Collective Classification of Social Network Spam

Jonathan Brophy and **Daniel Lowd**
Department of Computer Science
University of Oregon
Eugene, OR 97403
{jbrophy,lowd}@cs.uoregon.edu

Abstract

Unsolicited or unwanted messages is a byproduct of virtually every popular social media website. Spammers have become increasingly proficient at bypassing conventional spam filters, prompting a stronger effort to develop new methods that accurately detect spam while simultaneously acting as a more robust classifier against users that modify their behavior in order to avoid detection. This paper shows the usefulness of a relational model that works in conjunction with an independent model. First, an independent model is built using features that characterize individual comments and users, capturing the cases where spam is obvious. Second, a relational model is built, taking advantage of the interconnected nature of users and their comments. By feeding our initial predictions from the independent model into the relational model, we can start to propagate information about spammers and spam comments to jointly infer the labels of all spam comments at the same time. This allows us to capture the obfuscated spam comments missed by the independent model that are only found by looking at the relational structure of the social network. The results from our experiments demonstrates the viability of our method, and shows that models utilizing the underlying structure of the social network are more effective at detecting spam than ones that do not.

1 Introduction

The appearance of 'spam' in a social network can be any instance of unsolicited or unwanted actions by a user in the network. Traditional methods of classifying spam have resulted in many content-based approaches that characterize spam messages, such as email, social media comments, etc. The increase in scale of many popular social media websites has created greater incentives for spammers to find new ways of bypassing traditional filters. Even if only a small fraction of a spammer's campaign gets through a spam filter, there can be a large number of users that will see those messages, making it worthwhile for the spammer. These spammers can avoid detection by randomizing the content of their messages, or manipulating their content in a way that fools the content-based classifier.

An alternative method could classify spammers based on actions performed between users in a social network (Fakhraei et al. 2015). This is a more robust method of detecting spammers as changing user behavior to bypass spam filters is more difficult than altering spam messages individually. For example, a graph structure could be built around users following other users, where nodes represent users and directed edges represent users following other users. Perhaps legitimate behavior is characterized by having many followers (node with a large in-degree). Thus, a spammer might have difficulty simulating legitimate behavior because they cannot get other users to follow them, making them easier to find among all the other users.

By using aspects of both content and graph-based systems, a spam detection model can capture certain signals that a system focused on only one approach could miss. Combining these methods also gives the added benefit of finding spam messages and spammers. Detecting spam messages themselves may be sufficient for certain domains, but generally it is more useful to characterize the users that are generating the spam to more efficiently prevent similar spam campaigns in the future.

The combination of content and graph-based systems is an improvement, but both of these approaches still consider the data to be independent and identically distributed (i.i.d.). They look at incoming spam messages one at a time, and based on characteristics about the spam message itself, as well as the user's previous behavior, make a decision about whether it should be labeled as spam or not spam. Social network spam is not i.i.d. This provides the opportunity to exploit the interconnected nature of spammers, spam comments, and any other domain-specific entities that tie spammers to their spam.

Our approach is motivated by *SoundCloud.com*, a social network for sharing music. This domain is an excellent place to test out a statistical relational model because it can exploit the underlying structure between users, messages, and the uploaded tracks where messages are posted. Our approach uses a type of probabilistic graphical model called hinge-loss Markov random fields (HL-MRFs) (Bach et al. 2013) to jointly infer labels of spam messages.

The following sections describe the data in more detail, our independent model utilizing the combination of content and graph-based approaches to classifying spam comments, our relational approach improving upon this independent model, and an experimental validation on a real world

Entity	Count
non-spam comments	42,099,424
spam comments	684,338
tracks	7,796,019
follower actions	335,197,112
non-spammers	5,377,679
spammers	128,016

Table 1: Dataset Statistics.

SoundCloud dataset.

2 Data

The dataset comes from *SoundCloud.com* and includes an entire year's worth of comments from October 10, 2012 to September 30, 2013. The data came containing information about comments, tracks, follower actions, spam warnings, and spam reports.

For each comment, we have the anonymized user id of the user that posted the comment, the anonymized track id the comment was posted on, a timestamp of when the comment was posted, the actual message of the comment, and a label indicating whether the comment was marked as spam or not.

For each track, we have the anonymized track id, the anonymized user id of who uploaded the track, the duration of the track, and a timestamp of the track's last update.

SoundCloud allows each user the ability to follow other users. Thus, for each follower action, we have the anonymized user id of user y being followed, the anonymized user id of user x doing the following, and a timestamp indicating when user x started following user y.

For each spam warning, we have the anonymized user id of the user receiving the warning, the warning level, the reason for the warning, a timestamp of when the warning was issued, and a binary suspended label for each warning.

For each spam report, we have the anonymized user id of the user that posted the comment, a timestamp of when the comment was published, and a binary suspended label for each report.

The basic statistics of this dataset reveal just how imbalanced the class labels are distributed throughout the comments (Table 1).

2.1 Pre-Preprocessing

All pre-processing done to this dataset affects only the comments. Commas were stripped from the body of each comment in order to easily store the comments as a flat CSV file. Also, all comments were sorted in increasing order based on their respective timestamps. Finally each comment was given a unique identifier. Tracks, follower actions, spam reports, and spam warnings were also written to CSV files to make them easier to load into various data manipulation frameworks.

3 Methodology

The first task is to build an independent model that combines the best of both worlds from content and graph-based systems in order to create a solid baseline and starting point for our relational model. One list of engineered features focuses on the content of the messages and constitutes our content-based feature set. Another list of engineered features extracts user behavior which makes up the graph-based feature set. A third and final feature set includes engineered features based on relational aspects within the data and serves as candidate features to be explored further in the relational model.

The second task involves the creation of an HL-MRF (Bach et al. 2013) instantiated using *probabilistic soft logic*[1] (PSL) where efficient inference can be applied. Three relational models are explored in this domain, where each one builds on the previous model and attempts to capture a different relational signal among the users, spam messages, and tracks.

3.1 Independent Model

By looking at just the text of each comment, the following content-based features can be extracted: the number of characters, *NumChars*, and a boolean indicating whether or not the comment contains a link, *HasLink*. Bag of words and k-grams could also be explored, but this causes the feature space to increase very quickly and necessitates constant updating of these features. These two features make up the **Content** feature set (Table 2).

The next set of features look at user behavior based on the connectivity of users to other users using the list of follower actions. We can represent this list of affiliations as a graph by constructing a node for every user in the list, and then add a directed edge from user x to user y whenever user x starts to follow user y. The resulting directed graph contains 335,197,112 edges. We run the following following graph algorithms on it:

- **Pagerank** (Page et al. 1999), which gives us a sense of which nodes are more important than others by looking at which nodes receive the most links.
- **Triangle count** (Schank and Wagner 2005), which looks at how many triangles a node is a part of, indicating the connectivity of that node.
- **K-core** (Alvarez-Hamelin et al. 2005), an iterative algorithm that trims the least connected nodes on every iteration, and assigns the trimmed node an id number corresponding to the iteration when it was pruned.
- **In-degree** (Newman, Strogatz, and Watts 2001), the number of edges that enter a particular node.
- **Out-degree** (Newman, Strogatz, and Watts 2001), the number of edges that leave a particular node.

Each one of these algorithms represent a different way of finding essentially the same thing, how connected a user is to other user users in the network. This information is useful because it can give us insight into some of the behavior that characterizes spammers and non-spammers. This is the same approach as in (Fakhraei et al. 2015), except they explore a number of different actions between users, whereas this data set only gives one relation between users, the list of follower

[1] http://psl.umiacs.umd.edu/

actions. All features in this set are computed using *Graphlab Create*[2] and can be summed up in the **Graph-based** feature set (Table 2).

Independent model features	
Content	
NumChars	Number of chars in msg.
HasLink	True if msg has a URL, else False.
Graph-based	
Pagerank	Pagerank of each node in the follower graph G.
TriCnt	Number of triangles of each node in G.
KCore	Iteration each node in G was pruned.
InDegree	Number of edges entering a node in G.
OutDegree	Number of edges leaving a node in G.
Relational	
UUps	UserUploads, number of uploaded tracks per user.
UComs	UserComments, number of msgs posted by each user.
ULComs	UserLinkComments, number of msgs with a URL by each user.
ULRatio	UserLinkCommentsRatio, fraction of user msgs containing a URL.
TrackSpam	Number of spam msgs per track.
Trackham	Number of non-spam msgs per track.
CMSRatio	CommentMatchSpamRatio, fraction of spammy matching msgs.

Table 2: Features used in the independent model grouped into three different feature sets.

The final set of features in the independent model focuses on the connections between users, comments, and tracks being posted. *UserUploads* is the number of tracks uploaded by each user. *UserComments* is the number of comments posted by each user. *UserLinkComments* is the number of comments posted by each user that contain a link in their message. *UserLinkCommentsRatio* is the fraction of comments previously posted by each user that contain a link in the message over the total number of posted messages by that user.

TrackSpam and *TrackHam* are the number of spam and non-spam comments on each track, respectively. Finally, *CommentMatchSpamRatio* is, given a comment, the fraction of other matching comments that are marked as spam over the total number of other matching comments. For example, if a comment with the message 'hey' matches 3 other comments because they have the message 'hey', and 2 of them were marked as spam, then the fraction of matching spam messages would be $\frac{2}{3}$.

All features in this final set are computed in sequential order of comments based on their timestamp. For example, when computing *UserUploads* for comment #100 posted by user x, we record the number of tracks uploaded by user x up until comment #100. This creates a more realistic scenario as features are computed only based on previous comments. The only features not computed in sequential fashion are the graph-based features and *UserTracks*. This is due to the fact that tracks and affiliations are large separate entities from the comments, requiring computationally long queries for every comment. This final feature set is summarized in the **Relational** feature set (Table 2).

These relational features are important because they represent candidate features for our relational model. Each one of these features in some way connects users to comments, users to tracks, tracks to comments, and comments to comments. This independent model uses random forest as its classifier and information gain to rank relative feature importance. If these features have high relative importance, then there is a good chance their relative nature can be exploited further in a dedicated relational model.

By calculating features in the **Relational** set sequentially based on the labels given, the independent model simulates some of the behavior occurring in the relational model. This improves performance, but does not allow the model to reason about multiple relationships simultaneously. Jointly inferring labels for all messages based on those connections can capture complex signals missed by this independent model.

The next section discusses which relational features are worth exploring in more depth and how they can be exploited in the relational model.

3.2 Relational Model

The growing field of statistical relational learning (SRL) has developed various methods for doing collective classification (Getoor 2007). In this work, we utilize PSL, software developed by researchers in the LINQS SRL group at the University of Maryland (Bach et al. 2015). PSL allows us to construct weighted logical rules (formulas) such as:

$$\omega : Friends(x, y) \wedge Smokes(x) \rightarrow Smokes(y) \qquad (1)$$

Where ω is the weight or relative importance of the rule, Friends() and Smokes() are the predicates (atoms), and *x* and *y* are logical variables. This rule reads 'if *x* is friends with *y*, and *x* smokes, then *y* is more likely to smoke'. Everything to the left of the implication is the body of the formula, and everything to the right is the head. A ground atom is a predicate whose variables have all been replaced by constants. A formula has been grounded out when all predicates in a formula are ground atoms.

Hinge-loss Markov random fields (HL-MRFs) are log linear models that use hinge loss functions of the variable states as features and can be modeled as a conditional probability distribution as follows (Fakhraei et al. 2015):

$$P(Y|X) = \frac{1}{Z(\omega)} \exp\Big(- \sum_{i=1}^{n} \omega_i \phi_i(X, Y) \Big) \qquad (2)$$

where ϕ_i is the set of n continuous potential:

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

[2]Now known as Turi: https://turi.com/products/create/

and ℓ is a linear function of X and Y, where $p_j \in \{1, 2\}$.

HL-MRFs can be instantiated from PSL where each node represents a ground atom, and each feature represents the grounding of one of the formulas. It is apparent that most real-world data is complex and contains uncertainty. PSL provides us with a nice representation to model rules using logic, which handles complexity, and instantiate them as a probabilistic graphical model (HL-MRF), which handles uncertainty.

The semantics of PSL make it easy to create an HL-MRF with rules of the form as in (1) above. We combine multiple rules to create three relational models of increasing complexity. The first model takes the predictions from the independent model and separates each prediction into varying levels of confidence. For example, if the independent model classifies a comment as spam with a probability of 0.98, then our relational model can use this information to more confidently label this comment as spam (Figure 1).

$$\neg spam(c) \quad \text{(a)}$$
$$\neg spammer(u) \quad \text{(b)}$$
$$superSpam(c) \rightarrow spam(c) \quad \text{(c)}$$
$$semiSpam(c) \rightarrow spam(c) \quad \text{(d)}$$
$$superHam(c) \rightarrow \neg spam(c) \quad \text{(e)}$$
$$semiHam(c) \rightarrow \neg spam(c) \quad \text{(f)}$$
$$posts(u, c) \wedge spam(c) \rightarrow spammer(u) \quad \text{(g)}$$
$$posts(u, c) \wedge \neg spam(c) \rightarrow \neg spammer(u) \quad \text{(h)}$$
$$posts(u, c) \wedge spammer(u) \rightarrow spam(c) \quad \text{(i)}$$
$$posts(u, c) \wedge \neg spammer(u) \rightarrow \neg spam(c) \quad \text{(j)}$$

Figure 1: Relational model using predictions from an independent model as priors as well as user behavior to classify spam; c is short for comment, and u is short for user.

Rules (a) and (b) denote initial priors that most comments are not spam and that most users are not spammers, which is a fair assumption due to the imbalanced nature of the data. Rules (c)-(f) provide additional prior evidence for each comment from the independent model.

Rules (g)-(j) display the first example of how these rules can work together to propagate label information. Take rule (g) for example, if a user posts a comment, and that comment is spammy, then that user is more likely to be a spammer. Thus, if we have any information about that comment, whether it is more or less spammy, then we can start to get a sense of whether or not the user is a spammer or non-spammer. In this case, we do have some prior information about these comments due to the first six rules of our model.

Now that we have some information about the user, we can use that information to help label comments posted by that user. Take rule (i) for example, which says that if a user posts a comment, and that user is a spammer, then that comment is more likely to be spam. After gaining initial evidence that this user posted a spammy comment, it is now more likely that other comments posted by this user will also be spammy. PSL works to simultaneously satisfy all rules in the model, and it is for this reason as well as the interconnected structure of the network that we can reason about multiple users and comments at the same time.

After some preliminary experiments, the independent model identified a relational feature with high relative importance, *CommentMatchSpamRatio*. Thus, the second model builds on the first one by exploring the relational nature of this feature. This feature computes the fraction of matching comments that are marked as spam over the total number of matching comments. The intuition is that spammers tend to post comments from one or multiple accounts that are very similar or exactly the same. Using PSL, we can write rules similar to the bottom four in Figure 1 to capture this behavior (Figure 2).

$$model\ 1\ rules$$
$$+$$
$$inHub(h, c) \wedge spam(c) \rightarrow spamHub(h) \quad \text{(k)}$$
$$inHub(h, c) \wedge \neg spam(c) \rightarrow \neg spamHub(h) \quad \text{(l)}$$
$$inHub(h, c) \wedge spamHub(h) \rightarrow spam(c) \quad \text{(m)}$$
$$inHub(h, c) \wedge \neg spamHub(h) \rightarrow \neg spam(c) \quad \text{(n)}$$

Figure 2: Second relational model that adds 4 more rules about matching comments belonging to spam hubs or non-spam hubs; c is short for comment, and h is short for hub.

Four rules are added in this second relational model, and it follows the same structure as the last four rules in the first model. Rules (k) and (m) are used to propagate information in both directions, while rules (l) and (n) act as complements to these rules to gather information about non-spam comments and non-spam hubs. These rules could have been written in such a way that treats comments individually. For example, rule (k) could have been written in the form:

$$matches(c_1, c_2) \wedge spam(c_1) \rightarrow spam(c_2) \quad (3)$$

Rules (l)-(n) could be re-written in a similar fashion, and this would accomplish essentially the thing as rules (k)-(n) in Figure 2. We chose to group comments into hubs to increase efficiency and reduce the number edges connecting ground atoms. Suppose we were using (3) above and we had 100 comments, the resulting graphical model would need at least 10,000 edges to take care of every possible combination of comments. By grouping comments into hubs, where a hub represents all comments that match each other, we cut down on the number of possible configurations that can arise.

Now we can propagate information similar to model 1, if a comment is part of a hub, and that comment is spammy, then the hub itself becomes more spammy. If we see more evidence of spam comments belonging to this hub, then the relational model becomes more confident of labeling that

hub as a spamHub. Now if a comment is encountered and the model does not have strong evidence of whether it is spam or non-spam, but it sees that it is part of a spamHub, then it can be more confident that the comment should be labeled as spam.

This model starts out with no prior knowledge of whether hubs are spammy or non-spammy; they all start as neutral. As information begins to propagate between comments, users, and hubs, these hubs slowly start to turn more spammy or non-spammy. Ultimately, this helps classify comments that are hard to judge at first glance, but become more obvious by studying their connections to other users and comments.

The third model introduces four more rules added on to the second model, and these new rules bridge the relational gap between tracks and comments (Figure 3).

$$
\begin{array}{rl}
 & model\ 2\ rules \\
 & + \\
inTrack(t,c) \wedge spam(c) \rightarrow sTrack(t) & \text{(o)} \\
inTrack(t,c) \wedge \neg spam(c) \rightarrow \neg sTrack(t) & \text{(p)} \\
inTrack(t,c) \wedge sTrack(t) \rightarrow spam(c) & \text{(q)} \\
inTrack(t,c) \wedge \neg sTrack(t) \rightarrow \neg spam(c) & \text{(r)}
\end{array}
$$

Figure 3: Third relational model that adds 4 more rules identifying comments belonging to spammy tracks or non-spammy tracks; c is short for comment, t is short for track, and sTrack is short for spamTrack.

The rules added in this model correspond to the relational features *TrackSpam* and *TrackHam* present in the independent model. Those features represent the number of spam comments and non-spam comments for each track, respectively. The notion of tracks in this model is the same idea as hubs in the second model. Tracks start out as neutral, and as they accumulate evidence, become places more likely for spam or ham to be posted. PSL makes it easy to express these complex relationships with simple rules and syntax.

This third model is now much more powerful as it captures signals from various relationships within the data. All the rules work in conjunction as the HL-MRF attempts to satisfy them simultaneously. It is important to note though, that adding rules may improve the performance of the model, but that comes at the cost of more complexity and possible intractability, as too many rules makes inference over the graphical model too difficult.

A fourth model was explored that added four more rules creating hubs based on matching URLs. If a comment contained any URLs, then the first URL was extracted and put into its respective hub. The idea is that there are significant differences between the URLs that spammers post as opposed to the URLs that non-spammers post. After some preliminary experiments though, the addition of these rules did not improve the model very much, but this could be due to a lack of URLs in the comments being tested and may be worth exploring again on a data set with numerous URL postings.

It is preferable to learn the weights of these rules than to guess their relative importance. PSL allows the relational model to learn the weights of the rules using data. After initializing all the weights, they can then be learned by computing the gradient of the log-likelihood of (2) with respect to an individual weight ω_i, and applying expectation maximization to find the most probable explanation given the current set of weights (Kimmig et al. 2012).

Learning the weights to the relational model takes up the majority of the running time as it essentially performs inference many times as it computes the optimal weights given the data. Once the weights are learned, they can then be used to do joint inference on a different subset of the data, or on a separate domain entirely, given that the rules of the relational model make sense in both domains.

4 Experiments

We performed two sets of experiments to evaluate the performance of the relational model. Each experiment consists of the following:

- A subset of 2 million consecutive comments is chosen from the SoundCloud data set. Those 2 million comments are split in half to create two smaller data sets, D1 and D2. D1 is used for weight learning while D2 is used for testing and evaluation.
- The independent model trains on the first 70% of the of the data in D1, and then generates predictions on the remaining 30% of D1. The same is then done for D2.
- The relational model then learns weights for its rules using the generated predictions from D1, and finally does joint inference on the remaining 30% of comments in D2.

The area under the precision-recall curve (AUPR) is the main metric used in these experiments to evaluate how each model is performing. Predictions on the test set from the independent and relational models are recorded and run through *Sci Kit's*[3] metrics framework.

Before the AUPR is computed, a small amount of noise (a random number between 0 and 2.5×10^{-3}) is added to each prediction in both models. This maintains the underlying label the models have assigned to each comment, but prevents ties in the predictions to avoid optimistic estimates in the precision-recall curve. If the model contains numerous ties in predictions, as the level of recall varies, the precision does not necessarily change linearly, and thus linear interpolation between points is not an optimal strategy for computing the area under the curve (Davis and Goadrich 2006). Thus, adding a small random amount of noise can help combat this problem while calculating the AUPR.

The area under the Receiver Operating Characteristic (ROC), AUROC, is also recorded for each model, but due to the heavy skew present in the data, should not be viewed as the main factor when evaluating model performance. This is due to the fact that the ROC curve takes the true negatives into account when calculating the false positive rate (FPR), and thus a large change in false positives will greatly affect

[3] http://scikit-learn.org/stable/

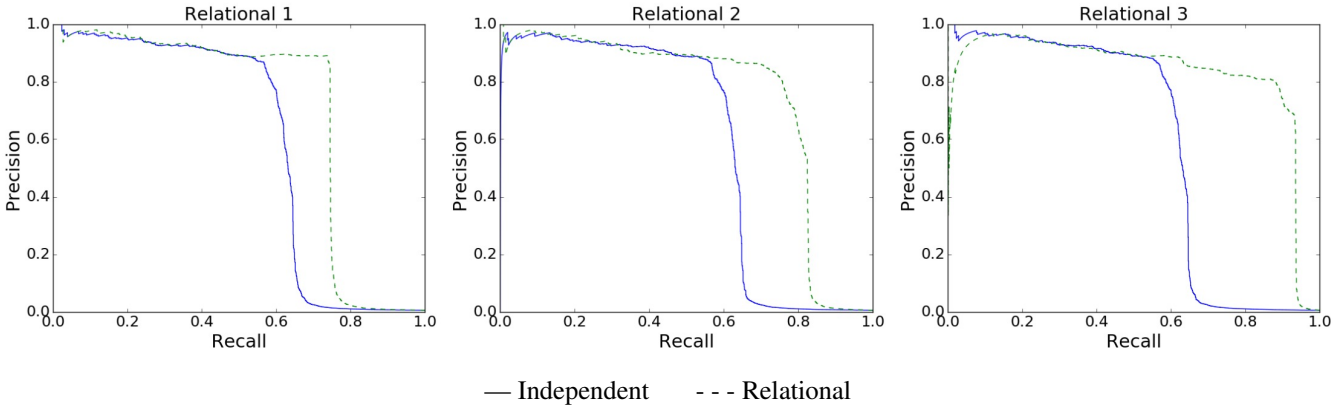

Figure 4: Three relational models of increasing complexity tested on the same data set (comments #31-33 million). Model 1 with priors and spammer information (left). Model 2 with rules added to capture matching comments (middle). Model 3 that adds rules about how spammy certain tracks are (right).

precision and the precision-recall curve, while inducing a small change on the FPR and the resulting ROC curve (Davis and Goadrich 2006), revealing overly optimistic model performance.

4.1 Varying Relational Classifiers

This section presents the results of three different models presented in Section 3.2. In each experiment, models are trained and tested on a 2 million comment subset of the data from comments 31 million to 33 million.

The addition of more rules increases the performance for each relational model tested on this set of comments (Figure 4). We already see an improvement in the first relational model by propagating information back and forth between users and comments, using the predictions from the independent model as priors to give us a starting point of information to work with. The second model improves upon the first by working to label the matching comments. If more comments had matches to other comments in this data set, then this model would most likely provide even bigger improvements. Finally, adding rules about spammy tracks makes a vast improvement over the second model (Table 3). This indicates that certain tracks tend to act as hubs for spam comments. Preliminary work found that the majority of spam comments are concentrated in a small fraction of the total number of tracks, and tended to be on tracks that contained many other comments. This makes sense, as spam posted on popular tracks is more likely to be seen by many users.

The third model, shown on the right in Figure 4, makes the biggest improvement on the right side of the curve, helping increase the recall while maintaining high precision. When the threshold is high, recall tends to be lower and precision tends to be higher since only the predictions that the model is very confident about get labeled correctly. This is what we see in the top left of the graph, where the independent model captures the easy comments with high confidence, and this is transferred over to the relational model so it can be confident about these comments as well. The relational model really shines in instances where comments are not labeled confidently one way or the other, so taking advantage of the comment's connections allow these comments to be labeled more accurately.

Model	AUPR	AUROC
Independent	0.586	0.819
Relational 1	0.695	0.869
Relational 2	0.739	0.908
Relational 3	0.825	0.962

Table 3: Comparison of different relational models to the baseline independent model on comments 31 million to 33 million.

4.2 Varying Comment Subsets

In this section, we choose three different subsets of 2 million consecutive comments each, spaced out among the full data set. Some statistics about each subset of comments are recorded (Table 4), such as the number of hubs containing more than one matching comment. These counts help visualize the characteristics of the test set and possibly provide some idea of how useful certain rules will be. For example, a data set with just as many tracks as comments means that each comment is posted on a different track, reducing the viability of the rules added in the third relational model.

Entity	6M-8M	31M-33M	38M-40M
Spam	1,882	1,688	2,427
Users	118,725	134,759	139,292
Hubs	9,927	11,012	11,884
Tracks	135,414	143,545	139,421

Table 4: Count of different entities contained in the test set of each comment group (last 300k comments).

Relative improvements of the relational model over the independent model can be seen in each data set (Figure

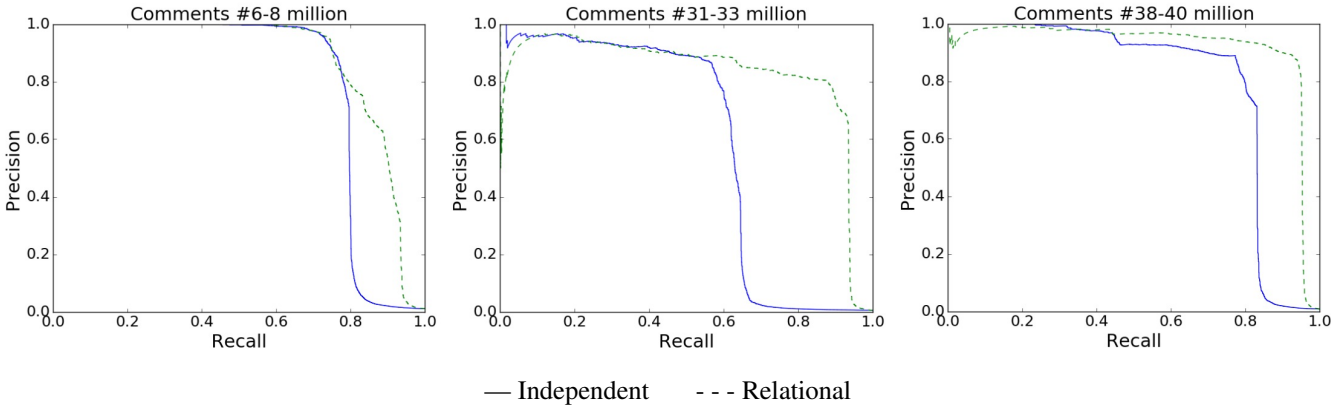

Figure 5: Relational model 3 tested on three different subsets of comments in the data. Comments used: 6 million to 8 million, tested on the last 300k comments (left). Comments used: 31 million to 33 million, tested on the last 300k comments (Middle). Comments used: 38 million to 40 million, tested on the last 300k comments (right).

Data	AUPR (Ind)	AUPR (Rel)
6M-8M	0.792	0.873
31M-33M	0.588	0.825
38M-40M	0.792	0.915

Table 5: Comparison of relational model 3 on different comment subsets in the data.

5). Some of the precision-recall curves for the independent model are already pretty good, leaving less room for our relational model to improve upon, but the difference is still noticeable. The AUPR and AUROC scores for both models in each comment subset is also recorded (Table 5).

The increase in performance of the relational model to the independent model is evident for each comment subset, but the improvements seem to get better for the comment subsets that occur later in time. This could be due to the fact that the independent model's engineered features have not seen as many examples for the earlier comments. This can cause less accurate predictions that get fed into the relational model which might propagate information less accurately.

One additional experiment used the previous 700K comments before the test set as extra evidence for the relational model. Thus, more hubs, users, and tracks were inferred as spammy or non-spammy, and this could help identify spam in the test set. For example, if a user shows up many times posting spam in the evidence set, and only posts one comment in the test set, it is easy to infer that the comment in the test set is probably also spam. Without these extra observations, we have no previous evidence of what kind of user they are, making it harder to classify their single comment in the test set.

After running relational model 3 with extra evidence on the 31 million to 33 million comment subset, the results did not make a big improvement over the model without extra observations. This could be due to a lack of overlap between users, matching comments, and tracks between the evidence and test sets, but this lack of improvement is also encouraging. Since we already know that large increases in performance can be made from doing joint inference over the test set, then we know that we can get good results from a smaller graphical model that uses less computation. So it is possible to reason that this evidence set can be replaced by a larger test set where joint inference can label even more comments at the same time with more accuracy than with a smaller test set. This implies the ability to scale the relational model to label more instances collectively.

5 Related Work

Spam is a long-standing problem for many domains, and identifying user behavior is a reasonable step towards capturing more sophisticated spammers. Liu et al. used topic modeling to detect spammers on *Weibo*, a popular social network in China (Liu et al. 2016). Wang and Pu extracted behavioral characteristics from users to help identify URL spam in *Twitter* data (Wang and Pu 2015).

Collective classification is a promising method of dealing with complex spam campaigns. Laorden et al. studied the prominence of email spam and created a text classification model using collective filtering in a semi-supervised setting (Laorden et al. 2011).

Online social networks offer up a network structure well suited for collective classification. Lee et al. devised a model to accurately detect emerging trending topics in *Twitter* to focus a model's attention in places where spam is most likely to occur (Lee et al. 2012). Zhu et al. studied spam content in one of China's most popular social network services, *renren.com*, in which they looked at the social activities between users and came up with a compact matrix factorization of the problem with social regularization to identify spammers (Zhu et al. 2012). Fakhraei et al. utilize user reports and PSL to collectively classify spammers in a social dating network (Fakhraei et al. 2015).

Rayana and Akoglu built a unified framework where features are extracted based on language and behavior models in addition to relational data from the network. In this case,

they used these features to detect fake reviews on various datasets on *Yelp.com* (Rayana and Akoglu 2015).

6 Discussion

We have shown the benefits of using a model that can leverage the underlying connections present between the data in *SoundCloud*. With the aid of an independent model to mark clear instances of spam and provide a starting point of information, our relational model can work to identify the perhaps intentionally obfuscated comments missed by the independent classifier.

We have seen that adding more rules to a relational classifier can increase its performance, but adding too many can cause a bottleneck in computation time. It is not hard to see the benefits of using a relational model, where implementations like PSL make it very easy to express simple rules that can capture complex relationships throughout the data.

One final aspect to note about this approach is the relationship between the independent and relational models. Since the predictions from the independent model are fed into the relational model, any performance improvement in the independent model will most likely translate to improved performance for the relational model as well. Thus, more in depth natural language processing (NLP) features could be engineered for the independent model, but these are not explicitly necessary to show the relative improvements of the relational model.

The next step of this work would involve testing this model on a different, but similar domain to see if these results can be replicated. *YouTube.com* would make an excellent choice, as its popularity certainly attracts many spammers, and its social network structure is similar to that of *SoundCloud's*. Tracks could be replaced by videos, since users post comments to other user's videos the same way users post comments to other user's tracks. All the other rules can essentially stay the same.

There is also the opportunity to learn weights in one domain, and then test their effectiveness on another domain. Also, more work needs to be done on characterizing the practical size of data instances that can be jointly labeled at one time, and how this characterization changes as the number of rules increase or decrease.

One segment of the data that was not used involved spam warnings and spam reports. The ability of one user to flag other users is a common feature in most social networks, and this information can lead to clues about who the spammers are, as well as the credibility of users doing the flagging, as in (Fakhraei et al. 2015).

The applications for this kind of model are not bound to social networks. Any type of data that houses underlying relations can benefit from this methodology, and it is exciting to see what other domains relational machine learning will impact.

References

Alvarez-Hamelin, J. I.; Dall'Asta, L.; Barrat, A.; and Vespignani, A. 2005. Large scale networks fingerprinting and visualization using the k-core decomposition. In *Advances in neural information processing systems*, 41–50.

Bach, S. H.; Huang, B.; London, B.; and Getoor, L. 2013. Hinge-loss Markov random fields: Convex inference for structured prediction. In *Uncertainty in Artificial Intelligence (UAI)*.

Bach, S. H.; Broecheler, M.; Huang, B.; and Getoor, L. 2015. Hinge-loss markov random fields and probabilistic soft logic. *arXiv preprint arXiv:1505.04406*.

Davis, J., and Goadrich, M. 2006. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, 233–240. New York, NY, USA: ACM.

Fakhraei, S.; Foulds, J.; Shashanka, M.; and Getoor, L. 2015. Collective spammer detection in evolving multi-relational social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1769–1778. ACM.

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

Kimmig, A.; Bach, S. H.; Broecheler, M.; Huang, B.; and Getoor, L. 2012. A short introduction to probabilistic soft logic. In *NIPS Workshop on Probabilistic Programming: Foundations and Applications*.

Laorden, C.; Sanz, B.; Santos, I.; Galán-García, P.; and Bringas, P. G. 2011. *Collective Classification for Spam Filtering*. Berlin, Heidelberg: Springer Berlin Heidelberg. 1–8.

Lee, K.; Caverlee, J.; Kamath, K. Y.; and Cheng, Z. 2012. Detecting collective attention spam. In *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality*, 48–55. ACM.

Liu, L.; Lu, Y.; Luo, Y.; Zhang, R.; Itti, L.; and Lu, J. 2016. Detecting" smart" spammers on social network: A topic model approach. *arXiv preprint arXiv:1604.08504*.

Newman, M. E.; Strogatz, S. H.; and Watts, D. J. 2001. Random graphs with arbitrary degree distributions and their applications. *Physical review E* 64(2):026118.

Page, L.; Brin, S.; Motwani, R.; and Winograd, T. 1999. The pagerank citation ranking: bringing order to the web.

Rayana, S., and Akoglu, L. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 985–994. ACM.

Schank, T., and Wagner, D. 2005. Finding, counting and listing all triangles in large graphs, an experimental study. In *International Workshop on Experimental and Efficient Algorithms*, 606–609. Springer.

Wang, D., and Pu, C. 2015. Bean: A behavior analysis approach of url spam filtering in twitter. In *2015 IEEE International Conference on Information Reuse and Integration*, 403–410.

Zhu, Y.; Wang, X.; Zhong, E.; Liu, N. N.; Li, H.; and Yang, Q. 2012. Discovering spammers in social networks. In *AAAI*.