GANs for Semi-Supervised Opinion Spam Detection

Gray Stanton[1] and **Athirai A. Irissappane** [2]
[1]Department of Statistics, Colorado State University
[2]School of Engineering and Technology, University of Washington, Tacoma
gray.stanton@colostate.edu, athirai@uw.edu

Abstract

Online reviews have become a vital source of information in purchasing a service (product). Opinion spammers manipulate reviews, affecting the overall perception of the service. A key challenge in detecting opinion spam is obtaining ground truth. Though there exists a large set of reviews, only a few of them have been labeled spam or non-spam. We propose spamGAN, a generative adversarial network which relies on limited labeled data as well as unlabeled data for opinion spam detection. spamGAN improves the state-of-the-art GAN based techniques for text classification. Experiments on TripAdvisor data show that spamGAN outperforms existing techniques when labeled data is limited. spamGAN can also generate reviews with reasonable perplexity.

1 Introduction

Opinion spam is a widespread problem in e-commerce, social media, travel sites, movie review sites, etc. [Jindal *et al.*, 2010]. Statistics show that more than 90% of consumers read reviews before making a purchase [1]. The likelihood of purchase is also reported to increase with more reviews. Opinion spammers exploit such financial gains by providing spam reviews which influence readers and thereby affect sales. We consider the problem of identifying spam reviews as a classification problem, i.e., a review can be classified as spam or non-spam.

One of the main challenges in identifying spam reviews is the lack of labeled data, i.e., spam and non-spam labels [Rayana and Akoglu, 2015]. While there exists a corpus of online reviews, only few of them are labeled. This is mainly because manual labeling is often time consuming, costly and subjective [Li *et al.*, 2018]. Research shows that unlabeled data, when used in conjunction with small amounts of labeled data can produce considerable improvement in learning accuracy [Ott *et al.*, 2011]. There is very limited research on using semi-supervised learning techniques for opinion spam detection [Crawford *et al.*, 2015]. The existing semi-supervised learning approaches [Li *et al.*, 2011; Hernández *et al.*, 2013; Li *et al.*, 2014] for identifying opinion spam use pre-defined set of features for training their classifier. In this paper, we will use deep neural networks which will automatically discover features needed for spam classification [LeCun *et al.*, 2015].

Deep generative models have shown promising results for semi-supervised learning [Kumar *et al.*, 2017]. Specifically, Generative Adversarial Networks (GANs) which have the ability to generate samples very close to real data have achieved state-of-the art results. However, most research on GANs are for images (continuous values) and not text data (discrete values) [Fedus *et al.*, 2018]. GANs operate by training two neural networks which play a min-max game: discriminator tries to discriminate real training samples from fake ones and generator tries to generate fake training samples to fool the discriminator. The drawbacks with GANs are: 1) when data is discrete, the gradient from the discriminator may not be useful for improving the generator, because the slight change in weights brought forth by the gradients may not correspond to a suitable discrete mapping in the dictionary [Huszár, 2015]; 2) the discrimination is based on the entire sentence not parts of it, leading to the sparse rewards problem [Yu *et al.*, 2017].

Very few GAN-based methods (SeqGAN [Yu *et al.*, 2017], StepGAN [Tuan and Lee, 2018], MaskGAN [Fedus *et al.*, 2018]) exists for text generation (not classification). However, they are limited by the length of the sentence that can be generated, e.g., MaskGAN considers 40 words per sentence. These approches are unsuitable for most online reviews which are relatively lengthy, e.g., the TripAdvisor review dataset used in our experiments has sentences with median length 132. The only existing GAN-based approach for text classification, CS-GAN [Li *et al.*, 2018] is not optimal for spam detection due to the length of reviews, subtlety of classification, lack of labeled data (CS-GAN is supervised) and computation time.

In this paper, we propose spamGAN, a semi-supervised GAN based approach for classifying opinion spam. spamGAN uses both labeled instances and unlabeled data to correctly learn the input distribution, resulting in better prediction accuracy for comparatively longer reviews. spamGAN consists of 3 different components: generator, discriminator, classifier which work together to not only classify spam reviews but also generate samples close to the train set. We conduct experiments on TripAdvisor dataset and show that spamGAN outperforms existing works when using limited labeled data.

Following are the main contributions of this paper: 1) To the best of our knowledge, we are the first to explore the potential of GANs for spam detection; 2) spamGAN improves the state-

[1] https://learn.g2crowd.com/customer-reviews-statistics.

of-the-art GAN based models for text classification as it leverages both labeled, unlabeled data in a semi-supervised manner (see Sec. 2 for details); 3) most existing research (non-deep learning methods) on opinion spam manually identify heuristics/features for classifying spamming behavior, however, in our GAN based approach, the features are learned by the neural network; 4) experiments show that spamGAN outperforms state-of-the art methods in classifying spam when limited labeled data is used; 5) spamGAN can generate spam/non-spam reviews very similar to the training set which can be used for synthetic data generation in cases with limited ground truth.

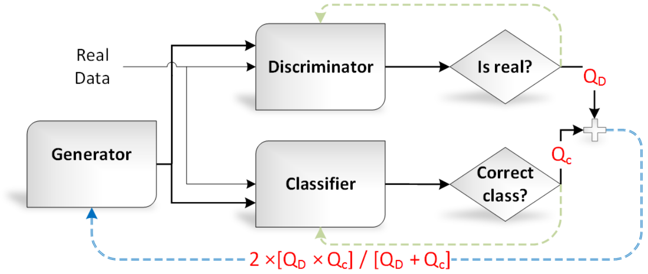

Figure 1: spamGAN Architecture

2 Related Work

Existing opinion spam detection techniques are mostly supervised methods based on pre-defined features. [Jindal and Liu, 2008] used logistic regression with product, review and reviewer-centric features. [Ott *et al.*, 2011] used n-gram features to train a Naive Bayes and SVM classifier. [Feng *et al.*, 2012], [Mukherjee *et al.*, 2013], [Li *et al.*, 2015] used part-of-speech tags and context free grammar parse trees, behavioral features, spatio-temporal features, respectively.

Neural network methods for spam detection consider the reviews as input without specific feature extraction. GRNN [Ren and Ji, 2017] used a gated recurrent neural network to study the contexual information of review sentences. DRI-RCNN [Zhang *et al.*, 2018] used a recurrent network for learning the contextual information of the words in the reviews. DRI-RCNN extends RCNN [Lai *et al.*, 2015] by learning embedding vectors with respect to both spam and non-spam labels. As RCNN, DRI-RCNN use neural networks, we will compare with these supervised methods in our experiments.

Few semi-supervised methods for opinion spam detection exist. [Li *et al.*, 2011] used co-training with Naive-Bayes classifier on reviewer, product and review features. [Hernández *et al.*, 2013; Li *et al.*, 2014] used only positively labeled samples along with unlabeled data. [Rayana and Akoglu, 2015] used review features, timestamp, ratings as well as pairwise markov random field network of reviewers and product to build a supervised algorithm along with semi-supervised extensions. Other un-supervised methods for spam detection [Xu *et al.*, 2015] exists, but, they are out of the scope of this work.

With respect to GANs for text classification, SeqGAN [Yu *et al.*, 2017] addresses the problem of sparse rewards by considering sequence generation as a Reinforcement Learning problem (RL). Monte Carlo Tree Search (MCTS) is used to overcome the issue of sparse rewards, however, it is computationally intractable. StepGAN [Tuan and Lee, 2018] and MaskGAN [Fedus *et al.*, 2018] use the actor-critic [Konda and Tsitsiklis, 2000] method to learn the rewards, but, they are limited by length of the sequence. Further, all of them focus on sentence generation. CSGAN [Li *et al.*, 2018] deals with sentence classification and incorporates a classifier in its architecture, but performance significantly degrades with sentence length as it uses MCTS and character-level embeddings. spamGAN differs from CSGAN in using the actor-critic reinforcement learning method for sequence generation and word-level embeddings, suitable for longer sentences. The RL architecture in spamGAN helps to mutually bootstrap the generator and classifier while the discriminator and generator are competing. To handle longer sentences, our RL architecture (inspired from stepGAN) has the advantage of requiring only a single pass of the generated sentence through the discriminator and classifier per example, reducing training time.

3 spamGAN

In this section, we will present the problem set-up, the three components of spamGAN as well as their interactions through a sequential decision making framework.

3.1 Problem Set-up

Let $\mathbb{D}_{\mathbb{L}}$ be the set of reviews labeled spam or non-spam. Given the cost of labeling, we hope to improve classification performance by also using $\mathbb{D}_{\mathbb{U}}$, a significantly larger set of unlabeled reviews[2]. Let $\mathbb{D} = \mathbb{D}_{\mathbb{L}} \cup \mathbb{D}_{\mathbb{U}}$ be a combination of labeled and unlabeled sentences for training[3]. Each training sentence $y_{1:T} = \{y_1, y_2, \ldots y_t, \ldots, y_T\}$ consists of a sequence of T word tokens, where $y_t \in \mathtt{Y}$ represents the t^{th} token in the sentence and $\mathtt{Y}$ is a corpus of tokens used. For sentences belonging to $\mathbb{D}_{\mathbb{L}}$, we also include a class label belonging to one of the 2 classes $\mathfrak{c} \in \mathbb{C} : \{\mathtt{spam}, \mathtt{non\text{-}spam}\}$.

To leverage both the labeled and unlabeled data, we include three components in spamGAN: the generator $\mathcal{G}$, the discriminator $\mathcal{D}$, and the classifier $\mathcal{C}$ as shown in Fig. 1. The generator, for a given class label, learns to generate new sentences (we call them `fake`[4] sentences) similar to the real sentences in the train set belonging to the same class. The discriminator learns to differentiate between real and fake sentences, and informs the generator (via rewards) if the generated sentences are unrealistic. This competition between the generator and discriminator improves the quality of the generated sentence.

We know the class labels for the fake sentences produced by the generator as they are controlled [Hu *et al.*, 2017], i.e., constrained by class labels $\{\mathtt{spam}, \mathtt{non\text{-}spam}\}$. The classifier is trained using real labeled sentences from $\mathbb{D}_{\mathbb{L}}$ and fake sentences produced by the generator, thus improving its ability to generalize beyond the small set of labeled sentences. The classifier's performance on fake sentences is also used as feedback to improve the generator: better classification accuracy

[2] $\mathbb{D}_{\mathbb{U}}$ includes both spam/non-spam reviews.

[3] Training (see Alg. 1) can use only $\mathbb{D}_{\mathbb{L}}$ or both $\mathbb{D}_{\mathbb{L}}$ and $\mathbb{D}_{\mathbb{U}}$.

[4] Fake sentences are those produced by the generator. Spam sentences are deceptive sentences with class label `spam`. Generator can generate fake sentences belonging to {`spam` or `non-spam`} class.

results in more rewards. While the discriminator and generator are competing, the classifier and generator are mutually bootstrapping. As the 3 components of spamGAN are trained, the generator produces sentences very similar to the training set while the classifier learns the characteristics of spam and non-spam sentences in order to identify them correctly.

3.2 Generator

If $P_R(y_{1:T}, \mathfrak{c})$ is the true joint distribution of sentences $y_{1:T}$ and classes $\mathfrak{c} \in \mathbb{C}$ from the real training set, the generator aims to find a parameterized conditional distribution $\mathcal{G}(y_{1:T}|z, c, \theta_g)$ that best approximates the true distribution. The generated fake sentence is conditioned on the network parameters θ_g, noise vector z, and class label c, which are sampled from the priors P_z, $P_{\mathfrak{c}}$, respectively. The context vector (consisting of z, c) is concatenated to the generated sentence at every timestep [Tuan and Lee, 2018], so that the actual class labels for each generated fake sentence is retained.

While sampling from $\mathcal{G}(y_{1:T}|z, c, \theta_g)$, the word tokens are generated auto-regressively, decomposing the distribution over token sequences into the ordered conditional sequence,

$$\mathcal{G}(y_{1:T}|z, c, \theta_g) = \prod_{t=1}^{T} \mathcal{G}(y_t|y_{1:t-1}, z, \mathfrak{c}, \theta_g) \tag{1}$$

During pre-training, we use batches of real sentences from $\mathbb{D}$ and minimize the cross-entropy of the next token conditioned on the preceding ones. Specifically, we minimize the loss (Eqn. 2) over real sentence-class pairs $(y_{1:T}, \mathfrak{c})$ from $\mathbb{D}_{\mathbb{L}}$ as well as unlabeled real sentences from $\mathbb{D}_{\mathbb{U}}$ with randomly-assigned class labels drawn from the class prior distribution.

$$\mathcal{L}^{\mathcal{G}}_{MLE} = -\sum_{t=1}^{T} \log \mathcal{G}(y_t|y_{1:t-1}, z, \mathfrak{c}, \theta_g) \tag{2}$$

During adversarial training, we treat sequence generation as a sequential decision making problem [Yu *et al.*, 2017]. The generator acts as a reinforcement learning agent, trained to maximize the expected rewards using policy gradients, where rewards are feedback obtained from discriminator, classifier for the generated sentences (See Sec. 3.5). For implementing the generator, we use a unidirectional multi-layer recurrent neural network with gated recurrent units as the base cell.

3.3 Discriminator

The discriminator $\mathcal{D}$, with parameters θ_d predicts if a sentence is real (sampled from P_R) or fake (produced by the generator) by computing a probability score $\mathcal{D}(y_{1:T}|\theta_d)$ that the sentence is real. Like [Tuan and Lee, 2018] instead of computing the score at the end of the sentence, the discriminator produces scores $Q_{\mathcal{D}}(y_{1:t-1}, y_t)$ for every timestep, which are then averaged to produce the overall score.

$$\mathcal{D}(y_{1:T}|\theta_d) = \frac{1}{T}\sum_{t=1}^{T} Q_{\mathcal{D}}(y_{1:t-1}, y_t) \tag{3}$$

$Q_{\mathcal{D}}(y_{1:t-1}, y_t)$ is the intermediate score for timestep t and is based solely on the preceding partial sentence $y_{1:t}$. In a setup reminiscent of Q-learning, we consider $Q_{\mathcal{D}}(y_{1:t-1}, y_t)$ to be the estimated value for the state $s = y_{1:t-1}$ and action $a = y_t$. Thus, the discriminator provides estimates for the true state-action values without the additional computational overhead of using MCTS rollouts.

We train the discriminator like traditional GANs by maximizing the score $\mathcal{D}(y_{1:T}|\theta_d)$ for real sentences and minimizing it for fake ones. This is achieved by minimizing the loss $\mathcal{L}^{(\mathcal{D})}$,

$$\mathcal{L}^{(\mathcal{D})} = \underset{y_{1:T}\sim P_R}{\mathbb{E}} - \left[\log \mathcal{D}(y_{1:T}|\theta_d)\right] + \underset{y_{1:T}\sim \mathcal{G}}{\mathbb{E}} - \left[\log\left(1 - \mathcal{D}(y_{1:T}|\theta_d)\right)\right] \tag{4}$$

We also include a discrimination critic $\mathcal{D}_{crit}$ [Konda and Tsitsiklis, 2000] which is trained to approximate the score $Q_{\mathcal{D}}(y_{1:t-1}, y_t)$ from the discriminator network, for the next token y_t based on the preceding partial sentence $y_{1:t-1}$. The approximated score $V_{\mathcal{D}}(y_{1:t-1})$ will be used to stabilize policy gradient updates for the generator during adversarial training.

$$V_{\mathcal{D}}(y_{1:t-1}) = \underset{y_t}{\mathbb{E}}\left[Q_{\mathcal{D}}(y_{1:t-1}, y_t)\right] \tag{5}$$

$\mathcal{D}_{crit}$ is trained to minimize the sequence mean-squared error between $V_{\mathcal{D}}(y_{1:t-1})$ and the actual score $Q_{\mathcal{D}}(y_{1:t-1}, y_t)$.

$$\mathcal{L}^{(\mathcal{D}_{\text{crit}})} = \underset{y_{1:T}}{\mathbb{E}} \sum_{t=1}^{T} \left\| Q_{\mathcal{D}}(y_{1:t-1}, y_t) - V_{\mathcal{D}}(y_{1:t-1}) \right\|^2 \tag{6}$$

The discriminator network is implemented as a unidirectional Recurrent Neural Network (RNN) with one dense output layer which produces the probability that a sentence is real at each timestep, i.e., $Q_{\mathcal{D}}(y_{1:t-1}, y_t)$. For the discrimination critic, we have a additional output dense layer (different from the one that computes $Q_{\mathcal{D}}(y_{1:t-1}, y_t)$) attached to the discriminator RNN, which estimates $V_{\mathcal{D}}(y_{1:t-1})$ for each timestep.

3.4 Classifier

Given a sentence $y_{1:T}$, the classifier $\mathcal{C}$ with parameters θ_c predicts if the sentence belongs to class $c \in \mathbb{C}$. Like the discriminator, it assigns a prediction score at each timestep $Q_{\mathcal{C}}(y_{1:t-1}, y_t, c)$ for the partial sentence $y_{1:t}$, which identifies the probability the sentence belongs to class c. The intermediate scores are then averaged to produce the overall score:

$$\mathcal{C}(y_{1:T}, c|\theta_c) = \frac{1}{T}\sum_{t=1}^{T} Q_{\mathcal{C}}(y_{1:t-1}, y_t, c) \tag{7}$$

The classifier loss $\mathcal{L}^{\mathcal{C}}$ is based on: 1) $\mathcal{L}^{(\mathcal{C}_{\text{R}})}$, the cross-entropy loss on true labeled sentences computed using the overall classifier sentence score; 2) $\mathcal{L}^{(\mathcal{C}_{\text{G}})}$ the loss for the fake sentences. Fake sentences are considered as potentially-noisy training examples, so we not only minimize cross-entropy loss but also include Shannon entropy $\mathcal{H}(\mathcal{C}(c|y_{1:T}, \theta_C))$.

$$\mathcal{L}^{\mathcal{C}} = \mathcal{L}^{(\mathcal{C}_{\text{R}})} + \mathcal{L}^{(\mathcal{C}_{\text{G}})} \tag{8}$$

$$\begin{aligned}
\mathcal{L}^{(\mathcal{C}_{\text{R}})} &= \underset{(y_{1:T}, c)\sim P_R(y, \mathfrak{c})}{\mathbb{E}} \left[-\log \mathcal{C}(c|y_{1:T}, \theta_c)\right] \\
\mathcal{L}^{(\mathcal{C}_{\text{G}})} &= \underset{c\sim P_c, y_{1:T}\sim \mathcal{G}}{\mathbb{E}} \big[-\log \mathcal{C}(c|y_{1:T}, \theta_c) \\
&\qquad - \beta\mathcal{H}(\mathcal{C}(c|y_{1:T}, \theta_C))\big]
\end{aligned}$$

In $\mathcal{L}^{(\mathcal{C}_\text{G})}$, β, the balancing parameter, influences the impact of Shannon entropy. Including $\mathcal{H}(\mathcal{C}(c|y_{1:T},\theta_C))$, for minimum entropy regularization [Hu *et al.*, 2017], allows the classifier to predict classes for generated fake sentences more confidently. This is crucial in reinforcing the generator to produce sentences of the given class during adversarial training.

Like in discriminator, we include a classification critic $\mathcal{C}_{crit}$ to estimate the classifier score $Q_{\mathcal{C}}(y_{1:t-1},y_t,c)$ for y_t based on the preceding partial sentence $y_{1:t-1}$,

$$V_{\mathcal{C}}(y_{1:t-1},c) = \mathbb{E}_{y_t}[Q_{\mathcal{C}}(y_{1:t-1},y_t,c)] \tag{9}$$

The implementation of the classifier is similar to the discriminator. We use a unidirectional recurrent neural network with a dense output layer producing the predicted probability distribution over classes $\mathfrak{c} \in \mathbb{C}$. The classification critic is also an alternative head off the classifier RNN with an additional dense layer estimating $V_{\mathcal{C}}(y_{1:t-1},c)$ for each timestep. We train this classification critic by minimizing $\mathcal{L}^{(\mathcal{C}_\text{crit})}$,

$$\mathcal{L}^{(\mathcal{C}_\text{crit})} = \mathbb{E}_{y_{1:T}} \sum_{t=1}^{T} \left\| Q_{\mathcal{C}}(y_{1:t-1},y_t,c) - V_{\mathcal{C}}(y_{1:t-1},c)\right\|^2 \tag{10}$$

3.5 Reinforcement Learning Component

We consider a sequential decision making framework in which the generator acts as a reinforcement learning agent. The current state of the agent is the generated tokens $s_t = y_{1:t-1}$ so far. The action y_t is the next token to be generated, which is selected based on the stochastic policy $\mathcal{G}(y_t|y_{1:t-1},z,c,\theta_g)$. The reward the agent receives for the generated sentence $y_{1:T}$ of a given class $\mathfrak{c}$ is determined by the discriminator and classifier. Specifically, we take the overall scores $\mathcal{D}(y_{1:T}|\theta_d)$ (Eqn.3) and $\mathcal{C}(y_{1:T},c|\theta_c)$ (Eqn. 7) and blend them in a manner reminiscent of the F1 score, producing the sentence reward,

$$R(y_{1:T}) = 2 \cdot \frac{\mathcal{D}(y_{1:T}|\theta_d)\cdot\mathcal{C}(y_{1:T},c|\theta_c)}{\mathcal{D}(y_{1:T}|\theta_d)+\mathcal{C}(y_{1:T},c|\theta_c)} \tag{11}$$

This reward $R(y_{1:T})$ is for the entire sentence delivered during the final timestep, with reward for every other timestep being zero [Tuan and Lee, 2018]. Thus, the generator agent seeks to maximize the expected reward, given by,

$$\mathcal{L}^{(\mathcal{G})} = \mathbb{E}_{y_{1:T}\sim\mathcal{G}}\left[R(y_{1:T})\right] \tag{12}$$

To maximize $\mathcal{L}^{(\mathcal{G})}$, the generator parameters θ_g are updated via policy gradients [Sutton *et al.*, 2000]. Specifically, we use the advantage actor-critic method to solve for optimal policy [Konda and Tsitsiklis, 2000]. The expectation in Eqn. 12 can be re-written using rewards for intermediate timesteps from the discriminator and classifier. The intermediate scores from the discriminator, $Q_{\mathcal{D}}(y_{1:t-1},y_t)$ and the classifier, $Q_{\mathcal{C}}(y_{1:t-1},y_t,c)$, are combined as shown in Eqn. 13 and the combined values serve as estimators for $Q(y_{1:t},c)$, the expected reward for sentence $y_{1:t}$. To reduce variance in the gradient estimates, we replace $Q(y_{1:t},c)$ by the advantage function $Q(y_{1:t},c) - V(y_{1:t-1},c)$, where $V(y_{1:t-1},c)$ is given by Eqn. 13. We use $\alpha = T - t$ in Eqn. 14 to increase the importance of initially-generated tokens while updating θ_g. α is a linearly-decreasing factor which corrects the relative lack of confidence in the initial intermediate scores from the discriminator and classifier.

Algorithm 1: spamGAN

```
Input: Labeled dataset $\mathbb{D}_\mathbb{L}$, Unlabeled dataset $\mathbb{D}_\mathbb{U}$
Parameters: Network parameters $\theta_g$ $\theta_d$ $\theta_c$ $\theta_{dcrit}$ $\theta_{ccrit}$
Perform pre-training as described in Sec. 3.6
for Training-epochs do
    for G-Adv-epochs do
        sample batch of classes $\mathfrak{c}$ from $\sim P(c)$
        generate batch of fake sentences $y_{1:T} \sim \mathcal{G}$ given $\mathfrak{c}$
        for $t \in 1:T$ do
            compute $Q(y_{1:t},c)$, $V(y_{1:t-1},c)$ using Eqn. 13
        update $\theta_g$ using policy gradient $\nabla_{\theta_g}\mathcal{L}^{(\mathcal{G})}$ in Eqn. 14
    for G-MLE-epochs do
        sample batch of real sentences from $\mathbb{D}_\mathbb{L}$, $\mathbb{D}_\mathbb{U}$
        Update $\theta_g$ using MLE in Eqn. 2
    for D-epochs do
        sample batch of real sentences from $\mathbb{D}_\mathbb{L}$, $\mathbb{D}_\mathbb{U}$
        sample batch of fake sentences from $\mathcal{G}$
        update discriminator using $\nabla_{\theta_d}\mathcal{L}^{(\mathcal{D})}$ from Eqn. 4
        compute $Q_{\mathcal{D}}(y_{1:t-1},y_t)$, $V_{\mathcal{D}}(y_{1:t-1})$ for fake sents.
        update $\mathcal{D}_\text{crit}$ using $\nabla_{\theta_{dcrit}}\mathcal{L}^{(\mathcal{D}_\text{crit})}$ from Eqn. 6
    for C-epochs do
        sample batch of real sentences-class pairs from $\mathbb{D}_\mathbb{L}$
        sample batch of fake sentence-class pairs from $\mathcal{G}$
        update classifier using $\nabla_{\theta_c}\mathcal{L}^{(\mathcal{C})}$ from Eqn. 8
        compute $Q_{\mathcal{C}}(y_{1:t-1},y_t,c)$,$V_{\mathcal{C}}(y_{1:t-1},c)$ on fake sents
        update $\mathcal{C}_\text{crit}$ using $\nabla_{\theta_{ccrit}}\mathcal{L}^{(\mathcal{C}_\text{crit})}$ from Eqn. 10
```

$$\begin{aligned} Q(y_{1:t},c) &= 2 \cdot \frac{Q_{\mathcal{D}}(y_{1:t-1},y_t)\cdot Q_{\mathcal{C}}(y_{1:t-1},y_t,c)}{Q_{\mathcal{D}}(y_{1:t-1},y_t)+Q_{\mathcal{C}}(y_{1:t-1},y_t,c)} \\ V(y_{1:t-1},c) &= 2 \cdot \frac{V_{\mathcal{D}}(y_{1:t-1})\cdot V_{\mathcal{C}}(y_{1:t-1},c)}{V_{\mathcal{D}}(y_{1:t-1})+V_{\mathcal{C}}(y_{1:t-1},c)} \end{aligned} \tag{13}$$

During adversarial training, we perform gradient ascent to update the generator using the gradient equation shown below,

$$\begin{aligned} \nabla_{\theta_g}\mathcal{L}^{(\mathcal{G})} = \mathbb{E}_{y_{1:T}} \sum_{t}^{T} \alpha \left[Q(y_{1:t},c) - V(y_{1:t-1},c)\right] \\ \times \nabla_{\theta_g} \log \mathcal{G}(y_t|y_{1:t-1},z,c,\theta_g) \end{aligned} \tag{14}$$

3.6 Pre-Training

Before beginning adversarial training, we pre-train the different components of spamGAN. The generator $\mathcal{G}$ is pre-trained using maximum likelihood estimation (MLE) [Grover *et al.*, 2018] by updating the parameters via Eqn 2. Once the generator is pre-trained, we take batches of real sentences from the labeled dataset $\mathbb{D}_\mathbb{L}$, the unlabeled dataset $\mathbb{D}_\mathbb{U}$ and fake sentences sampled from $\mathcal{G}(y_{1:T}|z,c,\theta_g)$ to pre-train the discriminator minimizing the loss $\mathcal{L}^{(\mathcal{D})}$ in Eqn. 4. The classifier $\mathcal{C}$ is pre-trained solely on real sentences from the labeled dataset $\mathbb{D}_\mathbb{L}$. It is trained to minimize the cross-entropy loss $\mathcal{L}^{(\mathcal{C}_\text{R})}$ on

Method	10% Labeled	30%	50%	70%	90%	100%
spamGAN-0%	0.700 ± 0.02	0.811 ± 0.02	0.838 ± 0.01	0.845 ± 0.01	0.852 ± 0.02	0.862 ± 0.01
spamGAN-50%	0.678 ± 0.03	0.797 ± 0.03	0.839 ± 0.02	0.845 ± 0.02	0.857 ± 0.02	0.856 ± 0.01
spamGAN-70%	0.695 ± 0.05	0.780 ± 0.03	0.828 ± 0.02	0.850 ± 0.01	0.841 ± 0.02	0.844 ± 0.02
spamGAN-100%	0.681 ± 0.02	0.783 ± 0.02	0.831 ± 0.01	0.837 ± 0.01	0.843 ± 0.02	0.845 ± 0.01
Base classifier	0.722 ± 0.03	0.786 ± 0.02	0.791 ± 0.02	0.829 ± 0.01	0.824 ± 0.02	0.827 ± 0.02
DRI-RCNN	0.647 ± 0.10	0.757 ± 0.01	0.796 ± 0.01	0.834 ± 0.18	0.835 ± 0.02	0.846 ± 0.01
RCNN	0.538 ± 0.09	0.665 ± 0.14	0.733 ± 0.09	0.811 ± 0.03	0.834 ± 0.02	0.825 ± 0.02
Co-Train (Naive Bayes)	0.655 ± 0.01	0.740 ± 0.01	0.738 ± 0.02	0.743 ± 0.01	0.754 ± 0.01	0.774 ± 0.01
PU Learn (Naive Bayes)	0.508 ± 0.02	0.713 ± 0.03	0.816 ± 0.01	0.826 ± 0.01	0.838 ± 0.02	0.843 ± 0.02

Table 1: Accuracy (Mean ± Std) for Different % Labeled Data

real sentences and their labels. The critic networks $\mathcal{D}_{\text{crit}}$ and $\mathcal{C}_{\text{crit}}$ are trained by minimizing their loses $\mathcal{L}^{(\mathcal{D}_{\text{crit}})}$ (Eqn. 6) and $\mathcal{L}^{(\mathcal{C}_{\text{crit}})}$ (Eqn. 10). Such pre-training addresses the problem of mode collapse [Guo *et al.*, 2018] to a satisfactory extent.

3.7 spamGAN algorithm

Alg. 1 describes spamGAN in detail. After pre-training, we perform adversarial training for `Training-epochs` (Lines 4-25). We create a batch of fake sentences using generator $\mathcal{G}$ by sampling classes c from prior P_c (Lines 6-7). We compute $Q(y_{1:t}, c)$, $V(y_{1:t-1}, c)$ using Eqn. 13 for every timestep (Line 9). The generator is then updated using policy gradient in Eqn. 14 (Line 10). This process is repeated for `G-Adv-epochs`. Like [Li *et al.*, 2017] the training robustness is greatly improved when the generator is updated using MLE via Eqn 2 on sentences from $\mathbb{D}$ (Lines 11-13). We then train the discriminator using real sentences from $\mathbb{D}_{\mathbb{L}}$, $\mathbb{D}_{\mathbb{U}}$ as well as fake sentences from the generator (Lines 15-16). The discriminator is updated using Eqn. 4 (Line 17). We also train the discrimination critic, by computing $Q_{\mathcal{D}}(y_{1:t-1}, y_t)$, $V_{\mathcal{D}}(y_{1:t-1})$ for the fake sentences and updating the gradients using Eqn. 6 (Line 18-19). This process is repeated for `D-epochs`. We perform a similar set of operations for the classifier (Lines 20-25).

4 Experiments

We use the TripAdvisor labeled dataset [Ott *et al.*, 2011] [5], consisting of 800 truthful reviews on Chicago hotels and 800 deceptive reviews obtained from Amazon Mechanical Turk. We remove a small number of duplicate truthful reviews to get a balanced labeled dataset of 1596 reviews. We augment the labeled set with $32,297$ unlabeled TripAdvisor reviews for Chicago hotels [6]. All reviews are converted to lower-case and tokenized at word level, with a vocabulary `Y` of 10000 [7]. The maximum sequence length $T = 128$ words, close to the median review length of the full dataset. `Y` also includes tokens: $\langle$`start`$\rangle$, $\langle$`end`$\rangle$ which are added to the beginning, end of each sentence, respectively; $\langle$`pad`$\rangle$ for padding sentences smaller than T (longer sentences are truncated, ensuring a consistent sentence length); $\langle$`unk`$\rangle$ replaces out-of-vocabulary words.

In spamGAN, the generator consists of 2 GRU layers of 1024 units each and an output dense layer providing logits for the $10,000$ tokens. The generator, discriminator and classifier are trained using ADAM optimizer. All use variational dropout=0.5 between recurrent layers and word embeddings with dimension 50. For generator, learning rate = 0.001, weight decay =1 $\times 10^{-7}$. Gradient clipping is set to a maximum global norm of 5. The discriminator contains 2 GRU layers of 512 units each and a dense layer with a single scalar output and sigmoid activation. The discrimination critic is implemented as an alternative dense layer. Learning rate =0.0001 and weight decay =1 $\times 10^{-4}$. The classifier is similar to discriminator. We set balancing coefficient $\beta = 1$. The train time of spamGAN using a Tesla P4 GPU was ~ 1.5 hrs.

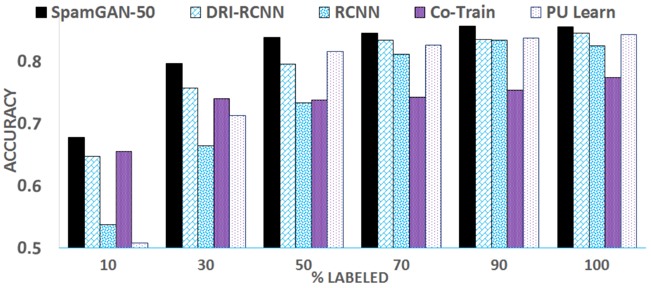

Figure 2: Comparison of spamGAN-50 with Other Approaches

We use a 80/20 train-test split on labeled data. We compare spamGAN with 2 supervised methods: 1) DRI-RCNN [Zhang *et al.*, 2018]; 2) RCNN [Lai *et al.*, 2015] and 2 semi-supervised methods: 3) Co-Training [Li *et al.*, 2011] with Naive Bayes; 4) PU Learning [Hernández *et al.*, 2013] with Naive Bayes (SVM performed poorly) using only spam and unlabeled reviews. We conduct experiments with $10, 30, 50, 70, 90, 100\%$ (proportions) of labeled data. To analyze the impact of unlabeled data, we also adjoin differing amounts of unlabeled data to the labeled data: spamGAN-0 (no unlabeled data), spamGAN-50 (50% unlabeled data), spamGAN-70 (70% unlabeled) and spamGAN-100. Co-Train, PU-Learn results are for 50% unlabeled data. We also show the performance of our base classifier (without generator, discriminator, trained on real labeled data to minimize $\mathcal{L}^{(\mathcal{C}_{\text{R}})}$). All experiments are repeated 10 times and the mean, standard deviation are reported.

Influence of Labeled Data

Table. 1 shows the classification accuracy on the test data. SpamGAN models, in general, outperform other approaches, especially when the % of labeled data is limited. When we merely use 10% of labeled data, spamGAN-0, spamGAN-

[5] http://myleott.com/op-spam.html

[6] http://times.cs.uiuc.edu/ wang296/Data/index.html

[7] Vocabulary includes all words from labeled data and most frequently occurring words from unlabeled data.

Method	10% Labeled	30%	50%	70%	90%	100%
spamGAN-0%	0.718 ± 0.02	0.812 ± 0.02	0.840 ± 0.01	0.848 ± 0.02	0.854 ± 0.02	0.868 ± 0.01
spamGAN-50%	0.674 ± 0.05	0.797 ± 0.03	0.843 ± 0.01	0.848 ± 0.02	0.860 ± 0.02	0.863 ± 0.01
spamGAN-70%	0.702 ± 0.05	0.784 ± 0.03	0.830 ± 0.02	0.856 ± 0.01	0.848 ± 0.02	0.854 ± 0.01
spamGAN-100%	0.684 ± 0.03	0.788 ± 0.03	0.839 ± 0.02	0.844 ± 0.01	0.846 ± 0.02	0.850 ± 0.01
Base classifier	0.731 ± 0.03	0.795 ± 0.03	0.803 ± 0.02	0.829 ± 0.01	0.832 ± 0.02	0.838 ± 0.02
DRI-RCNN	0.632 ± 0.07	0.754 ± 0.02	0.779 ± 0.00	0.812 ± 0.03	0.817 ± 0.03	0.833 ± 0.02
RCNN	0.638 ± 0.01	0.715 ± 0.01	0.754 ± 0.02	0.776 ± 0.05	0.820 ± 0.03	0.833 ± 0.02
Co-Train (Naive Bayes)	0.637 ± 0.02	0.698 ± 0.01	0.680 ± 0.02	0.677 ± 0.01	0.712 ± 0.01	0.726 ± 0.01
PU-Learn (Naive Bayes)	0.050 ± 0.02	0.636 ± 0.05	0.815 ± 0.02	0.837 ± 0.02	0.844 ± 0.02	0.852 ± 0.01

Table 2: F1-Score (Mean ± Std) for Different % Labeled Data

50, spamGAN-70, spamGAN-100 achieve an accuracy of $0.70, 0.678, 0.695, 0.681$, respectively, higher than supervised approaches DRI-RCNN (0.647), R-CNN (0.538) and semi-supervised approaches Co-train (0.655), PU-learning (0.508). Even without unlabeled data spamGAN-0 gets good results because the mutual bootstrapping between generator and classifier allows the classifier to explore beyond the small labeled training set using the fake sentences produced by the generator. Our base classifier has a higher value (0.722) than spamGAN models as GANs needs more samples to train, in general.

The accuracy of all approaches increases with % of labeled data. We select spamGAN-50 as a representative for comparison in Fig. 2. Though the difference in accuracy between spamGAN-50 and others reduces as the % of labeled data increases, spamGAN-50 still performs better than others with an accuracy of 0.856 when all labeled data are considered.

Table. 2 shows the F1-score. We can again see that spamGAN-0, spamGAN-50 and spamGAN-70 perform better than the others, especially when the % of labeled data is small.

Influence of Unlabeled Data

While unlabeled data is used to augment the classifier's performance, Fig. 3 shows that F1-score slightly decreases when the % unlabeled data increases, especially for spamGAN-100. In our case, as unlabeled data is much larger than the labeled, the generator does not entirely learn the importance of the sentence classes during pre-training (when the unlabeled sentence classes are randomly assigned), which causes problems for the classifier during adversarial training. However, when no unlabeled data is used, the generator easily learns to generate sentences conditioned on classes paving way for mutual bootstrapping between classifier and generator. We also attribute the drop in performance to the difference in distribution of data between the unlabeled TripAdvisor reviews and the handcrafted reviews from Amazon MechanicalTurk (unlabled data can improve performance only under assumptions about data distributions [Wasserman and Lafferty, 2008]).

Perplexity of Generated Sentence

We also compute the perplexity of the sentences produced by the generator (the lower the value the better). Fig. 4 shows that as the % of unlabeled data increases (spamGAN-0 to spamGAN-100), the perplexity of the sentences decreases. SpamGAN-100, SpamGAN-70 achieve a perplexity of $76.4, 76.5$, respectively. Fig. 3, Fig. 4 show that using unlabeled data improves the generator in producing realistic sentences but does not fully help to differentiate between the classes which again, can be attributed to the difference in the data distribution between the labeled and unlabeled data.

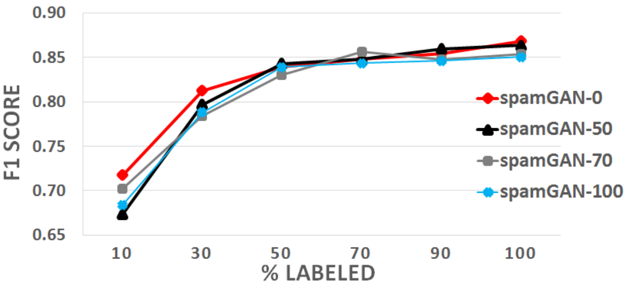

Figure 3: Influence of Unlabeled Data on F1-Score

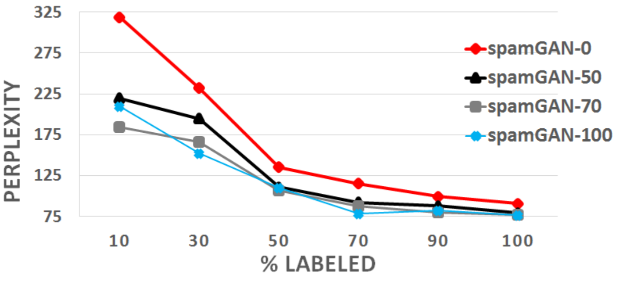

Figure 4: Influence of Unlabeled Data on Perplexity

Following is a sample (partial) spam sentence produced by the generator: "Loved this hotel but i decided to the hotel in a establishment didnt look bad ...the palmer house was anyplace that others said in the reviews..". We notice that spam sentences use more conservative choice of words, focusing on adjectives, reviewer, and attributes of the hotel, while non-spam sentences speak more about the trip in general.

5 Conclusion and Future Work

We propose spamGAN, an approach for detecting opinion spam with limited labeled data. spamGAN also helps to generate reviews similar to the training set. Experiments show that spamGAN outperforms state-of-the-art supervised and semi-supervised techniques when labeled data is limited. We further plan to conduct experiments on YelpZip data (overcoming the data distribution issue of MechanicalTurk reviews). As the overall spamGAN architecture is agnostic to the implementation details of classifier, we plan to use a more sophisticated design for the classifier than a simple recurrent network.

References

[Crawford *et al.*, 2015] Michael Crawford, Taghi M Khoshgoftaar, Joseph D Prusa, Aaron N Richter, and Hamzah Al Najada. Survey of review spam detection using machine learning techniques. *Journal of Big Data*, 2(1):23, 2015.

[Fedus *et al.*, 2018] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: Better text generation via filling in the _. *ICLR*, 2018.

[Feng *et al.*, 2012] Song Feng, Ritwik Banerjee, and Yejin Choi. Syntactic stylometry for deception detection. In *ACL*, 2012.

[Grover *et al.*, 2018] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In *AAAI*, 2018.

[Guo *et al.*, 2018] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In *AAAI*, 2018.

[Hernández *et al.*, 2013] Donato Hernández, Rafael Guzmán, Manuel Móntes y Gomez, and Paolo Rosso. Using pu-learning to detect deceptive opinion spam. In *Workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 38–45, 2013.

[Hu *et al.*, 2017] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. *arXiv preprint arXiv:1703.00955*, 2017.

[Huszár, 2015] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

[Jindal and Liu, 2008] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *WSDM*, 2008.

[Jindal *et al.*, 2010] Nitin Jindal, Bing Liu, and Ee-Peng Lim. Finding unusual review patterns using unexpected rules. In *CIKM*, 2010.

[Konda and Tsitsiklis, 2000] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *NIPS*, 2000.

[Kumar *et al.*, 2017] Abhishek Kumar, Prasanna Sattigeri, and Tom Fletcher. Semi-supervised learning with gans: manifold invariance with improved inference. In *NIPS*, 2017.

[Lai *et al.*, 2015] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, 2015.

[LeCun *et al.*, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[Li *et al.*, 2011] Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. Learning to identify review spam. In *IJCAI*, 2011.

[Li *et al.*, 2014] Huayi Li, Bing Liu, Arjun Mukherjee, and Jidong Shao. Spotting fake reviews using positive-unlabeled learning. *Computación y Sistemas*, 18(3):467–475, 2014.

[Li *et al.*, 2015] Huayi Li, Zhiyuan Chen, Arjun Mukherjee, Bing Liu, and Jidong Shao. Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns. In *AAAI-ICWSM*, 2015.

[Li *et al.*, 2017] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial Learning for Neural Dialogue Generation. 2017.

[Li *et al.*, 2018] Yang Li, Quan Pan, Suhang Wang, Tao Yang, and Erik Cambria. A generative model for category text generation. *Information Sciences*, 450:301–315, 2018.

[Mukherjee *et al.*, 2013] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. What yelp fake review filter might be doing? In *AAAI-ICWSM*, 2013.

[Ott *et al.*, 2011] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *ACL*, 2011.

[Rayana and Akoglu, 2015] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *KDD*, 2015.

[Ren and Ji, 2017] Yafeng Ren and Donghong Ji. Neural networks for deceptive opinion spam detection: An empirical study. *Information Sciences*, 385:213–224, 2017.

[Sutton *et al.*, 2000] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 2000.

[Tuan and Lee, 2018] Yi-Lin Tuan and Hung-Yi Lee. Improving conditional sequence generative adversarial networks by stepwise evaluation. *arXiv:1808.05599*, 2018.

[Wasserman and Lafferty, 2008] Larry Wasserman and John D Lafferty. Statistical analysis of semi-supervised regression. In *NIPS*, 2008.

[Xu *et al.*, 2015] Yinqing Xu, Bei Shi, Wentao Tian, and Wai Lam. A unified model for unsupervised opinion spamming detection incorporating text generality. In *AAAI*, 2015.

[Yu *et al.*, 2017] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, 2017.

[Zhang *et al.*, 2018] Wen Zhang, Yuhang Du, Taketoshi Yoshida, and Qing Wang. Dri-rcnn: An approach to deceptive review identification using recurrent convolutional neural network. *Information Processing & Management*, 54(4):576–592, 2018.