A Flexible Approach to Relational Modeling of Social Network Spam

Jonathan Brophy and Daniel Lowd
Department of Computer Science
University of Oregon
{jbrophy,lowd}@cs.uoregon.edu

Abstract

Social media websites face a constant barrage of spam, unwanted messages that distract, annoy, and even defraud honest users. These messages tend to be very short, making them difficult to identify in isolation. Furthermore, spammers disguise their messages to look legitimate, tricking users into clicking on links and tricking spam filters into tolerating their malicious behavior. Thus, some spam filters examine relational structure in the domain, such as connections among users and messages, to better identify deceptive content. However, even when it is used, relational structure is often exploited in an incomplete or ad hoc manner.

In this paper, we present Extended Group-based Graphical models for Spam (EGGS), a general-purpose method for classifying spam in online social networks. Rather than labeling each message independently, we group related messages together when they have the same author, the same content, or other domain-specific connections. To reason about related messages, we combine two popular methods: stacked graphical learning (SGL) and probabilistic graphical models (PGM). Both methods capture the idea that messages are more likely to be spammy when related messages are also spammy, but they do so in different ways – SGL uses sequential classifier predictions and PGMs use probabilistic inference. We apply our method to three different social network domains, each with millions of messages. EGGS is more accurate than an independent model in most experimental settings, especially when the correct label is uncertain. For the PGM implementation, we compare Markov logic networks to probabilistic soft logic and find that both work well with neither one dominating, and the combination of SGL and PGMs usually performs better than either on its own.

1 Introduction

Social spam [10] is any unsolicited or unwanted action by a user in a social network. Many methods have been developed to detect spam based on the content of the messages themselves [6,11,16,31,33,51,53,54], the graph structure among users and messages [1, 7, 13, 20, 35, 38], the timing of user actions [15, 47–49, 51, 55], and more. This works well when there are clear patterns that distinguish spam from non-spam, but can fail when spammers obfuscate their behavior. A complementary approach is to exploit relationships among different users and messages, so that known spammers and spam can be used to identify other spammers and spam [1, 9, 13, 27, 29, 38, 52]. This works well when there are strong predictive relationships linking entities, such as textual similarity among messages or friendship in a social network, but can fail when spammers start new campaigns that are not connected to previously known ones.

In this paper, we integrate these ideas into a flexible method for classifying social spam: Extended Group-based Graphical models for Spam (EGGS). EGGS begins with predictions from an independent classifier, which can use any number of domain-appropriate features. To incorporate relationships among different messages, EGGS defines groups of related messages that have the same author, the same text, or other domain-specific similarities such as the same hashtags. A message can be in multiple groups representing different types of relationships. Related messages are more likely to have the same label, although this probability may depend on the type of relationship. EGGS models these relationships with a probabilistic graphical model, using one of four different approaches: stacked graphical learning (SGL) [26], Markov logic networks (MLNs) [39], probabilistic soft logic (PSL) [3], or a new combination of SGL with either MLNs or PSL. We show that this integrated approach can accurately detect spam on large real-world datasets for multiple domains, requiring very few modifications.

In spite of the breadth of prior work in this area, most other methods fall short in one of three ways:

- They're *specialized for another domain*, such as detecting fake reviews or auction fraud. Social spam has its own structures which are distinct from other adversarial domains. Nonetheless, we can still gen-

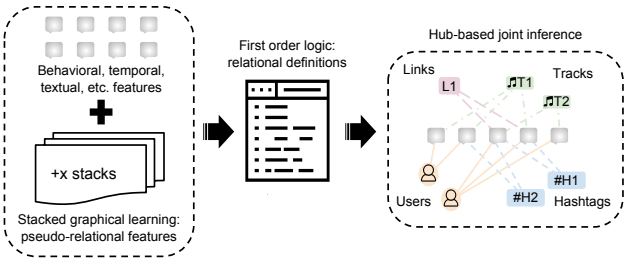

Figure 1: EGGS workflow, showing instantiations of example relational rules (right).

eralize among different social spam domains. In Section 3, we introduce a general framework for social spam and apply it to SoundCloud, Twitter, and YouTube spam.

- They *ignore content and other features*, and only use the relational structure plus a small number of known labels. In Section 5, we show that new social spam is often unconnected to previously seen spam, and that social spam forms many distinct connected components. This means that labels on the training data or a small number of labels in the test data will not cover most spam.

- They *ignore relational structure* and make predictions for each message independently. Our empirical results show that exploiting relational structure can lead to substantially improved performance over independent predictions.

Our primary contributions are EGGS, a flexible method for social network spam that overcomes all three of the above limitations, and the application and evaluation of EGGS on three different domains. As secondary contributions, we show that stacked graphical learning can be combined with other probabilistic models for improved performance, and that relational modeling can scale to large domains using simple methods.

2 Related Work

We give a brief overview of the many techniques used to detect spam from both independent and relational perspectives.

Spam Detection Much of the work done on social spam filtering involves an analysis of some narrow set of specific independent features. Bag-of-words models have been popular ever since the rise of email spam, and continue to see use in Youtube and Twitter models [32, 46, 53]. Word embeddings [41] and LDA topic modeling [46] have also been effective. Numerous works focus on URLs [4, 10, 18, 20, 32, 47, 50, 51], while others look at hashtags [10, 16, 32, 33, 43, 50] and mentions [10, 32, 50]. Each of these have unique advantages, but they can all be seen as content-related features, derived directly from the messages themselves.

User-based features attempt to characterize the behavior of a user in the network with the hope that this will distinguish normal users from malicious ones. A popular approach is to capture the 'burstiness' of user activity [15, 18, 25, 30, 46, 49]. Other approaches look at the number of follows [4, 32], the types and sequences of user actions [13, 49], the ratio of different types of messages users send [18], and account profiles [10, 32]. Graph-based features are conceptually a subset of user-based features, and are derived specifically from a directed graph built using user interactions (e.g. one user following another) on which graph features such as pagerank, betweenness, in/out degree, etc. are computed [13, 18, 32, 50, 51]. Similarly, works such as *CopyCatch* [5] and *Fraudar* [21] use the subgraph density of the graph topology from bipartite user-pages/reviews/products graphs to detect anomalous behavior.

Many of these works use a combination of approaches mentioned above, and Mateen et al. [32] investigate a more comprehensive hybrid approach combining features from all three categories to detect spam on Twitter.

Collective Filtering The works by Pandit et al. and Akoglu et al. [1, 37], *NetProbe* and *FraudEagle*, use Markov random fields (MRFs) with belief propagation (BP) to detect fraudsters on eBay and the Software Market App Store using only the network structure of users and products, with the edges between them representing positive or negative reviews [23]; Akoglu et al. [38] expand upon this with *SpEagle*, generating unsupervised priors for users and products to propagate in the network structure. Li et al. [29] use iterative classification (ICA) between review, user, and IP nodes to detect spammers on Dianping, a network similar to Yelp. These are all unsupervised methods, which are especially useful for detecting opinion spam [23] because true labels are often difficult to obtain for this problem domain; this has the added benefit of not having to train an additional supervised classifier on separate data to generate priors to propagate. For social spam, labeled training data is more prevalent, allowing us to build more powerful priors using supervised classifiers, which are then propagated by our relational methods.

Works on social spam, such as Duan et al. [12], experiment using ICA, BP, and relaxation labeling on the message-message graph connected by similar URLs

and hashtags to classify topics on a small Twitter dataset. Li et al. [28] use typed MRFs, connecting users with URLs and tweet bursts to spot spam campaign promoters on Twitter. Fakhraei et al. [13] leverage user reports using PSL to find spammers in the on-line dating network IfWe; however, the relational rules were only relevant to a small proportion of users, reducing the effectiveness of joint reasoning. Castillo et al. [8] use stacked graphical learning (SGL) to introduce one new relational feature in addition to their original features to better detect spam hosts on the *Webspam-UK2006* dataset.

Matrix factorization methods are a complementary approach. Zhu et al. [55] encode user-user relations from different user interaction types using collective matrix factorization [45] to detect spammers on RenRen. Shen et al. [44] extend this by adding a social interaction coefficient to spot spammers on Twitter. Chen et al. [9] tackle the spam and spammer problems simultaneously using relations between users and bookmarks on the website: delicious.com. Wu et al. [52] take the same approach but use different relations (user-user, user-message, and message-message connected by URLs and hashtags) to work on the platform Sina Weibo.

3 Methodology

We now introduce Extended Group-based Graphical models for Spam (EGGS), our framework for detecting social network spam. The basic approach is to predict the label of each message using standard classification methods, and then refine those predictions using relational reasoning methods on groups of related messages. EGGS is not a single, monolithic method, but a general approach that can incorporate any domain-specific features and relations, any type of classifier, and any type of probabilistic graphical model. In the following, we describe a set of features and methods that work well for detecting social network spam in several domains.

3.1 Independent Modeling

We begin with a standard classifier, which we refer to as the "independent model," since it makes predictions for each message separately. We highlight engineered features shown to work well for spam classification in social networks based on previous research. In addition to the features listed here, it is easy to extend our method to add others for different domains.

Content-based Features We use the text of each message to generate content-based features such as: the number of characters, hashtags, links, and the top 10,000 trigrams (selected based on term frequency) used as binary features (Table 1: Content).

User-based Features We aggregate user actions (although any relevant entity in the domain may be used: users, URLs, tracks, videos, hashtags) to create user-based features (Table 1: User). Features are computed in sequential order of messages based on their timestamp. For example, when computing the feature *UserUploads* for message #100 posted by user x, we record the number of tracks uploaded by user x up until message #100. This creates a more realistic scenario as features are computed only based on previous messages.

Graph-based Features As done in prior work (e.g., [13]), we create graph features using a list of follower actions. We can represent this list of affiliations as a graph, where we construct a node for every user in the list, and add a directed edge from user x to user y whenever user x starts to follow user y. Then we compute the following features on the resulting graph: *Pagerank* [36], *Triangle count* [42], *K-core* [2], *In/Out-degree* [34] (Table 1: Graph). Each feature represents a different aspect of connectivity for a user to the community, and capitalizes on the assumption that spammers tend to be less connected than non-spammers [13], or more connected with other suspicious users.

Table 1: Independent Model Features

Content	
NumChars	# of chars in msg [40]
NumHashtags	# of hashtags in msg [32]
NumLinks	# of links in msg [18, 32].
NumMentions	# of mentions in msg [10].
IsRetweet	1 if msg is a retweet else 0 [22].
Polarity	Msg. polarity [40].
Subjectivity	Msg. subjectivity [40].
N-grams	Top 10,000 tri-grams [46].
User	
UMsgs	# of msgs posted by each user [32].
UHRatio	Frac. of user msgs with a hashtag [32].
UMRatio	Frac. of user msgs with a mention [10].
ULRatio	Frac. of user msgs with a URL [18, 32].
UBlacklist	1 if user posts 3+ spam msgs else 0 [10].
UWhitelist	1 if user posts 10+ ham msgs else 0 [10].
UMsgMax	Max msg length posted by user [40].
UMsgMin	Min msg length posted by user [40].
UMsgMean	Mean msg length posted by user [40].
TMsgs	# of msgs per track.
Graph	
Pagerank	Pagerank of each node in the follower graph G [36].
TriCnt	# of triangles per node in G [42].
KCore	Iteration a node in G was pruned [2].
InDegree	# edges entering a node in G [32, 34].
OutDegree	# edges leaving a node in G [32, 34].

Table 2: Stacking Features

MMSRatio	Frac. of spammy matching msgs.
USRatio	Frac. of spammy msgs per user.
TSRatio	Frac. of spammy msgs per track.
HSRatio	Frac. of spammy msgs per hashtag.
MSRatio	Frac. of spammy msgs per mention.
LSRatio	Frac. of spammy msgs per link.
HSRatio	Frac. of spammy msgs per user hashtag.

3.2 Relational Modeling EGGS uses several types of relational modeling, separately or in combination, to improve on the independent model.

3.2.1 Stacked Graphical Learning Stacked graphical learning (SGL) [26] is a simple approach to performing collective classification, in which the label of each entity depends on the labels of its neighbors. Since the true labels of the neighbors are often unknown, SGL first uses an independent classifier to predict the label of every entity. The predicted labels can then be used to derive features for a second classifier, which makes a refined prediction for each entity. This process can be repeated multiple times, "stacking" classifiers on top of classifiers to any depth.

We apply this idea to social network spam by defining "pseudo-relational" features, each of which summarizes the predicted labels of related messages (Table 2. For example, "USRatio" is the fraction of messages written by the same user which are predicted to be spam. Since most spammers send multiple spam messages, a higher value of USRatio indicates that the message is more likely to be spam. We create pseudo-relational features based on each relation, contrary to the previous application of SGL on webspam where they create only one additional feature: the average predicted label of its neighbors. The learning procedure is similar to previous approaches [8, 26], except we use a holdout method instead of cross-validation when building the sub-model for each stack (Figure 2).

Using K stacks, we evenly split the training data $D = \bigcup_k^K D^k$, and train a base model f^0 on $D^0 = (X_g^0, Y^0)$, where X_g^0 are the original features for D^0. Each subsequent submodel f^k trains on D^k with additional pseudo-relational features:

$$f^k = \begin{cases} train(X_g^k, Y^k) & k = 0 \\ train(X_g^k + X_{p_{k-1}}^k, Y^k) & k > 1 \end{cases}$$

$X_{p_k}^j$ are pseudo-relational features for D^j using predictions from f^k, which we can define as a function of X_g^j and $\hat{Y}_k^j$ (predictions for D^j from f^k):

$$X_{p_k}^j = S(X_g^j, \hat{Y}_k^j)$$

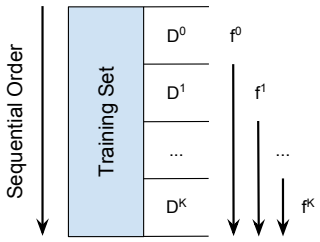

Figure 2: Stacked graphical learning with K stacks using the holdout method.

where

$$\hat{Y}_k^j = \begin{cases} f^k(X_g^j) & j > 0, k = 0 \\ f^k(X_g^j + S(X_g^j, \hat{Y}_{k-1}^j)) & j > 1, k > 0, k < j \end{cases}$$

Then given test data, X', inference is the same as the original SGL [26]:

$\hat{Y}_0' = f^0(X_g')$
For k = 1...K:
$\quad X_{p_k}' = S(X_g', \hat{Y}_{k-1}')$
$\quad \hat{Y}_k' = f^k(X_g' + X_{p_k}')$
return $\hat{Y}_k'$

This holdout method builds submodels in sequential order of the data, while also computing the pseudo-relational features in sequential order. This makes the problem more realistic since we do not ignore the temporal component of the data, contrary to the cross-validation technique. However, this comes with a cost; the cross-validation technique [8, 26] trains each submodel on the entire training set. Thus, more stacks generally does not decrease performance. In our case, the higher K becomes, the less data each submodel has to train on, introducing the possibility of underfit models. In practice, we find that 1-2 stacks works best, which is consistent with previous works [8, 14, 26], even though they use the cross-validation method.

3.2.2 Flexible Joint Inference The growing field of statistical relational learning (SRL) has developed various methods for doing collective classification [19], most promisingly using probabilistic graphical models (PGMs). We experiment with Markov logic networks (MLNs) [39] and probabilistic soft logic (PSL) [3], both of which use weighted formulas in first-order logic to define a template for a PGM. Like stacking, the goal is to improve the predicted label for each message by reasoning about the labels of related messages. However, instead of a sequential pipeline of classifiers, MLNs and PSL define a joint probability distribution

(a) $\neg spam(e)$

(b) $indPred(e) \rightarrow spam(e)$

(c) $hasRel(r, e) \wedge spam(e) \rightarrow spamRel(r)$

(d) $hasRel(r, e) \wedge spamRel(r) \rightarrow spam(e)$

Figure 3: Generalized relational model represented as first order logical rules. (a) Negative prior, (b) positive prior, (c, d) hub-based relational rules; e is short for entity (typically messages or users), r is short for relation.

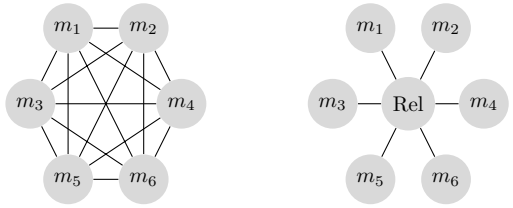

Figure 4: Pairwise connections between related messages (left). Hub-based approach to connect related messages (right).

over all possible labelings and perform probabilistic inference to reason about that distribution.

We first describe the formulas, which capture our intuition about how information should propagate among related messages, and then describe how we use Markov logic and PSL to turn these formulas into a full probabilistic model. Our relational model contains two main components: priors (negative and positive) and relations. The negative prior assumes that all messages are non-spam, which is a fair assumption given the problem domain, while the positive prior gives the model information to propagate (Figure 3 (a,b)). We focus on connecting messages to one another using any relation, but our model may connect entities of any type together, such as users, hashtags, URLs, etc.

The second part defines the relations to exploit from the network structure, grouping related messages together (Figure 3 (c,d)). As a concrete example, let's use matching text as our relation, the intuition being that spammers tend to post comments that are very similar or exactly the same from one or multiple accounts. We can instantiate rules (c, d) as follows:

$$hasText(t, m) \wedge spam(m) \rightarrow spamText(t) \tag{3.1}$$

$$hasText(t, m) \wedge spamText(t) \rightarrow spam(m) \tag{3.2}$$

Now if a spammy message is posted, then the text associated with that message is more likely to be spammy. Thus, the more evidence we have of messages with this text being predicted as spammy, the more confident we are to label the text itself as spammy. Then, if we encounter a message whose predicted label is ambiguous, we can confidently label this new message as spam as it is associated with this spammy text.

Previous work using Markov networks for joint reasoning over a set of related entities typically do so in a direct fashion [48]. For example, we could have written the previous relational rule as:

$$sameText(m_1, m_2) \wedge spam(m_1) \rightarrow spam(m_2) \tag{3.3}$$

This type of modeling provides propagation directly from one message to another, but makes it difficult to scale for large groups of related messages. For example, a group of 100 related messages would need $\binom{100}{2}$ edges to take care of every possible interaction.

Other researchers have defined user nodes with edges to all messages posted by a given user [37, 38, 52], mitigating the problem mentioned above. We extend this notion to work not only with users, but any chosen relation (Figure 1), such as similar text, links, hashtags, etc. This concept of a 'hub node' essentially creates a hyper-edge from a message to all related messages (Figure 4). This significantly reduces the number of edges to be linear in the number of related messages. As information starts to propagate, these 'hub nodes' become more or less 'spammy', which in turn gets propagated to messages that are difficult to detect at first glance, but become more obvious as their connections to other messages are revealed.

We use the outputs of the supervised classifier as message priors for our joint prediction model to propagate, but it is important to note that a message with no relations to any other messages in the dataset will not be affected by this model.

MLN Implementation We first implement our relational model as an MLN, and then convert our MLN into an MRF using the Libra toolkit [24] since belief propagation in Libra is better optimized than inference in existing MLN implementations. We can convert our MLN formulas into equivalent MRF factor potentials between a message node and a hub node (Table 3), where ϵ is tuned separately for each relation and inference is done using loopy belief propagation, similar to prior work on fraud detection [37, 38].

Table 3: Factor Potential Definition per Relation

	Message State	
Hub State	spam	non-spam
spam	1 - ϵ	ϵ
non-spam	ϵ	1 - ϵ

PSL Implementation The second implementation of our relational rules is as a PSL model, which builds a hinge-loss Markov random field (HL-MRF) [3]: a type of log-linear model that uses hinge loss functions of the variable states as features and can be modeled as a conditional probability distribution as follows [13]:

$$P(Y|X) = \frac{1}{Z(\omega)} \exp\Big(-\sum_{i=1}^{n} \omega_i \phi_i(X,Y) \Big) \tag{3.4}$$

where ϕ_i is the set of n continuous potentials:

$$\phi_i(X,Y) = [\max\{0, \ell_i(X,Y)\}]^{p_j}, \tag{3.5}$$

ℓ is a linear function of X and Y, and $p_j \in \{1,2\}$. We can learn the weights to these rules from data using gradient descent and expectation maximization [3].

Both of these models propagate information among the same sets of related messages, but there is an important difference: loopy belief propagation in MLNs can combine uncertain prior probabilities to arrive at more confident posterior probabilities. For example, if n messages all the have the same prior of 0.85, their posterior scores can be 'pushed' beyond 0.85, and this effect increases as n increases. On the contrary, spam scores in the PSL model stop increasing once their distance to satisfaction specified in (3.5) reaches zero.

4 Data

We evaluated our methods on spam data from three social networks: SoundCloud, Twitter, and YouTube.

SoundCloud SoundCloud is an online music sharing network where users can upload original tracks that other users can listen to and comment on. The dataset includes all comments posted from October 10, 2012 to September 30, 2013, on approximately 8M tracks. The basic statistics of this dataset reveal just how imbalanced the class labels are distributed throughout the comments compared to the other two datasets (Table 4).

YouTube YouTube is a video sharing service similar to SoundCloud, except users post comments to a specific video instead of a track. The data[1] was collected from October 31st, 2011 to January 17th, 2012, focusing on the most viewed and top-rated videos [35]. This dataset contains no user subscription attributes (analogous to followers in SoundCloud and Twitter), as the data collectors were often restricted from this information, and did not want this fact to cause inaccuracies in their experiments [35].

Twitter Twitter is a social network that allows users to post short messages to one another. We use the *HSpam14 Dataset*[2], curated from May 1, 2013 to June 31, 2013 with a hashtag oriented focus [43].

Table 4: Basic Statistics per Domain

Entity	SoundCloud	YouTube	Twitter
messages	42,783,305	6,431,471	8,845,979
spam	684,338	481,334	1,722,144
users	5,505,634	2,860,264	4,831,679
spammers	128,016	177,542	843,002
follows	335,000,000	N/A	128,000,000

5 Evaluation

We evaluate the effectiveness of EGGS at finding spam in three social network domains. We use the area under the precision-recall curve (AUPR) as our main metric. AUPR measures the ability to find most of the positive examples (spam) without too many negative examples (non-spam). With highly imbalanced classes, as in our SoundCloud dataset (1.6% spam), AUPR is usually a better way to differentiate between classifiers than alternatives such as area under the ROC curve.

Social network spam evolves rapidly, so a method that works well one month may work poorly the next. We incorporate this into our experiments by creating ten test sets for each domain, partitioned chronologically. Predictions from each test set are concatenated into one large test set, on which the AUPR is computed. Different test sets may reflect different spam campaigns, user policy changes for the network, and changing usage patterns among legitimate users over time. This is especially pertinent in the SoundCloud domain, where the data spans all comments over a period of one year. For each test set, we use data from the preceding time period for training[3]. This ensures that each prediction is only made using data from the past, never the future.

SoundCloud is evenly split into ten non-overlapping subsets of roughly 4M messages each (training on 70%, learning PSL weights on 1.25%, and testing on the rest: 1.15M messages). We use users, similar text, and links posted by a given user as the relations. YouTube is split into ten overlapping subsets of roughly 2M messages (75% for training, 2.5% for PSL, and the rest for testing) since this dataset only contains 6.4M messages, these subsets contain some overlap, but the test sets remain mutually exclusive. We use users and similar text as the relations.Twitter is evenly split into ten non-overlapping subsets of roughly 880K messages (70% for training, 6% for PSL, and the rest for testing). We use users,

[1] http://mlg.ucd.ie/yt/

[2] http://www.ntu.edu.sg/home/axsun/datasets.html

[3] For the Twitter dataset, time information was not available, so the tweets were ordered based on tweet ID.

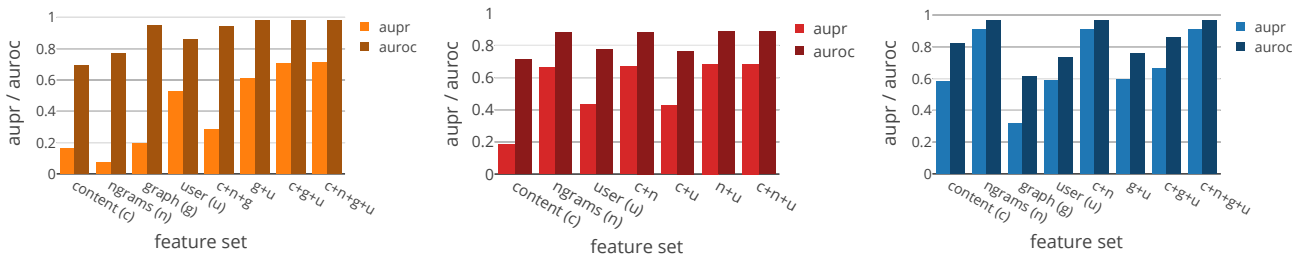

Figure 5: Feature set performance for SoundCloud (left), YouTube (middle), and Twitter (right).

similar text, and hashtags posted by a given user as the relations.

Experiments over multiple development sets indicate that a relatively small percentage of data is needed to learn decent weights for the PSL model. Thus, we reserve about 50K messages per subset for PSL weight learning. We pre-tune ϵ per relation for the MRF factor potentials on separate development data, and use those values for each test set.[4]

5.1 Independent Model Performance We use logistic regression as our independent classifier, since it generally outperformed other methods such as tree ensembles during development. Before testing our relational methods, we perform ablation tests on the independent model to determine which feature sets contribute the most. We test each feature set in isolation and in combination with each other on the first 10% of the data for each domain using 70% for training and tuning, and the remaining 30% for testing (Figure 5). The effectiveness of different feature types varies from domain to domain. For SoundCloud, a combination of content, graph, and user features (c+g+u) yields the best results; n-gram features are not very helpful. For YouTube and Twitter, the opposite is true: n-gram features on their own work almost as well as all feature types put together.

5.2 Relational Modeling Performance We perform four sets of experiments to test our relational models' capability of detecting spam. The 'full' experiments use all available features when learning and making predictions. The 'limited' experiments exclude some of the most informative features: we remove n-gram features from YouTube and Twitter, and graph features from SoundCloud. This makes the prediction problem more challenging, testing the ability of our relational methods to compensate for limited information. In adversarial settings like spam, it's common to only have limited or

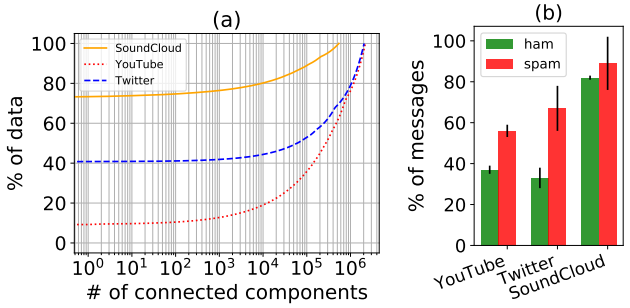

Figure 6: (a) The number of connected components it takes to cover the first five million messages for each domain. (b) Percentage of ham/spam test set messages with at least one connection to any training messages.

noisy information, since spammers quickly adapt their messages and networks to evade detection.

We also evaluate our models for test instances that have no relational connections to any messages in the training sets (Inductive), in addition to evaluating all test instances (Inductive + Transductive). For SoundCloud, a small number of connected components covers a significant portion of ham and spam messages (Figure 6), but this is less so for YouTube and Twitter, where transductive methods would need more and more known labels to propagate to the increasingly disconnected pockets of related messages.

In all settings, we compare against predictions from the independent model (top row). Then we start to incorporate relational structure, applying our joint inference models to the outputs of the independent model as well as to the relationally augmented independent models, which use up to two levels of stacked learning.

We find that not only can relational modeling improve performance for these domains (Table 5)[5], but in many cases stacked learning and joint reasoning work well together, achieving the best performance in many of the experiments. In the case of the limited feature

[4] PSL learning curves as well as PSL and MRF inference times can be found in the appendix A.1

[5] AUROC results can be found in the appendix A.2

Table 5: Relational Modeling Performance Results (AUPR)

		Inductive			Inductive + Transductive		
	Model	SoundCloud	YouTube	Twitter	SoundCloud	YouTube	Twitter
Limited	Independent	0.396	0.148	0.260	0.352	0.387	0.466
	SGL(1)	0.396	0.221	0.281	0.444	0.453	0.494
	SGL(2)	0.342	0.263	0.275	0.364	0.478	0.612
	PSL	0.363	0.225	0.249	0.332	0.438	0.498
	MRF	**0.431**	0.164	0.264	0.547	0.404	0.483
	SGL(1) + PSL	0.284	0.268	0.258	0.521	0.471	0.523
	SGL(1) + MRF	0.412	0.239	**0.285**	0.578	0.473	0.513
	SGL(2) + PSL	0.243	**0.282**	0.270	0.562	0.489	0.594
	SGL(2) + MRF	0.404	0.276	0.280	**0.583**	**0.493**	**0.630**
Full	Independent	0.409	0.321	**0.860**	0.460	0.539	**0.950**
	SGL(1)	0.434	**0.440**	0.848	0.530	0.614	0.946
	SGL(2)	0.369	0.439	0.838	0.483	**0.617**	0.942
	PSL	0.458	0.337	0.823	0.469	0.569	0.927
	MRF	0.487	0.347	0.857	0.579	0.557	0.948
	SGL(1) + PSL	0.289	0.433	0.817	0.602	0.588	0.925
	SGL(1) + MRF	**0.489**	0.439	0.845	**0.604**	0.609	0.944
	SGL(2) + PSL	0.324	0.426	0.813	0.582	0.581	0.923
	SGL(2) + MRF	0.435	0.438	0.835	**0.604**	0.614	0.939

set, relational modeling is able to improve performance for all three domains; in the fully featured case, relational modeling improves performance for the SoundCloud and YouTube domains. For Twitter, the independent model is already very effective when trained with n-grams (AUPR=0.95); stacked learning and joint inference offer no additional benefit. Against an evasive adversary that changes text to avoid detection, the independent model would be less effective and relational modeling more likely to help.

Comparing the different relational methods, we find that no one method dominates; thus, when applying EGGS to a new spam domain, we recommend testing several methods on validation data before committing to a single model. Having more layers in stacking sometimes helps a lot, especially in the Inductive + Transductive setting. For example, on the limited Twitter dataset, SGL goes from 0.494 to 0.612 AUPR with the addition of a second layer, SGL+PSL goes from 0.523 to 0.594, and SGL+MRF goes from 0.513 to 0.630. In other cases, such as the full SoundCloud dataset in the Inductive setting, performance can drop: SGL goes from 0.434 to 0.369, and SGL+MRF goes from 0.489 to 0.435. With more layers, each classifier in SGL is trained on less data, which may explain why performance sometimes decreases.

MRF usually achieves better AUPR than PSL: out of the 12 combinations of datasets and inductive or transductive settings, MRF outperforms PSL on 8, SGL(1)+MRF outperforms SGL(1)+PSL on 10, and SGL(2)+MRF outperforms SGL(2)+PSL on 11.[6] However, PSL usually achieves better AUROC, outperforming the MRF models in 25 out of 36 cases.

6 Discussion

We have shown how to flexibly incorporate relational structure from multiple separate domains using several approaches to build EGGS, a spam detection system that attacks the problem from as many angles as possible. Two of these approaches share a general relational framework that provides full joint inference over related messages, while the pseudo-relational features can be added to any independent model without the need for more complexity. We see that these methods are effective on real-world large-scale datasets in isolation and in combination with one another. Furthermore, if these specific techniques are a poor fit to a new domain, EGGS can easily be adapted to include other features, other types of classifiers, and other relational reasoning methods.

A promising direction of future work involves adversarial manipulations of the training or test data. It would be valuable to know how much relational modeling can increase classification performance *and* robustness for a wide range of adversarial attacks. Code for this application is available at: **https://github.com/snspam/sn_spam**.

[6] This would be statistically significant under a binomial test (29 successes out of 36 trials), but the trials are not independent.

Acknowledgments

This work was supported by ARO grant W911NF-15-1-0265.

References

[1] L. Akoglu, R. Chandy, and C. Faloutsos, *Opinion fraud detection in online reviews by network effects.*, ICWSM, 13 (2013), pp. 2–11.

[2] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani, *Large scale networks fingerprinting and visualization using the k-core decomposition*, in NIPS, 2005, pp. 41–50.

[3] S. Bach, M. Broecheler, et al., *Hinge-loss Markov random fields and probabilistic soft logic*, JMLR, 2017.

[4] F. Benevenuto, G. Magno, et al., *Detecting spammers on Twitter*, *CEAS*, 2010, pp 12.

[5] A. Beutel, W. Xu, et al. *Copycatch: stopping group attacks by spotting lockstep behavior in social networks*, *WWW*, 2013, pp. 119–130.

[6] E. Blanzieri, A. Bryl, *A survey of learning-based techniques of email spam filtering*, AI Review, 2008, pp. 63–92.

[7] P. Boykin, V. Roychowdhury, *Leveraging social networks to fight spam*, Computer, 2005, pp. 61–68.

[8] C. Castillo, D. Donato, et al. 2007. Know your neighbors: Web spam detection using the web topology. In *SIGIR*, 423–430. ACM.

[9] F. Chen, P. Tan, A. Jain, *A co-classification framework for detecting web spam and spammers in social media web sites*, in CIKM, ACM, 2009, pp. 1807–1810.

[10] Z. Chu, I. Widjaja, and H. Wang, *Detecting social spam campaigns on Twitter*, ACNS, 2012, pp. 455–472.

[11] J. Costa, C. Silva, M. Antunes, and B. Ribeiro, *Defining semantic meta-hashtags for Twitter classification*, in ICANNGA, Springer, 2013, pp. 226–235.

[12] Y. Duan, F. Wei, et al., *Graph-based collective classification for tweets*, *CIKM*, 2012, pp. 2323–2326.

[13] S. Fakhraei, J. Foulds, et al., *Collective spammer detection in evolving multi-relational social networks*, in SIGKDD, ACM, 2015, pp. 1769–1778.

[14] A. Fast, and D. Jensen. 2008. Why stacked models perform effective collective classification. *ICDM'08.*, 785–790. IEEE.

[15] G. Fei, A. Mukherjee, B. Liu, M. Hsu, et al., *Exploiting burstiness in reviews for review spammer detection.*, ICWSM, 13 (2013), pp. 175–184.

[16] P. Ferragina, F. Piccinno, and R. Santoro, *On analyzing hashtags in Twitter*, in ICWSM, 2015.

[17] E. Ferrara, O. Varol, et al., *The rise of social bots*, Communications of the ACM, 59 (2016), pp. 96–104.

[18] H. Gao, Y. Chen, et al., *Towards online spam filtering in social networks.*, *NDSS*, 2012, pp. 1–16.

[19] L. Getoor, *Introduction to statistical relational learning*, MIT press, 2007.

[20] S. Ghosh, B. Viswanath, et al., *Understanding and combating link farming in the Twitter social network*, in WWW, ACM, 2012, pp. 61–70.

[21] *B. Hooi, H. A. Song, et al. Fraudar: Bounding graph fraud in the face of camouflage*, *SIGKDD*, 2016, pp. 895–904.

[22] L. Jiang, M. Yu, et al., *Target-dependent Twitter sentiment classification*, *ACL*, 2011, pp. 151–160.

[23] N. Jindal and B. Liu, *Analyzing and detecting review spam*, in ICDM, IEEE, 2007, pp. 547–552.

[24] D. Lowd, and A. Rooshenas. 2015. The Libra toolkit for probabilistic models. *JMLR* 16:2459–2463.

[25] S. KC, and A. Mukherjee, *On the temporal dynamics of opinion spamming: Case studies on Yelp*, *WWW*, 2016, pp. 369–379.

[26] Z. Kou, and W. W. Cohen. *Stacked graphical models for efficient inference in Markov random fields. SDM*, 2007, pp. 533–538.

[27] C. Laorden, B. Sanz, et al., *Collective classification for spam filtering*, CISIS, 2011, pp. 1–8.

[28] H. Li, A. Mukherjee, et al., *Detecting campaign promoters on Twitter using Markov random fields*, *ICDM*, 2014, pp. 290–299.

[29] H. Li, Z. Chen, B. Liu, X. Wei, and J. Shao, *Spotting fake reviews via collective positive-unlabeled learning*, in ICDM, IEEE, 2014, pp. 899–904.

[30] H. Li, G. Fei, S. Wang, B. Liu, et al., *Bimodal distribution and co-bursting in review spam detection*, *WWW*, 2017, p. 1063–1072.

[31] L. Liu, Y. Lu, et al., *Detecting" smart" spammers on social network: A topic model approach*, NAACL-HLT, 2016.

[32] M. Mateen, M. A. Iqbal, M. Aleem, and M. A. Islam, *A hybrid approach for spam detection for Twitter*, *IBCAST*, 2017, pp. 466–471.

[33] Z. Ma, A. Sun, and G. Cong, *On predicting the popularity of newly emerging hashtags in Twitter*, Journal of the Association for Information Science and Technology, 2013, pp. 1399–1410.

[34] M. E. Newman, S. H. Strogatz, and D. J. Watts, *Random graphs with arbitrary degree distributions and their applications*, Physical review E, 2001.

[35] D. O'Callaghan, M. Harrigan, J. Carthy, and P. Cunningham, *Network analysis of recurring Youtube spam campaigns.*, in ICWSM, 2012.

[36] L. Page, S. Brin, R. Motwani, and T. Winograd, *The pagerank citation ranking: bringing order to the web.*, (1999).

[37] S. Pandit, D. H Chau, S. Wang, and C. Faloutsos. *Netprobe: a fast and scalable system for fraud detection in online auction networks.*, *WWW*, 2007, pp. 201–210.

[38] S. Rayana and L. Akoglu, *Collective opinion spam detection: Bridging review networks and metadata*, in SIGKDD, ACM, 2015, pp. 985–994.

[39] M. Richardson, and P. Domingos. 2006. Markov logic networks. *Machine learning* 62(1-2):107–136.

[40] H. Sanchez, and S. Kumar, *Twitter bullying detection. NSDI*, 2011, 12:15–15.

[41] J. Savigny, and A. Purwarianti, *Emotion classification on Youtube comments using word embedding*, *ICAICTA*, 2017, pp. 1–5.

[42] T. Schank and D. Wagner, *Finding, counting and listing all triangles in large graphs, an experimental study*, in WEA, Springer, 2005, pp. 606–609.

[43] S. Sedhai and A. Sun, *Hspam14: A collection of 14 million tweets for hashtag-oriented spam research*, in SIGIR, ACM, 2015, pp. 223–232.

[44] H. Shen, and X. Liu, *Detecting spammers on Twitter based on content and social interaction*, *ICNISC*, 2015, pp. 413–417.

[45] A. P. Singh and G. J. Gordon, *Relational learning via collective matrix factorization*, *SIGKDD*, 2008, pp. 650–658.

[46] L. Song, R. Y. Lau, and C. Yin, *Discriminative topic mining for social spam detection.*, *PACIS*, 2014, pp. 378.

[47] G. Stringhini, C. Kruegel, G. Vigna, *Detecting spammers on social networks*, ACSAC, 2010, pp. 1–9.

[48] E. Tan, L. Guo, et al., *Unik: Unsupervised social network spam detection*, in CIKM, ACM, 2013, pp. 479–488.

[49] B. Viswanath, M. A. Bashir, et al., *Towards detecting anomalous user behavior in online social networks.*, USENIX, 2014, pp. 223–238.

[50] Wang, A. H., *Don't follow me: Spam detection in Twitter*, *SECRYPT*, 2010, pp. 1–10.

[51] D. Wang and C. Pu, *Bean: A behavior analysis approach of URL spam filtering in Twitter*, in 2015 IEEE IRI, Aug 2015, pp. 403–410.

[52] F. Wu, J. Shu, Y. Huang, and Z. Yuan, *Social spammer and spam message co-detection in microblogging with social context regularization*, in CIKM, ACM, 2015, pp. 1601–1610.

[53] H. Xu, W. Sun, and A. Javaid, *Efficient spam detection across online social networks*, in ICBDA, IEEE, 2016, pp. 1–6.

[54] X. Zhang, S. Zhu, and W. Liang, *Detecting spam and promoting campaigns in the Twitter social network*, in ICDM, IEEE, 2012, pp. 1194–1199.

[55] Y. Zhu, X. Wang, et al., *Discovering spammers in social networks*, in AAAI, 2012.

A Appendix

A.1 Learning Curves and Running Times A small amount of data is needed to learn decent weights for our PSL models (Figure 7).

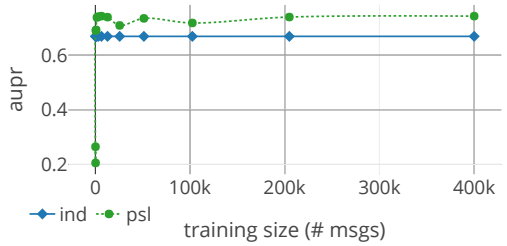

Figure 7: Typical learning curve for the PSL model.

The running times for our joint inference models (Figure 8) show that the PSL model is able to scale more effectively than the MRF model based on the number of edges between relations in the test set.

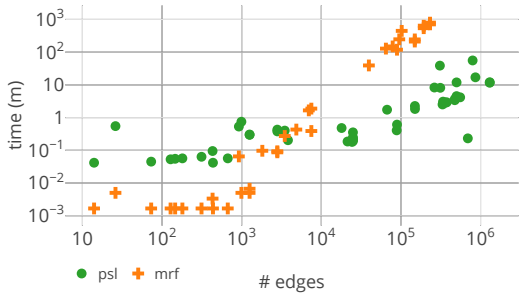

Figure 8: Inference times for MRF and PSL.

A.2 Relational Results (AUROC) We present the AUROC results from our relational models as an additional metric to the main AUPR metric (Table 6).

Table 6: Relational Modeling Performance Results (AUROC)

		Inductive			**Inductive + Transductive**		
	Model	SoundCloud	YouTube	Twitter	SoundCloud	YouTube	Twitter
Limited	Independent	0.814	0.702	0.699	0.905	0.776	0.720
	SGL(1)	0.763	0.748	0.706	0.922	0.812	0.738
	SGL(2)	0.796	0.761	0.707	0.926	0.821	0.806
	PSL	**0.838**	0.745	0.668	0.913	0.807	0.728
	MRF	0.799	0.673	0.700	0.896	0.747	0.726
	SGL(1) + PSL	0.787	0.763	0.684	0.926	0.822	0.745
	SGL(1) + MRF	0.725	0.740	0.707	0.928	0.804	0.744
	SGL(2) + PSL	0.814	**0.767**	0.699	**0.932**	**0.826**	0.789
	SGL(2) + MRF	0.755	0.759	**0.708**	0.931	0.819	**0.808**
Full	Independent	0.929	0.785	**0.957**	0.965	0.843	**0.975**
	SGL(1)	0.929	0.840	0.947	**0.972**	0.880	0.971
	SGL(2)	0.871	0.823	0.943	0.971	0.872	0.969
	PSL	**0.936**	0.803	0.948	0.964	0.857	0.966
	MRF	0.932	0.773	0.956	0.962	0.829	**0.975**
	SGL(1) + PSL	0.934	**0.845**	0.939	**0.972**	**0.884**	0.962
	SGL(1) + MRF	0.933	0.832	0.946	0.971	0.870	0.971
	SGL(2) + PSL	0.882	0.826	0.934	0.971	0.872	0.959
	SGL(2) + MRF	0.870	0.814	0.941	0.969	0.864	0.968