, SVN, Smola, Alex J, and Kriegel, Hans-Peter. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

Hagberg, Aric A., Schult, Daniel A., and Swart, Pieter J. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pp. 11–15, Pasadena, CA USA, August 2008.

Horváth, Tamás, Gärtner, Thomas, and Wrobel, Stefan. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pp. 158–167, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: 10.1145/1014052.1014072. URL http://doi.acm.org/10.1145/1014052.1014072.

Kashima, Hisashi, Tsuda, Koji, and Inokuchi, Akihiro. Marginalized kernels between labeled graphs. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 321–328, 2003.

Kriege, Nils and Mutzel, Petra. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483*, 2012.

Mitchell, Amy, Gottfried, Jeffrey, Kiley, Jocelyn, and Matsa, Katerina Eva. Political polarization and media habits, Oct 2014. URL http://www.journalism.org/2014/10/21/political-polarization-media-habits/.

Nikolentzos, Giannis, Meladianos, Polykarpos, Tixier, Antoine Jean-Pierre, Skianis, Konstantinos, and Vazirgiannis, Michalis. Kernel graph convolutional neural networks. *arXiv preprint arXiv:1710.10689*, 2017.

Ramon, Jan and Gärtner, Thomas. Expressivity versus efficiency of graph kernels. In *Proceedings of the first international workshop on mining graphs, trees and sequences*, pp. 65–74, 2003.

Scholkopf, Bernhard and Smola, Alexander J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001. ISBN 0262194759.

Shervashidze, Nino, Schweitzer, Pascal, Leeuwen, Erik Jan van, Mehlhorn, Kurt, and Borgwardt, Karsten M. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

Sugiyama, Mahito, Ghisu, M. Elisabetta, Llinares-Lpez, Felipe, and Borgwardt, Karsten. Graphkernels: R and Python packages for graph comparison. *Bioinformatics*, pp. btx602, 2017. doi: 10.1093/bioinformatics/btx602. URL +http://dx.doi.org/10.1093/bioinformatics/btx602.

Vishwanathan, S. V. N., Schraudolph, Nicol N., Kondor, Risi, and Borgwardt, Karsten M. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, August 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1859891.

Yanardag, Pinar and Vishwanathan, SVN. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374. ACM, 2015.