Defuse the News: Predicting Misinformation and Bias in News on Social Networks via Content-Blind Learning

Contents

Acknowledgments

I am indebted to many individuals for their help in this project. Thanks to David Parkes, my mentor, advisor, and friend, for his steady guidance, boundless expertise, and remarkable ability to conjure up extra time despite an airtight schedule. Thanks also to Nir Rosenfeld, maestro of network science, for his advice and keen eye for detail.

I am appreciative of Deb Roy, Soroush Vosoughi, and the Lab for Social Machines at MIT for opening their dataset to me and thus allowing this work to take on some semblance of social relevance. Thanks as well to Christopher Hase for his help with preliminary experimentation regarding bias and topic models and to Sasha Rush for his advice and provision of computational resources. Thank you to Hal Roberts and Rob Faris of the Berkman Klein Center, Duncan Watts and Markus Mobius of Microsoft Research, and Michael Bronstein of the Radcliffe Institute for stimulating conversations along the way.

I am grateful to Margo Levine for her advice and friendship in matters both academic and decidedly not. I also thank Sarah Iams for helping me think about transforming this work from a highly technical and esoteric machine learning paper into a thesis accessible to a wider audience.

I thank my friends, especially Camille N'Diaye and Joe Kahn, for their companionship during late nights, for providing me with sustenance in the form of coffee and encouraging thoughts, and for making each day at Harvard fun, interesting, and bright. Thanks as well to the Harvard Ultimate community for keeping me sane through this process and for entertaining lots of fake news jokes that weren't very funny.

Last, thanks to my family – David Szanto, Lesley Rosenthal, Ted Rosenthal, Istvan Szanto, Vali Sajber, Nancy Fadem, and all the others – for more love, support, and wisdom than I could have ever asked for. None of this would be possible without you all.

All control, in essence, is about who controls the truth.

Joseph Rain

1 Introduction

As social media becomes the world's largest source of real-time news, understanding how false information spreads is vitally important. The stakes are high: over three billion people worldwide use social media platforms like Twitter and Facebook, and more than two thirds of Americans consume their news primarily through social media (Kemp, 2017; Shearer & Gottfried, 2017). While the social benefits of these technologies are undeniable, the risks to truth are striking. 2016 saw fake news become not just a cultural phenomenon but a genuine threat to democracy and the civic fabric. Researchers estimate that Americans clicked on false stories 760 million times before that year's presidential election, an average of nearly four such stories per adult (Allcott & Gentzkow, 2017). The ubiquity of fake news, combined with the critical mass of social media users, has led some analysts (Parkinson, 2016; Read, 2016) to conclude that these stories contributed directly, even decisively, to the outcome of the election. No less alarming is the recent and sharp uptick in online racial extremism, terrorism, and harassment closely linked with the proliferation of fake news (Rainie et al., 2017).

Having already become a key vector for the spread of hate and civic unrest, fake news now has the potential to cause irreparable damage to the social order. Worryingly, there is little hope to slow or prevent its spread through conventional means, as it operates at a scale where human fact checkers fall somewhere between stretched thin and entirely useless. As such, false stories will continue to propagate ever faster and more pervasively through social networks if left unchecked. The pivotal question then becomes how quickly and accurately we can identify fake news using methods that do not require human oversight, and how resistant these methods are to malicious interference by the originators of false stories. Technological problems call for technological solutions.

Social media platforms like Twitter and Facebook are well-represented by networks (sometimes known as graphs) whose nodes are users and whose connecting edges are aspects of their interactions. Just as experiments on physical social networks helped early public health officials implement vaccination schemes to prevent epidemics, research on social media networks could provide insight into the ways that information is broadcast and spread online (Bailey et al., 1975). It is thus fitting that rather than viruses it is viral articles that are used to uncover patterns in the diffusion of misinformation.

While much work has been done on the separate fields of network classification (making predictions about a network never seen before) and social network modeling (applying graph theory to the field of interpersonal interaction), there is little research at their intersection. Because of the power that social media commands in arenas from politics to security, deciphering the dynamics of information flow through these networks via methods at this nexus has critical implications for analyzing and predicting matters of societal importance. Indeed, presidential election outcomes, terrorist attack planning, and the viral spread of misinformation each rely heavily on large-scale social networks.

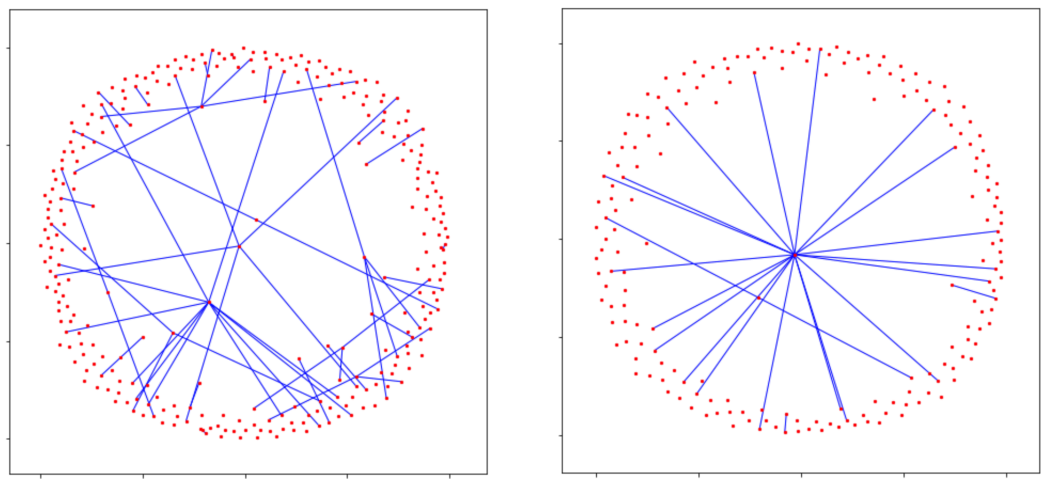
This work studies Twitter as a substrate for information and rumor diffusion. Founded in 2006, the microblogging platform has amassed over 300 million users, becoming one of the ten most popular sites on the internet (Molina, 2017). Twitter has also cemented itself as one of the foremost online hubs for current events: on the day of the 2016 election, Twitter saw over 40 million election-related posts, making it the world's largest source of breaking news (Isaac & Ember, 2016). Because of the way that users interact by replying to and sharing each other's content, techniques from network science are advantageous for constructing predictive models for how information spreads through the Twittersphere. This research examines whether the characteristics of the network of tweets about a particular news article contain information about the textual content of the article.

In particular, we focus on discerning three latent features of a news article – veracity, political bias, and topic – knowing only information about the shape 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 – not even the title of the article or the names of the users sharing it – is known to the predictor. Instead, content-blind models only have access to the patterns of user interaction with respect to the article, attempting to make a determination about the characteristics of the news based solely upon the structural diffusion pattern of the article through social media. We posit that models that use these fundamental attributes of a social network to predict information about news articles have several attractive features not found in content-aware systems, including portability across worldwide languages, easy application to a wide variety of prediction problems, and, most importantly, resistance to adversarial agents trying to pass off fake news as real.

1.1 Overview

Network modeling is fundamental to a wide variety of areas whose core abstraction is a structure typified by entities with distinctive relationships between them. Systems as essential as those that underlie power grids, air traffic control, and GPS all rely on such models. One important domain in which network models are vital is social media, characterized by users connected by their interactions on the platform. A network is defined in a formal sense as a collection of entities called *nodes* (also known as vertices) and the connective relationships between them called *edges*. Edges may be directed – drawn from one node towards another, for example denoting the direction of electricity flow – or undirected, for example in a social network in which there are no unrequited friendships. In this work, we model discussions of current events on Twitter via directed networks.

On Twitter, each user shares *tweets* with their followers. A tweet may include text, multimedia content, or a link to an external website, such as a news article. A user


(a) New York Times

(b) Breitbart News

Figure 1.1: Article networks for one New York Times story and one Breitbart News story. Red dots denote tweets sharing the story in question, and blue edges denote retweet relationships between tweets. The content-blind domain requires that a model take in information only about the shape of the networks and make a determination as to, for example, the political leaning of each story. A successful model will discriminate between stories based only on the topology of the networks, in this case labeling these two networks correctly as politically liberal and conservative, respectively.

viewing a tweet authored by one of their followers may elect to *reply* to it directly or *retweet* it to their own follower circle, thus extending the reach of that tweet. The pattern of retweeting for one or more related tweets is known as a *cascade*. The essence of this work is to determine whether the underlying characteristics, or topology, of a particular cascade is informative of the veracity, political bias, or topic of the news article encapsulated by the network.

We formalize our analysis of discussions on Twitter by rendering each tweet as a node and treating retweet connections as directed edges. For example, if tweets A and B are each nodes, then there is a directed edge between A and B if B is a retweet of A.

Within this paradigm, we can begin analyzing the patterns in the networks associated with various types of news. To do this, we first assemble all the tweets that reference a particular news article. Then, we build the network induced by these tweets; we term this the network that *surrounds* the article. For example, Figures 1.1a and 1.1b illustrate the networks surrounding stories from The New York Times and Breitbart News. The structure of one such network, defined by the tweet nodes and retweet edges for a given article, is the input to a content-blind model. We collect Twitter data for many thousands of news articles, building a corpus of cascades as a foundation for analysis. For each cascade, we *tag* the news article it surrounds as real or fake, liberally or conservatively biased, and political or apolitical[1]. Once we have amassed this collection of articles and tags, however, the question remains as to how to analyze them to uncover the latent patterns in their structure. For this, we use an ensemble of methods from the field of machine learning.

The approaches taken in this work are members of the class of techniques known as *supervised learning* algorithms. This methodology relies on a two-part dataset, of which one segment is used to learn patterns (the *training* phase) and the other is used to validate the performance of the model (the *testing* phase). With a model trained on one subset of data, the task at hand is to make a prediction about an example found in the other subset, namely to determine a tag (i.e., veracity, bias, or topic) associated with

[1]For a discussion as to how we determine these tags, see Chapter 2.2.

it. A veracity model, for example, after having learned topological patterns associated with the truthfulness of news on Twitter, makes an informed decision as to whether a network it has not encountered before corresponds to an article that is real or fake. Because we have the "correct" tags for each example, repeating this experiment for each network in the testing set and comparing the model's output with the ground truth will allow us to deduce roughly how accurate the model is. A fully trained and tested model, then, could be released to the Internet and make accurate predictions about networks forming in real time.

If we had a simple metric that could determine the similarity between two networks, this problem would be easily solved: to decide if a network corresponds to a real or fake article, we would just ask whether it is more similar to those that surround real articles or fake ones. However, networks are complex structures with exponentially many possible configurations for any given number of nodes. As such, the task of determining the similarity between two networks is extremely difficult (Garey & Johnson, 1990). This work leverages a family of algorithms known as graph kernels (Vishwanathan et al., 2010) to approximate this similarity by measuring the divergence in shape between two networks at the level of small neighborhoods of nodes. Once a kernel has calculated the similarity between each pair of graphs in the training data, we can use the kernel to make a prediction for a novel example, tagging it based on the tags of the networks in the training group to which it bears the greatest similarity. To evaluate the success of a kernel, we apply simpler models that depend on basic statistics (such as the number of nodes and edges) called *network features* to the same set of tasks. If the kernel outperforms these baseline models, then we can conclude that the shape of a network encodes information interpretable by the kernel about the characteristics of the content that the network surrounds.

1.2 Contributions

This thesis makes three main contributions. First, it introduces the content-blind domain and several archetypal problems therein, including the determination of veracity, bias, and topic for a news article based solely on the topology of the social network that surrounds it. Second, it employs two methodologies for supervised machine learning on social media networks within this domain, applying both network feature and graph kernel techniques to the tasks defined. We start by gathering the Twitter retweet networks that form around a large set of news articles and rumors from two datasets. The first is built from scratch, drawing on news articles from a wide variety of journalistic outlets spanning the political spectrum. The other is based on work by Vosoughi et al. (2018), who collated 126,000 rumor cascades on Twitter, labeling them as true or false by consulting the determinations of independent fact-checking organizations such as Snopes. Comprising over 4.5 million tweets, it is the largest dataset of fake news stories and social networks ever assembled. The analysis in this thesis begins by using standard network feature predictive models as baseline performance indicators for the task of gleaning information about the underlying stories. Then we present a novel use of graph kernel learning models to capture information-rich topological encodings of the networks in an attempt to improve on the baselines. This work is the first application of machine learning models to these data, representing the first set of results for fake news detection at this scale. Strikingly, topological graph kernels significantly outperform the baseline models, predicting the veracity of news articles at 84% accuracy, the political bias at 93%, and the topic at 76% while adhering to the content-blind guidelines. This performance is competitive with or superior to state-of-the-art systems for fake news detection, whose accuracies fall between 78% and 89% (Yu et al., 2017; Ruchansky et al., 2017). Remarkably, while all current models for fake news detection are content-aware and depend on large corpora of article texts, user behavior histories, and website data (Ma et al., 2016; Liu et al., 2015), the methods presented here require nothing but the basic propagation contours of information through Twitter to classify its veracity accurately.

The third contribution of this work is to demonstrate that for the problem of misinformation detection, graph kernel methods are much faster than standard models and are robust to adversarial interference from the producers of fake news. The compliance of graph kernels with content-blindness means that even the cleverest of fake news writers would have to go beyond writing a story that could fool a content-aware veracity model, such as one that analyzes the text of an article. Instead, producers of false content could only dupe a content-blind model by changing the network topology writ large, which would require much broader influence over the behavior of the crowd interacting with the false content than is currently viable.

1.3 Related Work

Despite the relatively recent rise of fake news as a cultural fixture, there has long been academic interest in the propagation of true and false information through social media networks. Friggeri et al. (2014) describe the ways in which rumors spread on Facebook, finding that trust relationships play an important role. Bakshy et al. (2012) model information propagation and virality using field experiments to determine whether strong or weak ties between users are more responsible for the diffusion of novel information. And Gabielkov et al. (2016) report results on Twitter in particular, studying different types of retweet cascade patterns. Most recently, Allcott & Gentzkow (2017) explore the role of fake news in the 2016 presidential election, including a statistical analysis of the rate at which Americans consumed false stories on social media prior to Election Day.

There have also been several systems proposed to predict information veracity on social media platforms. Castillo et al. (2011), Liu et al. (2015), Ma et al. (2016), and Tacchini et al. (2017) each use statistical models of article and tweet text combined with detailed user histories to classify rumors. More recently, deep learning approaches as in Yu et al. (2017) and Ruchansky et al. (2017) combine these features with news website data, using convolutional and recurrent neural networks to achieve good predictive performance on datasets from Twitter and its Chinese analog Weibo. However, this research is only tangential to our work, as we deny our models access to such rich linguistic and user-based data, instead focusing strictly on topological information as predictive of veracity. It is this insight that makes our results robust to adversarial writers of fake news or clever botnet administrators.

At the intersection of network theory and fake news, the study that is most similar to our research is Vosoughi et al. (2018), which characterizes the differential diffusion of true and false rumors through Twitter. Indeed, in this work we perform analysis on the dataset that they constructed. However, while their conclusions are retrospective, ours go one step further, to the predictive: whereas they report how real and fake news spread through social networks differently without attempting to identify misinformation, we determine that real and fake news can indeed be differentiated by the shapes of their cascades. Thus, this is the first work to perform predictive analytics on this important dataset.

Much of the existing 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 for the purpose of drug discovery.

Walk-based kernel methods compare two graphs in terms of the paths taken by random walks on those graphs. Borgwardt et al. (2005) make use of random walk graph kernels to predict protein function based on their molecular shape. 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 comparable 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) count matching structures within subtrees over pairs of graphs. These types of graph kernels were also successfully applied to chemical compound classification tasks by Shervashidze et al. (2011), who were the first to introduce the Weisfeiler-Lehman graph kernel used in this work.

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 graph kernel based on a convolutional neural network that performs well on a synthetic dataset but has 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 particularly effective.

1.4 Roadmap

The remainder of this work will proceed as follows: Chapter 2 contextualizes the content-blind domain, provides a formal definition of the supervised learning task over news articles and Twitter networks, and describes the data collection and curation methodology. Chapter 3 details the technical approach in two parts: Section 3.1 specifies the network feature models built as baseline performance indicators and Section 3.2 provides an overview of graph kernel methods before introducing the particular algorithms employed. Chapter 4 presents experimental results, comparing the baseline and graph kernel methods along dimensions of predictive power and computational efficiency while providing topological interpretations of the findings. The end of this chapter highlights the advantages of content-blind methods over alternative approaches, such as those that rely on natural language models. Finally, Chapter 5 concludes and discusses extensions of this work to allied areas of research.

The amount of energy needed to refute bullshit is an order of magnitude greater than that needed to produce it.

Linda Gordon

2 The News Classification Problem

The task of interest in this work is to deduce a tag associated with a news article, having access only to the social network that surrounds the article. The exposition of this problem has two parts: a mathematical formulation and the curation of corresponding real-world data.

2.1 Formal Definition

We define the problem space as follows: each data point is a tuple (a_i, G_i), where a_i is an abstract piece of online content with arbitrary underlying features and G_i is a directed graph (network[1]) 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.

G_i, a network in Ω, is defined as a 3-tuple (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 $L_i : V_i \rightarrow \Sigma$ is a many-to-one labeling function that assigns labels (letters in a finite alphabet Σ) to nodes. Thus, $L_i(v)$ is the label of node $v \in V_i$ in network G_i. We define the size of G_i as the cardinality of the set of nodes, $|V_i|$, referred to later as n. In this work, a $v \in V_i$ that interacts with a_i is a tweet about an 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 discrete value associated with an article reflecting some aspect of its content. Given a tagging function $y : A \rightarrow \Lambda$, an article a_i's tag is $y(a_i) \in \Lambda$. For

[1]Because of the interdisciplinary nature of this work, consistency in nomenclature is difficult. Graph theorists refer to graphs, while those who study information propagation talk about networks. We use the terms interchangeably.

example, in a veracity model the tag $y(a_i)$ is an indicator for whether the content in article a_i is fake, and the corresponding $\Lambda = \{0, 1\}$.

Thus, the problem of interest is as follows: given a set of N training examples $(y(a_1), G_1), \dots, (y(a_N), G_N)$, learn a function $F : \Omega \to \Lambda$ to predict the tag $y(a_i)$ for article a_i, looking only at the network G_i associated with the article.

In our dataset, one (a, G) pair consists of a news article and the Twitter network that surrounds it. Each article is given a set of tags: whether the article is political in nature, whether (conditional on the article's being political[2]) the article's source is a liberal or conservative media outlet, and whether the article's content is false. We discuss the assignment of node labels in Section 3.2.3, as their use is confined to graph kernel methods.

There are thus three concrete tasks related to the formal problem above: given just the social network that surrounds a news article, predict whether the article is political (POL), whether the article is liberal or conservative (BIAS), and whether the article is fake news (VER). Section 2.2 describes the construction of datasets for these problems.

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

2.2 Data Collection and Curation

The process of data collection for this work was done in two parts. The first involved the creation of a novel dataset from scratch for use in the POL and BIAS tasks, while the second drew upon work done by Vosoughi et al. (2018), benefiting from the authors' special agreement with Twitter to amass large-scale cascade data for true and false rumors.

For the first part, we aggregated URLs and other metadata for 10,351 news stories over the course of 14 nonconsecutive days in November and December 2017[3]. These articles were drawn from several dozen journalistic outlets with a variety of readerships and political leanings, including The New York Times, Breitbart News, The Wall Street Journal, Fox News, The Huffington Post, and others. We tagged an article as political if and only if the item had a title or URL that included one of the strings in the set {"clinton", "elect", "democrat", "republican", "trump", "senate", "congress", "bill", "politic", "legislat", "gop", "government", "campaign", "vot", "ballot", "conservative", "liberal", "progressive", "libertarian"}. Each political article received an additional tag corresponding to its political leaning, assigned by consulting findings from Mitchell et al. (2014). This work is 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

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

its readers are more conservative than the average internet user. Articles from outlets not expertly categorized in this manner are omitted from the dataset.

Next, for each article, we query the Twitter Standard Application Programming Interface (API)[4] for tweets that included a link to the article. This API does not grant unfettered historical access, however. Instead, it imposes strict limits on the content that can be retrieved. In particular, only tweets published in the last seven days are searchable, and searches are run against a 1% sample of these tweets. As such, there was some difficulty in constructing a complete and accurate dataset under these constraints. Moreover, extreme care had to be taken to ensure that despite the 1% sampling, the cascades constructed from the API results resembled real networks, i.e., edges connected exactly two nodes and there were no orphan tweets with an in-edge but no corresponding parent tweet.

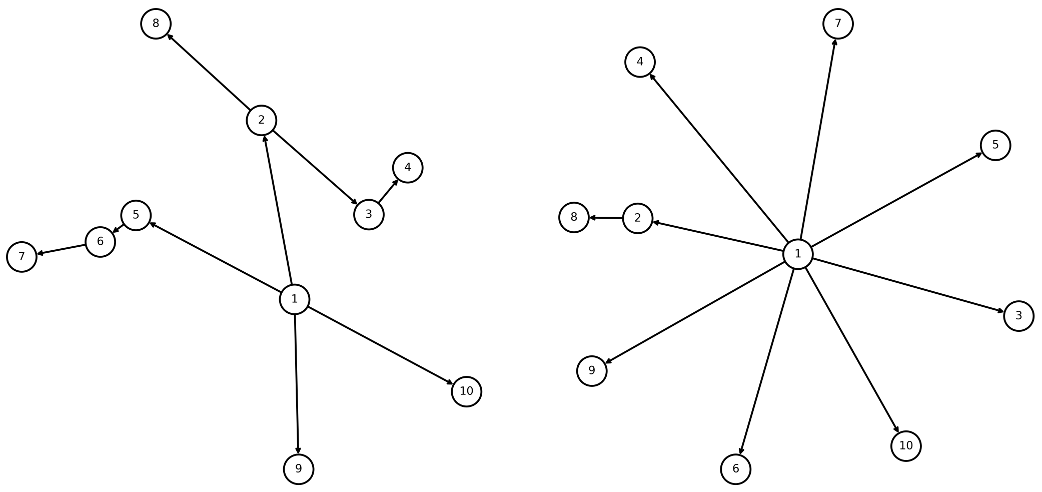
We built a network per article by representing tweets about the article as nodes and drawing edges where one tweet was a retweet of another. We filtered our results to include only those networks with at least 50 nodes and discarded networks whose set of tweets included retweets of a particular tweet but not the tweet itself[5]. Complicating this task significantly was the fact that Twitter's Standard API does not always return full retweet chains. Rather, if tweet B retweets tweet A, and tweet C retweets B,

[4]https://developer.twitter.com/en/docs/tweets/search/overview/standard

[5]This was an indication that our search results were incomplete for that particular query, since by definition a retweet must have a corresponding tweet.

then the API may compress the path from A to C through B by simply returning two retweet edges: one between A and C and one between A and B, making no mention of the (B,C) edge. It is unclear under what conditions this compression occurs, but the median longest path over all networks in this dataset was 2, and the overall longest path was 3. Clearly, this suggests a fairly high rate of path compression. By way of example, Figure 2.1a depicts a true Twitter cascade, while Figure 2.1b illustrates its compressed counterpart that might have been returned in response to a Standard API query. As shown, the chains between Node 1 and Node 7 and between Node 1 and Node 4 have been collapsed, but the chain between Node 1 and Node 8 remains intact. This compression proved important for tuning the graph kernel because it meant that in almost all cases the model would not learn anything new by looking for topological patterns in locations more than one edge away from any particular node.

Ultimately, this dataset was composed of 2,652 tweet networks, each tagged as political or non-political, and 1,842 tagged as liberal or conservative. From here, four datasets were constructed. For each tag (POL/BIAS), we created both a class-balanced version and a class-unbalanced version. For the class-balanced version, we used all the examples in the minority class (political and conservative, respectively) and randomly sampled from the majority class to give a class-balanced dataset. For the class-unbalanced version, we used all examples in both classes. We report the final number



(a) Original network **(b)** Network with some edges compressed

Figure 2.1: The Twitter Standard API collapses some retweet edges, compressing the cascade topology in certain network localities.

of examples in each class for each problem in Tables 2.1 and 2.2.

Table 2.1: Class Sizes for Balanced and Unbalanced POL Datasets

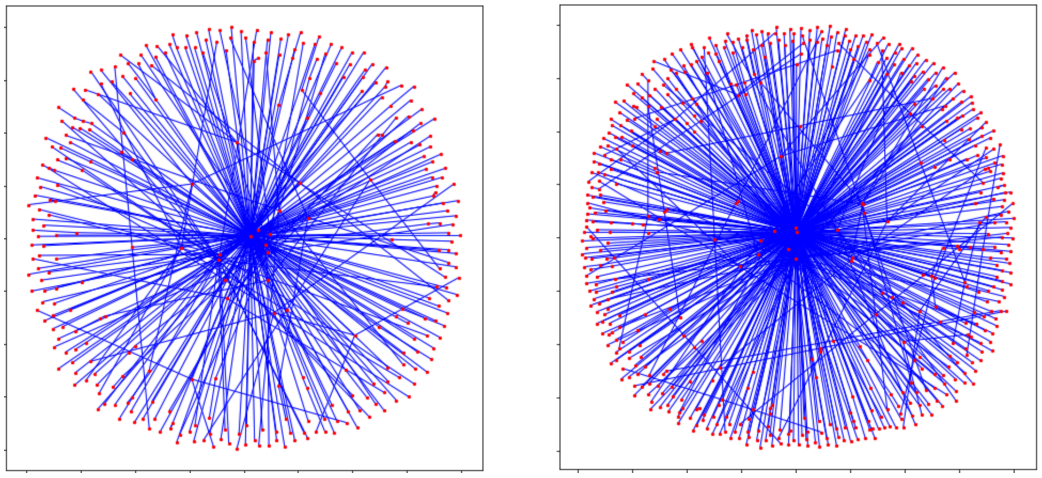
Label	Unbalanced	Balanced
Political	936	936
Not Political	1,716	936

From there, we calculated the requisite network feature statistics for each example and split the datasets into training and testing portions in accordance with Sections 3.1 and 3.3.

For the VER task, we operate on the same dataset as in Vosoughi et al. (2018), which comprises a set of cascades, each corresponding to a rumor that has been fact-checked and labeled as true or false. Benefiting from a direct relationship with Twitter, the

Table 2.2: Class Sizes for Balanced and Unbalanced BIAS Datasets

Label	Unbalanced	Balanced
Liberal	1,444	398
Conservative	398	398



(a) Network surrounding real news article **(b)** Network surrounding fake news article

Figure 2.2: Article networks for one real and one fake story. Significant differences in topology cannot be easily discerned by the naked eye. However, techniques such as graph kernels may be able to tag these two networks correctly.

authors were able to construct a larger and more complete set of rumor cascades than any prior work has. They begin building their collection of networks by searching the full Twitter historical archive for any tweets whose replies include a link to a fact-checking article from one of six websites[6]. These tweets were termed *rumors*. They confirmed that the fact-checking article was related to the content of the rumor by computing a word embedding for the tweet's text (Vosoughi et al., 2016) and for the

[6]snopes.com, politifact.com, factcheck.org, truthorfiction.com, hoax-slayer.com, and urbanlegends.about.com

article's content (Le & Mikolov, 2014), then measuring the cosine similarity between the two. If this metric, a measure of semantic correspondence between two texts, was high enough, they deemed the fact-checking article to have been written about the rumor. Then they collected all the retweets of the rumor, building the full cascade while discarding tweets that were determined by a bot-detection algorithm (Varol et al., 2017) to have been written automatically. Last, they applied a method known as time-inferred diffusion (Goel et al., 2012) to uncompress implied retweet edges by consulting the full Twitter follower graph, an index of the relationships among all Twitter users that is not made available to the public. As such, the authors' close ties to Twitter were pivotal to the construction of a complete dataset that sidestepped the issue of path compression affecting the data collected via the Standard API.

In total, this dataset contains about 126,000 cascades made up of over 4.5 million tweets. Surprisingly, however, 97% of these cascades had size less than 50, so for our analysis we discarded them as we did for the networks in the POL/BIAS dataset. Last, we performed class balancing, extracted network features for the baseline models[7], and split the examples into training and test sets. The class sizes for the balanced and unbalanced datasets in the VER dataset are given in Table 2.3, while basic descriptive statistics for the networks in all the tasks can be found in Table 2.4. Note the difference in longest paths between the POL/BIAS and VER data; this is illustrative of the path

[7] As described in Section 3.1.

compression associated with the Standard API.

Table 2.3: Class Sizes for Balanced and Unbalanced VER Datasets

Label	Unbalanced	Balanced
Fake	2,995	468
Real	468	468

Table 2.4: Descriptive Statistics for Networks By Task

Statistic	POL	BIAS	VER
Mean # Nodes	435	461	636
Mean # Edges	330	350	635
Max # Nodes	18,095	17,736	30,967
Max # Edges	21,908	19,546	30,966
Total # Nodes	1,153,620	849,162	2,202,468
Total # Edges	875,160	644,700	2,199,005
Median Longest Path	2	2	4
Max Longest Path	3	3	17

The authors' use of six fact-checking websites to collect the largest fake news dataset of its kind exposes a critical problem in the domain of misinformation prediction: the state of the art for deciding conclusively if a story is true or false is hoping that one of these websites invests time and money in an individual to manually label a rumor. This observation further motivates our work, which hopes to develop an automatic procedure to make veracity prediction both instant and free– a vast improvement over the human-driven procedure.

Artificial intelligence is no match for natural stupidity.

Albert Einstein

3 Supervised Learning Models

This work focuses on two methodologies for extracting information from retweet networks in order to make accurate predictions about the articles they surround. In the first, we tabulate per-network feature vectors that correspond to standard statistics about the nodes and their connections. Using these data, we tune logistic regression, support vector machine, random forest, and multilayer perceptron models to establish a set of baseline performance results 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 networks in the dataset in order to learn a linear separation function in the kernel space.

Our analysis considers six datasets, as described in Section 2.2. We apply the network feature and graph kernel methodologies to each.

3.1 Network Features

For a network $G = (V, E, \Sigma)$, we compute a set of 14 network features, concatenating them into a vector x that reflects attributes of the shape of the network. These features are enumerated as $x_1 \ldots x_{14}$ and fall into several groups.

To begin, observe some basic definitions relating to node and edge statistics: let n and m be the *number of nodes and edges* ($|V|$; $|E|$), respectively. Let a node v's *neighborhood* $\mathfrak{N}(v)$ be the set of nodes it points to via directed edges: $\mathfrak{N}(v) = \{v'|(v, v') \in E\}$. Let the *in-degree* of v be $deg^{-}(v)$, defined as $|\{v'|(v', v) \in E\}|$, or the number of edges that terminate at v. Let the *out-degree* of v $deg^{+}(v)$ be defined analogously as $|\{v'|(v, v') \in E\}|$, or the number of edges that emanate from v. Note that this is equiv-

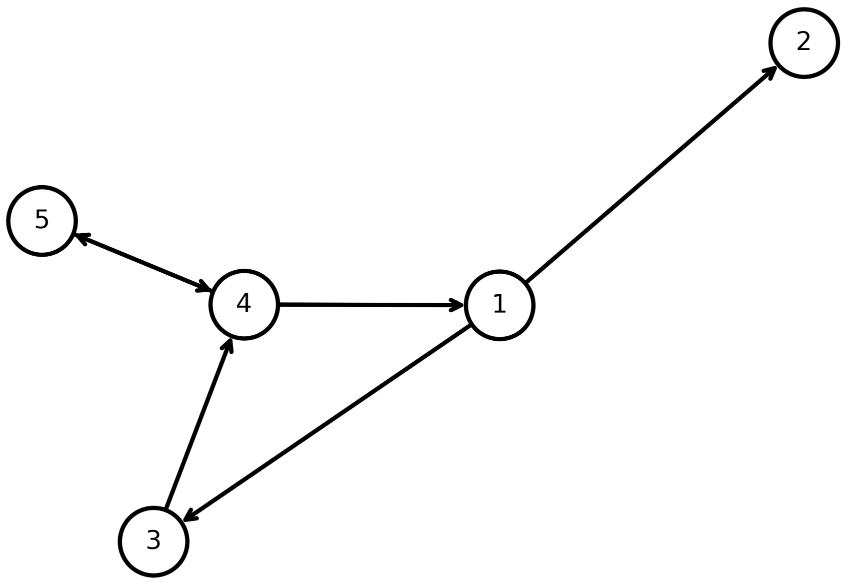


Figure 3.1: Network 1

alent to the size of the neighborhood of v, or $|\mathfrak{N}(v)|$. Last, let the *degree* of a node be the sum of its in- and out-degrees: $deg(v) = deg^{-}(v) + deg^{+}(v)$.

Features x_1, x_2, and x_3 correspond to basic statistics about the network: x_1 is the number of nodes n and x_2 is the number of edges m. x_3 is the *density* of the network, a statistic that proxies the level of connectedness among the nodes. The density is a ratio of the number of edges to the number of potential edges given the number of nodes in the network. Density is thus calculated as

$$x_3 = \frac{2m}{n(n-1)} \tag{3.1}$$

For example, in Network 1, Node 4 has an in-degree of 2 and an out-degree of 2 (noting its bidirectional connection with Node 5), and the network as a whole has the following basic features: $x_1 = 5$, $x_2 = 6$, $x_3 = \frac{6}{10} = 0.6$.

The next feature relates to a network's *assortativity.* Assortativity refers to the extent to which similar nodes are connected. For example, in a social network, high assortativity indicates that gregarious people tend to talk to popular people, whereas quiet people talk (albeit less) to fellow loners. Though the first application of assortativity was the epidemiological study of the spread of disease within networks, recent work has found that it tends to be important vis-à-vis the propagation of Internet content as well(Yao et al., 2017) [1].

The standard metric of network assortativity is the Pearson correlation between the degrees of nodes that connect to each other (Newman, 2003). Define e_{ij} to be the fraction of edges in the network that connect nodes of out-degree $i-1$ to nodes of in-degree $j-1$, noting that over all i and j these values compose a joint probability distribution. We offset i and j by one because we are interested in the *excess degree* of each vertex, i.e., the number of nodes that it connects to outside of the current edge being counted. Then let $\alpha_i = \sum_j e_{ij}$ and $\beta_j = \sum_i e_{ij}$. These are the proportions of

[1]Assortative network architectures tend to exhibit slow but robust cascades, making information spread difficult to stop even when certain producers of content are cut out of the network after the original publication. On the other hand, disassortative networks spread information faster, but are easier to "immunize" by removing certain producers from the network in order to quash a cascade in progress.

edges that begin and end at nodes of out- and in-degree $i-1$ and $j-1$. As they each define univariate probability distributions, their standard deviations σ_α and σ_β are also easily obtainable. Finally, the assortativity coefficient, termed x_4 in our network feature vector, is calculated as

$$x_4 = \frac{\sum_{ij} ij(e_{ij} - \alpha_i \beta_j)}{\sigma_\alpha \sigma_\beta} \tag{3.2}$$

Feature x_4 is high if similarly popular nodes tend to connect and low if nodes that connect tend to be different in popularity.

Features x_5 through x_{14} relate to two ways of describing a network's *centrality.* The *degree centrality* of a network is associated with the distributions of in- and out-degree centrality of its nodes. One node v's in- and out- degree centralities are calculated as

$$D^{-}_{cen}(v) = \frac{deg^{-}(v)}{n-1} \tag{3.3}$$

$$D^{+}_{cen}(v) = \frac{deg^{+}(v)}{n-1} \tag{3.4}$$

For example, the out-degree centrality of Node 4 in Network 1 is $\frac{2}{5-1} = 0.5$, or the proportion of other nodes in the network that Node 4 connects to.

As such, features x_5 through x_8 are tabulated as the mean, max, standard deviation, and skew of the in-degree centrality distribution over all the nodes, while x_9 through

Table 3.1: Out-degree centrality statistics for Network 1

Node	Out-Degree Centrality
1	0.5
2	0
3	0.25
4	0.5
5	0.25

Feature	Statistic
Mean (x_9)	0.3
Max (x_{10})	0.5
Std. (x_{11})	0.19
Skew (x_{12})	-0.34

x_{12} are the corresponding statistics for the out-degree centrality distribution. Table 3.1 gives the out-degree centrality statistics for Network 1, corresponding to x_9 through x_{12} for that network.

Out-degree centrality statistics describe how powerfully the nodes can push information through the network, while in-degree centrality statistics denote the propensity of the nodes for learning information that is moving through the network. For example, a high max out-degree centrality value indicates the existence of an important influencer in the network who might be able to trigger a large cascade independently, and a low standard deviation of in-degree centrality suggests that every node is about equally likely to receive a piece of information as every other node.

Another metric for measuring centrality is *closeness centrality.* Terming the length of the shortest path between nodes v and u the *distance* $d(v,u)$ between them, the closeness centrality of a node v is

$$C(v) = \frac{n-1}{\sum\limits_{u \in V} d(v,u)} \tag{3.5}$$

or the sum of reciprocal distances to each other node, normalized by the number of shortest paths it takes into account. x_{13} is calculated as the average closeness centrality of a network:

$$x_{13} = \frac{\sum\limits_{v \in V} C(v)}{n} \tag{3.6}$$

x_{14} is the standard deviation of the distribution of node closeness centralities:

$$x_{14} = \sqrt{\frac{\sum\limits_{v \in V} (C(v) - x_{13})^2}{n-1}} \tag{3.7}$$

Having constructed x as this 14-dimensional vector, the baseline performance models comprise a diverse set of classification techniques. For all three tasks, logistic regression (LR), support vector machines (SVM), random forests (RF), and multilayer perceptrons (MLP) are trained, representing a wide variety of model types and complexities (Maroco et al., 2011). Implementation details, such as methods for hyperparameter tuning, may be found in Section 3.3.

3.2 Graph Kernels

3.2.1 Overview

Graph kernels are algorithms for computing topologically rich similarity metrics between networks. 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, incorporating information from both small and large neighborhoods of nodes. Consider the task of determining the similarity between the two small networks shown in Figure 3.2 (Feragen, 2013). While to the human eye the networks seem nearly identical, it is a non-trivial problem to compute network similarity even at this small scale. Indeed, it is unclear which standard metrics could be computed over the two networks in Figure 3.2 to give insight into their near-isomorphism. Graph kernels move beyond tabulating these traditional features 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 a graph kernel, the resulting matrix may be fed into a kernelized learning model to compute a linear separation between networks of different types.

Formally, a graph kernel implicitly represents a network $\mathcal{X} \in \Omega$ as a vector $\phi(\mathcal{X})$ in a reproducing Hilbert space $\mathcal{H}$ (Scholkopf & Smola, 2001). A graph kernel $k : \Omega^2 \to \mathbb{R}$

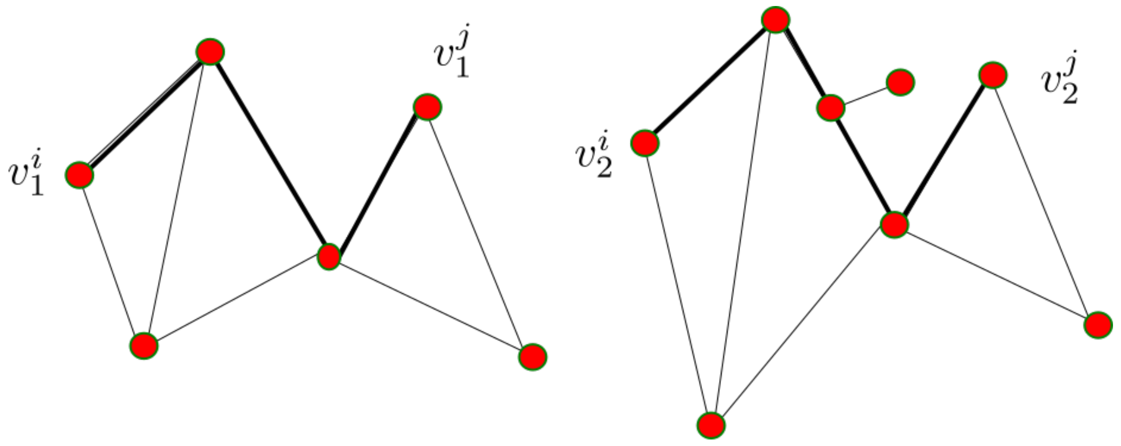


Figure 3.2: Graph kernels compute a rich comparison between two networks that takes into account the shape of the networks, something that naive feature models are unable to do.

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:

$$k(\mathcal{X}, \mathcal{X}') = \langle \phi(\mathcal{X}), \phi(\mathcal{X}') \rangle_{\mathcal{H}} \tag{3.8}$$

This inner product represents a similarity metric between two networks. Intuitively, a graph kernel attempts to find topology-informed representations of two networks in $\mathcal{H}$ and determine their geometric distance from each other. The larger $k(\mathcal{X}, \mathcal{X}')$ is, the more similar $\mathcal{X}$ and $\mathcal{X}'$ are.

To use a graph kernel in a classification model, we arrange our data as a sequence of networks $G_1, \ldots, G_n$ and construct a matrix K such that $K_{ij} = k(G_i, G_j)$. This matrix is the basis for kernelized learning techniques like support vector machines (Cortes &

Vapnik, 1995), Gaussian Processes (MacKay, 1997), and Kernel PCA (Scholkopf et al., 1998).

Graph kernels generally have one of three flavors, corresponding to their core network algorithm: random walks, shortest paths, and subtrees. Paths on the Twitter networks are very short (the median longest path is between two and four over all datasets), so we would expect random walk and shortest path approaches to perform poorly due to their inability to encode information about networks 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 each network's structure despite the relatively short average paths thereof.

3.2.2 The Weisfeiler-Lehman Kernel

We apply the Weisfeiler-Lehman kernel (Shervashidze et al., 2011) to the problem of content-blind social network classification. The WL kernel is a subtree-based approach that measures the similarity of labeled networks 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 networks, which it applies sequentially to compute a metric for how "close to" (or far from) isomorphism two networks are.

The computation of the WL kernel begins with a network $G = (V, E, L) \in \Omega$ and the choice of a number of iterations p. We proceed by iteration, with each indexed by i. Iteration i associates a label $\ell_i(v) \in \Sigma$ and a multiset of strings $M_i(v)$ for each vertex $v \in V$, where $\ell_0(v)$ is set initially to $L(v)$.

In 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 label from the previous iteration, such that $s_i(v) := \ell_{i-1}||s_i(v)$, where $||$ is the concatenation operator. Last, we compress the new $s_i(v)$ by encoding it with a hash $h : \Sigma^* \rightarrow \Sigma$, 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[2]). 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. Notice that at iteration i, the label for each node depends on information about vertices i edges removed from the node. At the end of p iterations, we compute $c_i(G, \sigma_{ij})$, which is a count of the number of times the label $\sigma_{ij} \in \Sigma$ occurs in the labels of G at iteration i. Formally, let the set of labels associated with each vertex of 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 lexically sorted. Then the mapping $c_i : \Omega \times \Sigma \rightarrow \mathbb{N}$ represents the number of times that the label σ_{ij} oc-

[2]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 an integer.

curs in Σ_i. Applied to G and each label in Σ, c_i induces a vector corresponding to the topological features of G:

$$\phi(G) = (c_0(G, \sigma_{00}), \ldots, c_0(G, \sigma_{0|\Sigma_1|}), \ldots c_p(G, \sigma_{p0}, \ldots c_p(G, \sigma_{p|\Sigma_p|})) \tag{3.9}$$

That is, $\phi(G)$ is the concatenated values of the counts for each label at each iteration, a topologically rich encoding of G in $\mathcal{H}$. Then the WL kernel is computed as

$$k(G, G') = \phi(G)^T \phi(G') \tag{3.10}$$

Figure 3.3 (Shervashidze et al., 2011) shows the computation of a WL kernel with $p = 1$ for a small network. A kernel value for a network pair is large if the two sequences of WL merges are alike, i.e., they have similar collections of node neighborhoods. For p large, a high kernel value means that not only are the networks similar around small groups 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 merges are identical for arbitrary iterations.

To calculate the time required to compute $\phi(G)$ as described above, let $m = |E|$ and $n = |V|$, noting that $m \geq n$. For one iteration of the kernel, each node receives and

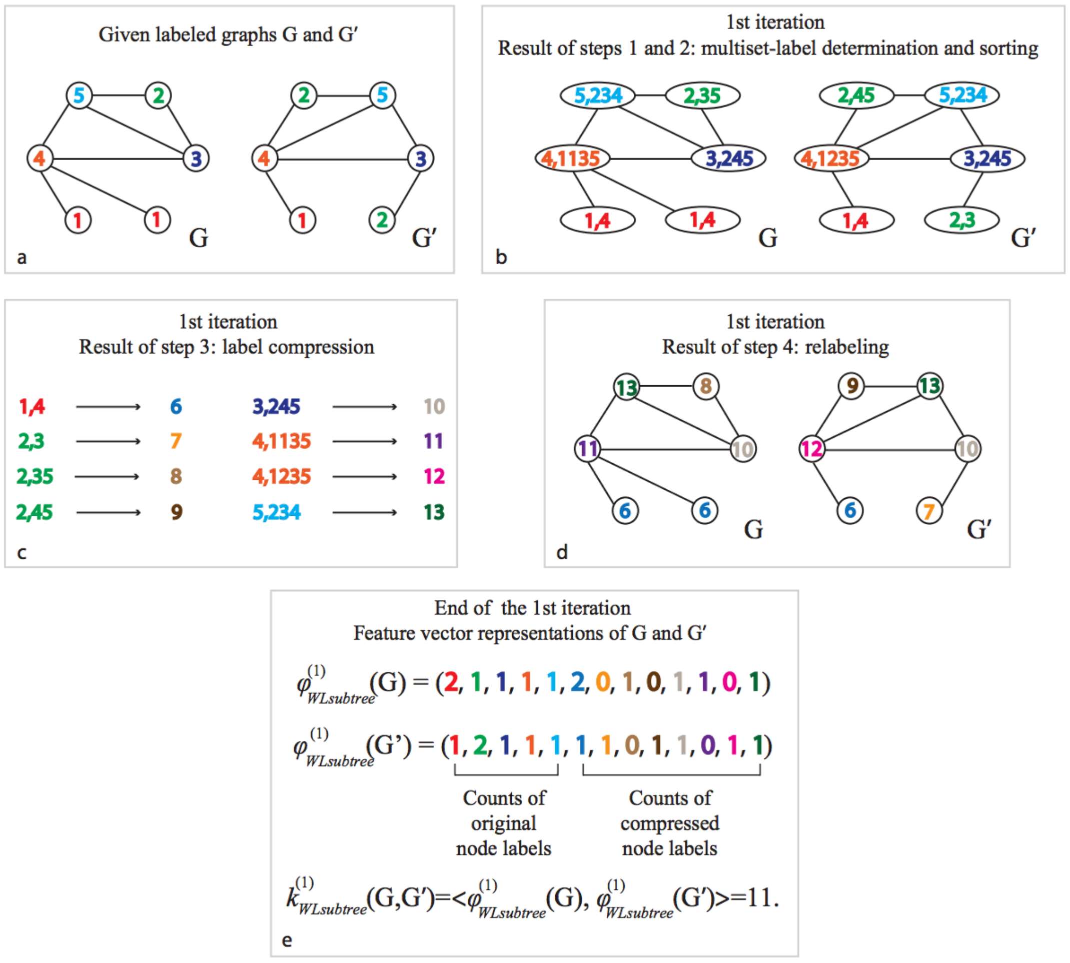

Given labeled graphs G and G′
G
G′
a
1st iteration
Result of steps 1 and 2: multiset-label determination and sorting
b
1st iteration
Result of step 3: label compression
1,4 → 6
2,3 → 7
2,35 → 8
2,45 → 9
3,245 → 10
4,1135 → 11
4,1235 → 12
5,234 → 13
c
1st iteration
Result of step 4: relabeling
d
End of the 1st iteration
Feature vector representations of G and G′
$\varphi^{(1)}_{WLsubtree}(G) = (2, 1, 1, 1, 1, 2, 0, 1, 0, 1, 1, 0, 1)$
$\varphi^{(1)}_{WLsubtree}(G') = (1, 2, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1)$
Counts of original node labels
Counts of compressed node labels
$k^{(1)}_{WLsubtree}(G,G')=<\varphi^{(1)}_{WLsubtree}(G), \varphi^{(1)}_{WLsubtree}(G')>=11.$
e

Figure 3.3: Computation of the WL graph kernel for one iteration. The initial labels are merged along the edges for each node, sorted, then finally compressed (hashed) so that the set of total labels increases in size. The final representation of a network is a vector constructed by counting the number of occurrences for each label. The kernel is then computed as the inner product of the vectors associated with the two networks. Figure from Shervashidze et al. (2011).

sorts its neighbors' labels. Because the list of all labels is finite and known, bucket sort can be used to order each node's multiset of neighbors' labels. There are n buckets, one for each label, and m elements to sort, i.e., one label passed along each edge. The sorting time is thus $O(n + m)$. The time to run p iterations of the kernel is thus $O(p(n + m)) = O(mp)$.

The naive implementation of the WL graph kernel is both slow and exceedingly wasteful. At first glance, the calculation of k for all pairs of N networks for p iterations seems to be an $O(N^2mp)$ algorithm. Worse, there are exponentially many possible label sets that could result from a merge, meaning that the space complexity to store $\phi(G)$ could be as high as $O(2^n)$. However, the implementation of two important optimizations ameliorates both issues. First, instead of representing $\phi(G)$ as a vector with exponentially many dimensions, it is stored as a hashmap from (iteration, vertex) pairs to nonzero values. The space required to store this sparse $\phi(G)$ is $O(np)$: each of n nodes will have exactly p labels, one assigned at the end of every iteration[3]. These sparse vectors support linear time addition and multiplication, as such operations merely involve n constant time lookups. Second, $\phi(G)$ and $\phi(G')$ are not computed lazily for each pair of networks. Rather, values of $\phi(G)$ are cached for each network in the dataset. These vectors are then stacked to form the sparse design

[3]The actual number of entries in the sparse vector will be weakly less than np because nodes with the same label will be counted in one entry.

matrix Φ (of worst-case dimension $N \times np$) before computing the kernel matrix as $K = \Phi^T\Phi$. Thus, computing the kernel over all pairs requires time $O(Nmp + N^2np)$ and space $O(Nnp)$, a vast improvement over the naive implementation's runtime and space requirement.

3.2.3 Node Labels

Assigning initial labels to nodes is a three-step process. First, a *labeling scheme* assigns a *value* to a node. Then, nodes of similar values are grouped into *buckets*. Finally, nodes are given *labels* based on their assigned buckets.

A labeling scheme is a way to allocate initial values to nodes in a network. A well-chosen labeling scheme is key to the performance of the WL kernel. We experiment with two types of labeling schemes: uninformative and informative. *Uninformative* labeling schemes use only the information inherent to the definition of the retweet network, for example assigning a node the value corresponding to its out-degree. *Informative* labeling schemes use a statistic associated with the user who published the tweet represented by a node, for example the number of followers or followees that the user has. In order to preserve content-blindness, care is taken with informative labeling schemes to ensure that individual users cannot be identified by their labels. This is done by assigning the same label to nodes with similar values, a procedure known as bucketing.

Nodes are assigned to one of several variable-size buckets based on their values within the labeling scheme. For example, for an informative labeling scheme that assigns values based on the number of followers a user has, we choose a bucket size B and assign each node v the label

$$L(v) = \left\lfloor \frac{Fol(v)}{B} \right\rfloor \tag{3.11}$$

This means that nodes whose users have between 0 and $(B - 1)$ followers will share a label, nodes whose users have between B and $(2B - 1)$ followers will share a label, and so on. Decreasing B makes each node more distinguishable from its peers, allowing the model to ingest richer information from the network. However, increasing B reduces the number of distinct kinds of nodes, which may mitigate overfitting. Bucketing therefore allows for tunable regularization for labeled kernel models.

In the case of an informative labeling scheme, bucketing also enforces content-blindness by ensuring that a model never has any user-identifying information available to it. Recalling that nodes correspond to tweets, a user could conceivably be identified by a model by matching the labels of two nodes within the same labeling scheme. Given the bucketing procedure, this would be possible if, for example, a bucket were to contain only one node. However, given the kernel parameters chosen by cross-validation, the bucket with the smallest number of nodes in the training set contained 13 examples. This provides good protection against the model being able to single out individual users at test time, thus preserving content-blindness. Further evidence that the model learns topological features rather than user-identifying features is presented in Section 4.2.

One final concern is that if a single user strongly associated with one tag (e.g., fake) participated in many networks across the training and testing sets, then their node label might be used to trivially predict tags for the test set networks in which they participate. A WL kernel that implicitly re-identified users like this would see optimal performance at $p = 0$, calculating the kernel matrix solely from the initial labels without having to look at network connections. However, our results (as described in Section 4.2) demonstrate that ideal kernels run for up to four iterations, learning rich topological encodings at the large neighborhood level rather than stopping before they combine information from adjacent nodes. As such, both the theory and empirical results indicate that kernels employing informative labeling schemes maintain content-blindness, using the labels as topological building blocks.

For the POL and BIAS tasks, we report results only for a labeling scheme corresponding to a node's degree due to lack of access to user-level information[4]. For the

[4]Twitter APIs impose heavy restrictions on public access to information about users' followers, meaning that for POL and BIAS we could not construct informative labeling schemes. The VER dataset includes user-level information returned by an internal API, so our analysis on the VER task includes

VER task, we experiment with a wide variety of informative and uninformative labeling schemes, including those based on followers, followees, user engagement scores[5], and the age in days of a user's account.

3.3 Implementation

The implementation of the models in this work is in Python 3.5, making use of methods from the networkx (Hagberg et al., 2008) package and Sugiyama et al. (2017). Random forest, logistic regression, and SVM classifiers drew on functions from scikit-learn (Pedregosa et al., 2011), while the neural networks were implemented in PyTorch (Paszke et al., 2017).

The train/test split was executed by shuffling the data, then selecting the last 20% to be the test set. Tuning across regularization constants, penalty functions, and other hyperparameters shown in Table 3.2 for all models was done using 10-fold cross validation on the training set. Parameters were updated via the Adam optimizer (Kingma & Ba, 2014) with at most 1000 epochs for the neural network and via the LIBLINEAR solver (Fan et al., 2008) with at most 100 epochs for LR, with training halting via early stopping when appropriate. Modules that depend on random states were seeded deterministically for consistency. The parameter space was explored by grid search, with

experimentation with several informative labeling schemes.

[5]Number of user tweets, retweets, replies and favorites divided by the account's age in days.

Table 3.2: Hyperparameters Tuned

Model	Parameter	Description	Values Tested
LR	C	Regularization Constant	0.0001-100
	Penalty	Regularization Type	L1, L2
SVM	λ_b	Regularization Constant	0.0001-100
	k	Kernel Function	RBF, Linear
RF	n_{est}	Number of Estimators (Trees)	100-10,000
	max-depth	Decision Tree Max Depth	1-100
	max-features	Max Features Per Split	1-14
MLP	σ	Nonlinearity	sigmoid, tanh, ReLU
	H_{sz}	Hidden Layer Sizes	(20)-(100); (10, 10)-(50, 50); (5, 5, 5)-(20, 20, 20)
WL	p	Number of Iterations	0-10
	S_ℓ	Labeling Scheme	Followers, Followees, Age, Followers-Followees, Engagement, Deg., Unique
	λ_{wl}	SVM Regularization Constant	0.0001-100
	B	Bucket Size	1-10,000

candidate values for each parameter listed in Table 3.2.

A lie can travel halfway around the world

while the truth is putting on its shoes.

Charles Spurgeon

4 Results

We refer to the political, bias, and veracity tasks with and without class balancing as POL-B, POL-U, BIAS-B, BIAS-U, VER-B, and VER-U, respectively.

4.1 Predictive Performance

4.1.1 Metrics

To analyze the predictive performance of each model, we report two standard classification metrics: accuracy and F1 score. While accuracy – the proportion of articles tagged correctly – is the most straightforward, the F1 score tells a more complete story and may be more important in determining the real-world applicability of a model. This is especially true in the VER task, where the implications for tagging a real story as fake and a fake story as real are particularly fraught. The F1 score is calculated as the geometric average of two important metrics, precision and recall. Letting TP, FP, and FN be the number of true positives, false positives, and false negatives, respectively, precision ($prec$) and recall (rec) are calculated as:

$$prec = \frac{TP}{TP + FP} \tag{4.1}$$

$$rec = \frac{TP}{TP + FN} \tag{4.2}$$

In the VER task, for example, precision measures the proportion of stories that were indeed fake out of all the stories that the model tagged as fake. Recall measures the proportion of the fake stories in the full dataset that the model was able to correctly tag as fake. Intuitively, low precision indicates that a model is tagging many true stories as fake, and low recall indicates that a model is tagging many fake stories as true. Integrating the two, the F1 score is calculated as:

$$F1 = 2 \cdot \frac{prec \cdot rec}{prec + rec} \tag{4.3}$$

4.1.2 Experimental Results

POL and BIAS accuracy and F1 score for the network feature baselines and the WL kernel are reported in Table 4.1. The WL kernel significantly outperforms the baseline accuracy and F1 on the balanced and unbalanced versions of both tasks. 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 methods. Notably, the 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 idiosyncrasies in the class distribution.

Shown in Table 4.2, results from the VER task are similarly positive. On the balanced dataset, the graph kernel model improves significantly on the network feature baseline, discriminating between true and false stories with 84% accuracy. While the

Task	Model	acc	F1
BIAS-B	LR	0.65	0.65
	SVM	0.70	0.69
	RF	0.69	0.68
	MLP	0.71	0.62
	WL	**0.93**	**0.94**
BIAS-U	LR	0.80	0.88
	SVM	0.77	0.87
	RF	0.82	0.91
	MLP	0.77	0.87
	WL	**0.90**	**0.93**
POL-U	LR	0.67	0.66
	SVM	0.68	0.66
	RF	0.67	0.65
	MLP	0.67	0.64
	WL	**0.76**	**0.72**
POL-B	LR	0.74	0.52
	SVM	0.71	0.49
	RF	0.74	0.57
	MLP	0.74	0.58
	WL	**0.82**	**0.74**

Table 4.1: Classification performance for the POL and BIAS tasks. The best performer for each problem and for each metric is **bolded**.

baseline models had trouble on the unbalanced dataset, which was skewed 87%-13% towards fake stories, the WL kernel performed at 93%– far better than a naive model that predicts all positives.

This is the first set of results for fake news prediction based solely on network topology. Because we are the first to perform classification on the dataset compiled by Vosoughi et al., there is naturally no baseline against which to directly compare these performance results. For context, however, previous work has developed content-aware models that look at both article text and author publication history. These systems have been able to classify rumors as true or false with between 78% and 89% accuracy (Liu et al., 2015; Ma et al., 2016; Ruchansky et al., 2017; Yu et al., 2017). The WL kernel model is thus fully competitive with content-aware models while preserving the advantages associated with content-blindness[1]. This novel result suggests that real-world content classification systems could benefit from the inclusion of topological models.

[1]The benefits of content-blind systems are explained in more detail in Section 4.5.

Task	Model	acc	F1
VER-B	LR	0.58	0.58
	SVM	0.59	0.59
	RF	0.66	0.67
	MLP	0.62	0.66
	WL	**0.84**	**0.85**
VER-U	LR	0.79	0.51
	SVM	0.81	0.53
	RF	0.86	0.63
	MLP	0.86	0.65
	WL	**0.93**	**0.77**

Table 4.2: Classification performance for the VER task. The best performer for each metric is **bolded**.

4.2 Label Tuning Results

For the POL and BIAS task, one iteration of WL was sufficient for high predictive power. Recall that at iteration $i = 1$, the label for a node v is

$$\ell_1(v) = h(L(v)||s_1(v)) \tag{4.4}$$

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 these results demonstrate), then even the compressed shapes of the local neighborhood structures around single nodes in a retweet network are highly predictive of the article that the network surrounds. Moreover, the result lends weight to

Table 4.3: VER Performance by WL Kernel Parameters

Kernel Parameters				Task Accuracy	
Labeling Scheme	Bucket Size	Informative	WL-p	VER-B	VER-U
Unique	–	×	1	0.51	0.86
Degree	1	×	2	0.61	0.87
Degree	10	×	2	0.60	0.85
Followers	100	✓	3	0.84	0.91
Followers-Followees	100	✓	4	0.81	0.93
Account Age	100	✓	3	0.79	0.92
Account Age	1,000	✓	4	0.77	0.90

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 networks. Because average paths are short in these networks, however, graph kernels based on random walks or shortest paths are expected to underperform subtree kernels.

For the VER task, the choice of initial labels for the WL kernel was vital to the performance of the model. Table 4.3 describes performances for informative and uninformative labeling schemes on test data, along with their corresponding bucket sizes. The unique label baseline[2] performs at-chance since it does not encode any topological information, while the degree labels perform slightly better than at-chance, since they begin to incorporate information about the structure of the network. Indeed, labels serve to divide tweets into "types" such that the kernel can associate patterns of

[2]This uninformative labeling scheme assigned a different label to each node in the dataset, effectively erasing all topological information.

interactions between tweets of different types with the networks that surround real and fake stories. For example, the uninformative degree labels roughly correspond to the importance of particular tweets in terms of their propensity for being retweeted. Once a kernel takes in such information as a tweeting user's number of followers or the age of their Twitter account, however, it performs far better than both the uninformative kernels and the network feature baselines. The optimal value for the number of kernel iterations was 3 or 4, depending on the labeling scheme. With the VER networks fully uncompressed, the kernels were able to discern topologically meaningful patterns across paths of several edges, in contrast with the result that one iteration was optimal for the other tasks. The highest accuracy for the balanced VER task was obtained by using the Followers labeling scheme with a bucket size of 100, while for the unbalanced task it was Followers-Followees with a bucket size of 100. The relatively small difference in performance between these two parameter sets indicates that they hold similar predictive power, not that the class distribution of the test set affects the optimal labeling.

Tuning the bucket sizes required methodical experimentation. Figure 4.1 illustrates the results of applying a variety of bucket sizes to the four best-performing labeling schemes. The F1 score is relatively low when buckets are both small (since this imposes artificial distinctions between truly similar users) and large (since this erases

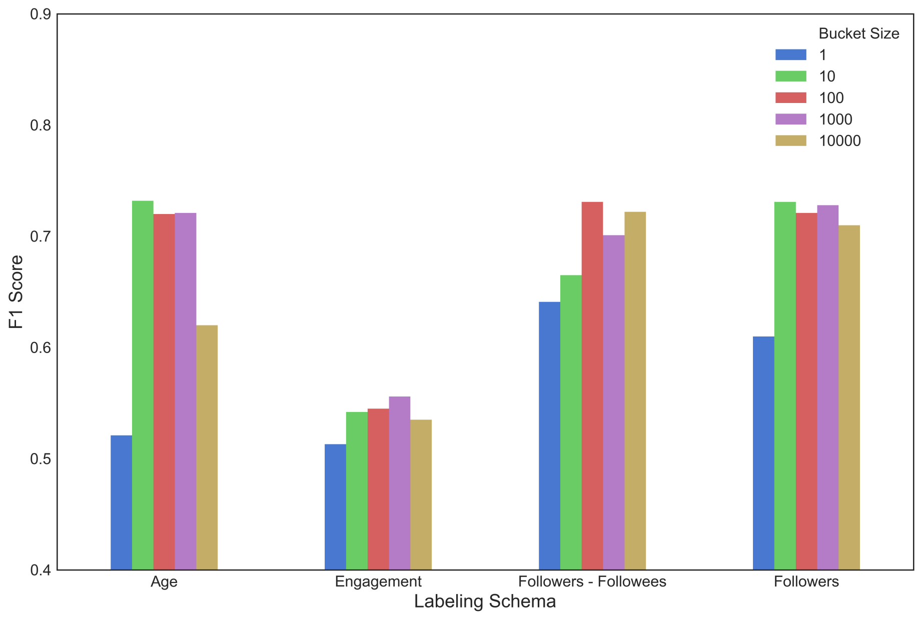


Figure 4.1: VER-U F1 Score by labeling scheme and bucket size. Predictive power peaks somewhere in the middle of the bucket size range for each label, indicating that neither overspecifying nor overgeneralizing user types is helpful for classification.

differences between users, treating them all uniformly). As such, grid search found that choosing bucket sizes around 100 led to optimal performance. This result is also significant because it demonstrates that an ideal model does not "cheat" within the content-blind domain by simply trying to identify individual users by their characteristics. Instead, the kernel makes a bona fide effort to learn the latent types of users in the network, performing best when the space of users is neither too granular nor too coarse.

One challenge inherent to any labeling scheme is that the labels induced are discrete. As such, there is no easy way to interpolate between labels or to combine them meaningfully. Because several labeling schemes have good predictive performance, further work may involve ensemble models that train a separate kernel for each set of kernel parameters and make predictions based on a majority vote across the members of the ensemble.

4.3 Interpretation

To analyze the network features that were most predictive for the highest-performing baseline model (RF), we calculate the Gini importance (Breiman et al., 1984) for each component of the network feature vector. This coefficient roughly corresponds to the decrease in prediction error associated with the inclusion of the feature, averaged

Table 4.4: Random Forest Gini Coefficient, Ranked By Feature Per Task

Rank	POL		BIAS		VER	
1	Max O-D Cen.	0.196	Mean Cl. Cen.	0.163	Mean O-D Cen.	0.081
2	Assort.	0.100	Std. Cl. Cen.	0.132	Mean Cl. Cen.	0.078
3	Mean O-D Cen.	0.088	Std. O-D Cen.	0.100	Assort.	0.059
4	Density	0.083	Assort.	0.082	Std. O-D Cen.	0.056
5	Mean Cl. Cen.	0.072	Mean O-D Cen.	0.064	Density	0.054

across all the trees in the random forest ensemble. With full results reported in Table 4.4, the most important feature for the POL task was the max out-degree centrality, while the mean closeness centrality was most predictive for the BIAS task. In traditional terms, this indicates that the subject matter of an article most affects network features associated with the behavior of high-influence users, while the political leaning of an article affects those features associated with the behavior of average users. Mean out-degree centrality and closeness centrality ranked highest by Gini importance for the VER task and were approximately equal in value, though neither was particularly predictive. The baseline performance results indicate that while there is some signal in basic network statistics, kernel methods that integrate full topology are more powerful. Indeed, Figure 4.2 demonstrates that baseline models had trouble separating networks by their tags.

A similar feature importance analysis for the kernel model is much more difficult due to the dimensionality of the feature vectors associated with the model. While the baseline models took in 14-dimensional feature vectors for each network, the graph kernel regularly computed representations of networks with over 100,000 dimensions. Moreover, because the kernelized learning method does not operate over singular features, isolating specific topological motifs that are highly predictive is intractable. As such, deducing the particular subgraphs that are most associated with each tag is a topic for future work.

To visualize the contrast between the network feature baselines and the kernelized methods, two flavors of principal component analysis (PCA) can be used to compute a dimensionality reduction from the feature space and from the kernel space. First, PCA based on singular value decomposition (SVD) may be used to visualize the feature vectors in two dimensions (Jolliffe, 1986). This method diagonalizes the covariance matrix of the demeaned data to extract a low-dimensional representation of the network features. Next, kernel PCA (Scholkopf et al., 1998) can be used for the WL models. This variation on standard PCA performs the linear operations associated with PCA in the Hilbert space of the kernel. The results of this operation on the test sets are shown for the balanced version of each task in Figure 4.2. The success of the topological models are evident in this dimensionality reduction, with Kernel PCA resulting in a visually interpretable linear boundary between classes for all three problems.

Because of the amount of training data available, the WL kernel model did not reach peak performance on any of the tasks. By varying the size of our training sets, perfor-

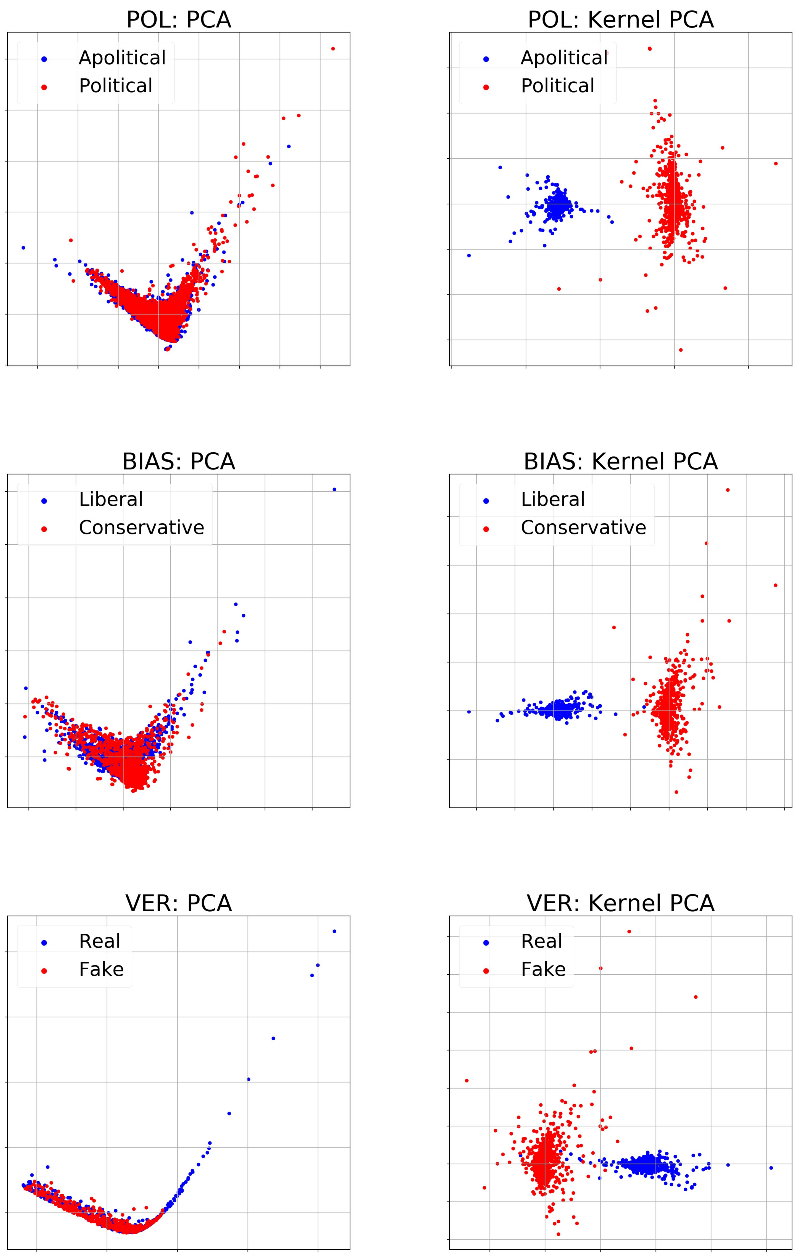


Figure 4.2: PCA and Kernel PCA for POL, BIAS, and VER demonstrate that the topological kernel is able to compute an interpretable linear boundary between networks with different tags (right), while the feature models struggle to extract predictive information from the networks (left).

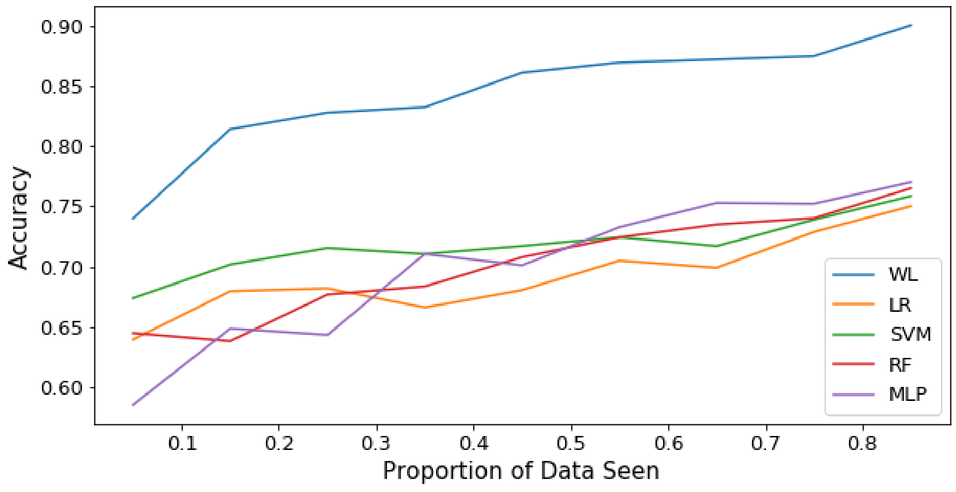


Figure 4.3: Model Convergence: BIAS

mance is observed to increase as the model consumes more training data, but not yet converge. However, a similar experiment demonstrates that it is less obvious that the baseline models have not converged, especially for the VER task. 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 4.3, 4.4, and 4.5 demonstrate the disparity between WL kernel and network feature model convergence for the BIAS and POL tasks, respectively.

4.4 Timing Results

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

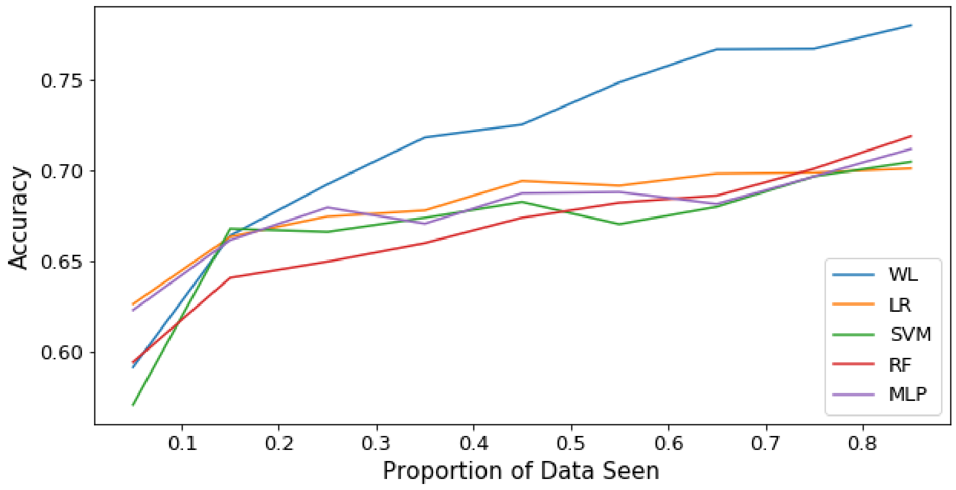


Figure 4.4: Model Convergence: POL

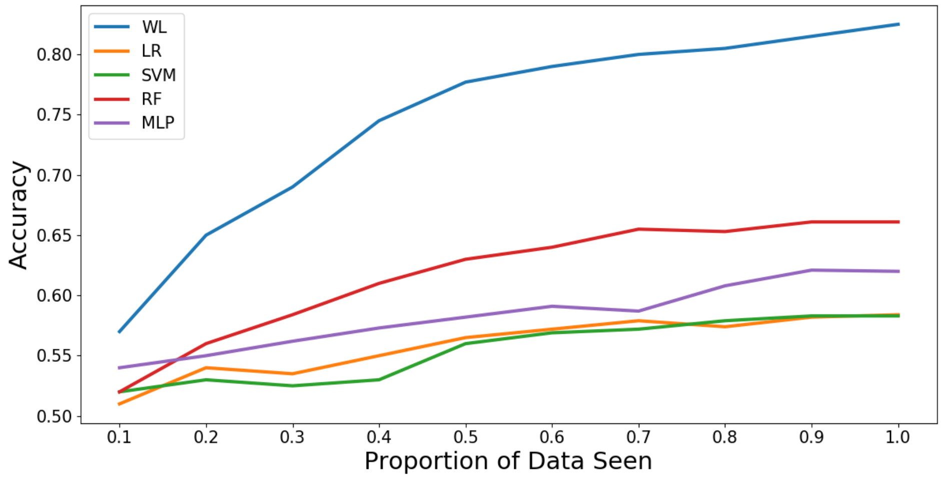


Figure 4.5: Model Convergence: VER

time algorithm (Garey & Johnson, 1990). Thus, many state-of-the-art graph kernels have runtimes that are cubic or worse (Vishwanathan et al., 2010). However, the domain of online content tagging is a good fit for graph kernels since the networks are significantly smaller than those in traditional graph kernel applications like biochemistry, in which networks may have hundreds of millions of nodes (Hall et al., 2017). To test the runtime of the network feature baseline, the feature extraction routine was run ten times for each (unbalanced) dataset and averaged. Timing the graph kernel was done by recording the time taken to complete one iteration of the WL kernel ten times after having assigned initial labels (a trivial operation). Table 4.5 presents runtime results for the model algorithms, showing that the optimized implementation of the WL kernel is significantly faster than the network feature extraction routine. Even when considering that the optimal WL parameters call for three or four iterations, the kernel routine takes much less time than the baseline model. Times for the VER task are higher than times for POL or BIAS because VER networks were larger and denser on average. One reason for the difference in runtime between the kernel and baseline models is that the closeness centrality features in the baseline require the computation of the all-pairs shortest path matrix for a network, an expensive operation that is at least quadratic[3] in n. In contrast, the WL kernel is linear in both n and m, as

[3]Since there are n^2 pairs of nodes, finding the shortest path between all of the pairs (even if finding a shortest path were a constant time operation) is $\Omega(n^2)$.

Table 4.5: Runtime: Total seconds, seconds per network, and seconds per edge to compute network features and one iteration of WL graph kernel. All experiments are performed on a Micosoft Azure NC6 cloud machine running Linux Ubuntu 16.04 with 6 CPUs, 1 NVIDIA Tesla K80 GPU, and 56GB RAM.

task	model	total	per network	per edge ($\times 10^{-5}$)
POL	Network Features	120.8	0.046	3.61
	WL Kernel	19.4	0.007	0.58
BIAS	Network Features	100.8	0.019	3.53
	WL Kernel	15.1	0.003	0.53
VER	Network Features	205	0.167	17.1
	WL Kernel	34.7	0.028	2.87

demonstrated in Section 3.2.2.

4.5 Adversarial Robustness

An important feature of the content-blind domain is that models that abide by its rules are immediately resistant to tampering by malicious actors intent on inducing them to mislabel news stories. Consider an arbitrarily complex natural language model for fake news classification that is not content-blind. By definition, such a model is trained on the language of historical news, analyzing the veracity of a new article by feeding the text of the article through the model to make a prediction. Due to the nature of the model, a producer of content has *sole control* over the factors that will influence the model to describe the content as real or fake. Simply put, the writer of news produces the full breadth of the information that the model can use to determine the veracity of the story. This puts the burden on the model to be resilient enough to perform well in the face of adversarial reverse-engineering.

This dynamic is reminiscent of the battle between producers of spam and the email providers trying to filter it. Early email providers frequently released complex spam classifier systems that relied on advanced natural language processing techniques. Spam producers in turn quickly learned how to fool each one by adjusting the language of their emails in careful ways to generate their profits before the next classifier appeared. Each classifier had a short lifespan because spam writers were in full control of the input to the model, allowing them to write emails whose contents were malicious but that seemed harmless to the classifier (Blanzieri & Bryl, 2008). For spam and social network classification alike, no matter how complicated or precise a content-aware system may be, its fundamental nature pits the strength of the model squarely against the cleverness of the one intent on beating it.

However, a content-blind system for veracity classification deftly sidesteps this issue. Drawing its predictive power from network topology rather than from news text, such a system is much more difficult to disrupt. The shape of a cascade surrounding an article is a loose proxy for the overarching behavior of the crowd as it interacts with the content. As such, an adversary that would be able to arbitrarily alter the input to a content-aware model immediately faces a series of challenges when confronted with a content-blind model. Most important among them is that the adversary would have to orchestrate a coordinated attack in which a critical mass of network participants changes its behavior such that the model mis-classifies the article. This is clearly a much harder task than rewriting the article craftily to fool a content-aware model, instead requiring significant organizational influence within the network.

The success of such models is evident in the spam detection space: modern solutions for spam classification depend on models that operate in a domain tangential to content-blindness. Looking at network relationships between email senders in addition to linguistic content, these systems have more success and longevity (Tseng & Chen, 2009) than their predecessors. This is because the input to such models is not only the text of an email but also a network of relationships involving the sender, which is much more challenging to falsify. Spam senders' networks are topologically different from those of non-spam senders – even if the spam is written carefully to fool a language model – due to the fundamental nature of the spammer's email usage. Spam detection systems that rely on non-content input break the coevolution cycle between classifiers and spam writers, catalyzing the success of spam detection systems in use today (Bhowmick & Hazarika, 2016).

Fake news writers face a similar predicament when faced with a topological model. Recall that the kernels that had the most success in the VER task ran for three or four iterations, meaning that the network motifs most predictive of the various tags involved subgraphs containing at least as many tweets. With the kernels taking into consideration $|V|$ such subgraphs for each network and there being exponentially many configurations of subgraphs, the model ingests topological information at the granularity of thousands of tweets, making these patterns extremely challenging to forge. The content-blind domain thus shifts the advantage from the adversary to the classifier, forcing a malevolent agent to solve a much more difficult problem than rewriting a fake article in a clever way to avoid detection.

Truth is powerful and it prevails.

Sojourner Truth

5 Conclusion

This work introduced a novel set of tasks and methods at the intersection of machine learning, news classification, and network science. Specifically, the content-blind domain contextualized the techniques available for identifying attributes of online articles using nothing but the topologies of the networks that surround them. Constructing two novel real-world datasets of Twitter news networks and performing the first machine learning work on an important third dataset, this research used both network feature models as baseline performance indicators and a fast implementation of the Weisfeiler-Lehman graph kernel to classify news articles across three important dimensions. Tested on both class-balanced and class-unbalanced datasets, the WL kernel outperforms the baseline in accuracy in the POL and BIAS tasks by up to 31% and in runtime efficiency by up to 108%. Most strikingly, the topology of Twitter networks, combined with some basic information about the users involved, was enough to accurately discriminate between real and fake news at 84% accuracy, a score that meets or beats previous work that relies on much richer content-aware training data. These results were then interpreted topologically by reasoning that analyzing Twitter networks at the node-neighborhood granularity is important for content-blind classification and by demonstrating mathematically that the WL kernel does exactly that. Last, the model's robustness to adversarial agents attempting to fool the system was established by way of arguing that the problem such an agent would have to solve becomes intractable within the content-blind domain.

These findings demonstrate that information encoded in the shape of the network surrounding an article is rich enough to inform prediction as to the article's topic, political leaning, and veracity. These findings raise important questions at the heart of online discourse: how might these models be deployed on social media platforms, and what rates of false negatives and positives would be reasonable with respect to a user-facing fake news classification system? Are false negatives worse than false positives? Can we predict other latent factors of news articles, such as sentiment, hate content, or extremism, strictly by their topologies? How hard is it really to coordinate the behavior of thousands of Twitter users to trick a content-blind model?

This work is not meant to suggest that the methodology presented could be used "straight out of the box" in production to boost or diminish the influence of certain articles– or more extremely, to regulate their publishing outright. Obviously, use in the real world would require careful consideration of error and confidence tolerances, failure modalities, and mechanisms of communicating predictions to end users. However, topology does clearly hold significant predictive power and could play an important role in fake news prediction systems in the real world, especially given the advantages inherent in the content-blind formulation. One type of hybrid model that may hold promise would combine elements of content-blindness with sophisticated natural language processing techniques to leverage the advantages and successes of each.

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. Further experimentation is necessary with even larger datasets: one candidate is Facebook, a platform with nearly ten times as many users as Twitter. Development of additional prediction tasks is another important avenue of research; a multi-class topic classification problem that extends our POL task is one possibility. Yet another area of future work 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.

Yet even a perfect fake news detector is useless without a list of candidate stories and an effective way of communicating predictions. Put otherwise, early detection of false content is useless if there is no way to gather candidate articles for consideration, and what good is the knowledge that a story is real or fake if a user will not believe it when told?. The former issue is part of a larger open question about how to identify stories poised to go viral before they reach critical mass, while the latter is a problem tangential to the domain of human behavior. Cheng et al. (2014) find success when trying to predict the future virality of news stories, while Zhao et al. (2015) are able to identify trending unsubstantiated rumors that have not yet been fact-checked.

A combination of those models with the methods developed here may be useful in constructing a successful end-to-end system for real-world deployment. The issue of communicating model predictions to a user is thornier. An individual's implicit biases, political leanings, and demographic profile all affect the ways in which they can be convinced that a model's output is correct. Recent research by Pennycook & Rand (2017) demonstrates that simply telling a user that an article's factuality is disputed results in a negligible decrease in the user's belief in the story. As such, more work is necessary to uncover the underlying motivations of news consumers, including the mechanisms by which people change their opinions and shift from belief to disbelief with respect to a news story.

Despite these challenges, the result that fake news may be discoverable through the topology of its surrounding networks is heartening. It is certainly an academically novel result, linking crowd behavior with network structures that may be disentangled and translated via machine learning. But perhaps more important is its potential for use in slowing the perpetuation of fake news– with the high stakes of truth, democracy and our civic fabric, a small step forward towards a more accurate information ecosystem is truly newsworthy indeed.

References

Allcott, H. & Gentzkow, M. (2017). Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2), 211–36.

Bailey, N. T. et al. (1975). *The mathematical theory of infectious diseases and its applications*. Charles Griffin & Company Ltd, 5a Crendon Street, High Wycombe, Bucks HP13 6LE.

Bakshy, E., Rosenn, I., Marlow, C., & Adamic, L. (2012). The Role of Social Networks in Information Diffusion. *arXiv:1201.4145 [physics]*. arXiv: 1201.4145.

Bhowmick, A. & Hazarika, S. M. (2016). Machine learning for e-mail spam filtering: Review, techniques and trends. *arXiv preprint arXiv:1606.01042*.

Blanzieri, E. & Bryl, A. (2008). A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review*, 29(1), 63–92.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., & Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1), i47–i56.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks.

Castillo, C., Mendoza, M., & Poblete, B. (2011). Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web* (pp. 675–684).: ACM.

Cheng, J., Adamic, L., Dow, P. A., Kleinberg, J. M., & Leskovec, J. (2014). Can cascades be predicted? In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 (pp. 925–936). New York, NY, USA: ACM.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3), 273–297.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9, 1871–1874.

Feragen, A. (2013). Robust geometric graph kernels for bioimaging applications.

Friggeri, A., Adamic, L., Eckles, D., & Cheng, J. (2014). Rumor cascades.

Gabielkov, M., Ramachandran, A., Chaintreau, A., & Legout, A. (2016). Social clicks: What and who gets read on twitter. (pp. 179–192).

Garey, M. R. & Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness.* New York, NY, USA: W. H. Freeman & Co.

Goel, S., Watts, D. J., & Goldstein, D. G. (2012). The structure of online diffusion networks. In *Proceedings of the 13th ACM conference on electronic commerce* (pp. 623–638).: ACM.

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

Hall, R. J., Murray, C. W., & Verdonk, M. L. (2017). The fragment network: A chemistry recommendation engine built using a graph database. *Journal of Medicinal Chemistry*, 60(14), 6440–6450. PMID: 28712298.

Horváth, T., Gärtner, T., & Wrobel, S. (2004). 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: ACM.

Isaac, M. & Ember, S. (2016). For election day influence, twitter ruled social media.

Jolliffe, I. T. (1986). *Principal Component Analysis.* Berlin; New York: Springer-Verlag.

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

Kemp, S. (2017). Number of social media users passes 3 billion with no signs of slowing.

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kriege, N. & Mutzel, P. (2012). Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483*.

Le, Q. & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International Conference on Machine Learning* (pp. 1188–1196).

Liu, X., Nourbakhsh, A., Li, Q., Fang, R., & Shah, S. (2015). Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (pp. 1867–1870).: ACM.

Ma, J., Gao, W., Mitra, P., Kwon, S., Jansen, B. J., Wong, K.-F., & Cha, M. (2016). Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (pp. 3818–3824).: AAAI Press.

MacKay, D. J. (1997). Gaussian processes-a replacement for supervised neural networks?

Maroco, J., Silva, D., Rodrigues, A., Guerreiro, M., Santana, I., & de Mendonça, A. (2011). Data mining methods in the prediction of dementia: A real-data comparison of the accuracy, sensitivity and specificity of linear discriminant analysis, logistic regression, neural networks, support vector machines, classification trees and random forests. *BMC Research Notes*, 4(1), 299.

Mitchell, A., Gottfried, J., Kiley, J., & Matsa, K. E. (2014). Political polarization and media habits.

Molina, B. (2017). Twitter overcounted active users since 2014, shares surge on profit hopes.

Newman, M. E. (2003). Mixing patterns in networks. *Physical Review E*, 67(2), 026126.

Nikolentzos, G., Meladianos, P., Tixier, A. J.-P., Skianis, K., & Vazirgiannis, M. (2017). Kernel graph convolutional neural networks. *arXiv preprint arXiv:1710.10689*.

Parkinson, H. J. (2016). Click and elect: how fake news helped donald trump win a real election.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Pennycook, G. & Rand, D. G. (2017). Assessing the effect of "disputed" warnings and source salience on perceptions of fake news accuracy.

Rainie, H., Anderson, J. Q., & Albright, J. (2017). *The future of free speech, trolls, anonymity and fake news online*. Pew Research Center Washington, DC.

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

Read, M. (2016). Donald trump won because of facebook.

Ruchansky, N., Seo, S., & Liu, Y. (2017). Csi: A hybrid deep model for fake news. *arXiv preprint arXiv:1703.06959*.

Scholkopf, B., Smola, A., & Muller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5), 1299–1319.

Scholkopf, B. & Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, USA: MIT Press.

Shearer, E. & Gottfried, J. (2017). News use across social media platforms 2017.

Shervashidze, N., Schweitzer, P., Leeuwen, E. J. v., Mehlhorn, K., & Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep), 2539–2561.

Sugiyama, M., Ghisu, M. E., Llinares-López, F., & Borgwardt, K. (2017). Graphkernels: R and Python packages for graph comparison. *Bioinformatics*, (pp. btx602).

Tacchini, E., Ballarin, G., Della Vedova, M. L., Moret, S., & de Alfaro, L. (2017). Some Like it Hoax: Automated Fake News Detection in Social Networks. *arXiv:1704.07506 [cs]*. arXiv: 1704.07506.

Tseng, C.-Y. & Chen, M.-S. (2009). Incremental svm model for spam detection on dynamic email social networks. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4 (pp. 128–135).: IEEE.

Varol, O., Ferrara, E., Davis, C. A., Menczer, F., & Flammini, A. (2017). Online human-bot interactions: Detection, estimation, and characterization. *CoRR*, abs/1703.03107.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., & Borgwardt, K. M. (2010). Graph kernels. *J. Mach. Learn. Res.*, 11, 1201–1242.

Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. *Science*, 359(6380), 1146–1151.

Vosoughi, S., Vijayaraghavan, P., & Roy, D. (2016). Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval* (pp. 1041–1044).: ACM.

Yanardag, P. & Vishwanathan, S. (2015). Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1365–1374).: ACM.

Yao, D., van der Hoorn, P., & Litvak, N. (2017). Average nearest neighbor degrees in scale-free networks. *arXiv preprint arXiv:1704.05707.*

Yu, F., Liu, Q., Wu, S., Wang, L., & Tan, T. (2017). A convolutional approach for misinformation identification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (pp. 3901–3907).: AAAI Press.

Zhao, Z., Resnick, P., & Mei, Q. (2015). Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 (pp. 1395–1405). Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee.