Learning Adversarial Networks for Semi-Supervised Text Classification via Policy Gradient

Yan Li
Dept. of Computational Medicine and Bioinformatics
University of Michigan
Ann Arbor, MI, 48109
yanliwl@umich.edu

Jieping Ye
Dept. of Computational Medicine and Bioinformatics
University of Michigan
Ann Arbor, MI, 48109
jpye@umich.edu

ABSTRACT

Semi-supervised learning is a branch of machine learning techniques that aims to make fully use of both labeled and unlabeled instances to improve the prediction performance. The size of modern real world datasets is ever-growing so that acquiring label information for them is extraordinarily difficult and costly. Therefore, deep semi-supervised learning is becoming more and more popular. Most of the existing deep semi-supervised learning methods are built under the generative model based scheme, where the data distribution is approximated via input data reconstruction. However, this scheme does not naturally work on discrete data, e.g., text; in addition, learning a good data representation is sometimes directly opposed to the goal of learning a high performance prediction model. To address the issues of this type of methods, we reformulate the semi-supervised learning as a model-based reinforcement learning problem and propose an adversarial networks based framework. The proposed framework contains two networks: a predictor network for target estimation and a judge network for evaluation. The judge network iteratively generates proper reward to guide the training of predictor network, and the predictor network is trained via policy gradient. Based on the aforementioned framework, we propose a recurrent neural network based model for semi-supervised text classification. We conduct comprehensive experimental analysis on several real world benchmark text datasets, and the results from our evaluations show that our method outperforms other competing state-of-the-art methods.

CCS CONCEPTS

• **Computing methodologies** → **Adversarial learning**; **Semi-supervised learning settings**; **Neural networks**; **Policy iteration**;

KEYWORDS

Semi-supervised Learning, Adversarial Networks, Policy Gradients, Text Classification

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219956

ACM Reference Format:
Yan Li and Jieping Ye. 2018. Learning Adversarial Networks for Semi-Supervised Text Classification via Policy Gradient. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3219956

1 INTRODUCTION

In many real world applications, labeling data for a learning problem often requires numerous efforts from skilled human experts. Therefore, it is very expensive or even impossible to acquire a large amount of labeled data. However, acquiring a large amount of unlabeled data is relatively easy and inexpensive. Semi-supervised learning [41] is a branch of machine learning methods, which aims at utilizing the surplus unlabeled data with a small amount of labeled data to improve the accuracy of prediction models.

The latest advances in deep learning technologies [17] provide new sophisticated paradigms to obtain end-to-end learning models from complex data. In the context of deep learning, most of the commonly used semi-supervised learning algorithms [8, 9, 15, 27, 30, 37] are built based on the generative model based scheme. Under this scheme, the deep generative model is employed as an approximator of the data distribution, and the learned distribution is used as auxiliary information to augment the training process of classification model [8, 30]. Goodfellow et al. introduce the generative adversarial networks (GAN) [11] framework, where two networks are trained to contest with each other and play a zero-sum game. Specifically, a discriminator network D is trained to distinguish whether a given data instance is real or not, and a generative network G is trained to fool D by generating high quality data. This adversarial learning framework has achieved great success in computer vision tasks, and has been successfully extended to semi-supervised image classification [9, 27, 30]. However, this framework cannot be directly extented to semi-supervised text classification as GAN is designed for generating continuous data, which does not naturally work on discrete data generation, e.g., text [39].

To overcome the natural limitation of GAN and take advantage of the adversarial training framework in the context of semi-supervised text classification, we propose a discriminative adversarial networks (DAN) [28] based approach, which roots from self-training. Self-training [25, 38] is the most straightforward scheme for semi-supervised learning. It is built based on a heuristic approach where the model is bootstrapped with additional labeled data obtained from its own highly confident predictions. Thus, its performance is unstable since poor predictions might be reinforced. In this paper, we bridge the idea of self-training and adversarial

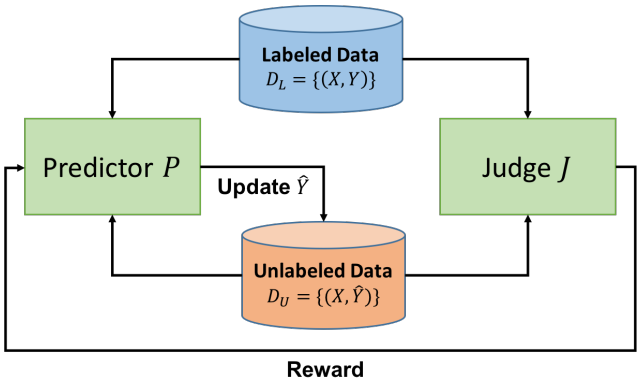


Figure 1: Illustration of the proposed framework. The predictor network P is trained with both labeled and unlabeled data to predict a label $\hat{y}_i$ given a data point x_i. P is trained by policy gradient where the reward signal is provided by the judge network J, which determines whether a certain input data-label pair has a predicted label or a true label. The J is trained using both the real data-label pairs $\{(x_i, y_i) \in D_L\}$ as positive examples and the predicted data-label pairs $\{(x_i, \hat{y}_i)|x_i \in D_U\}$ as negative examples.

networks to overcome their issues. Specifically, the model built under self-training framework does not need to approximate the data distribution via instance reconstruction and hence it overcomes the limitation of GAN-based semi-supervised learning methods. On the other hand, inspired by the adversarial networks, a judge network J is introduced to self-training for telling apart whether a given label of a certain data instance is real or not, and hence reduces the risk of reinforce poor predictions and make the self-training become more stable and robust.

To bridge the aforementioned two learning frameworks together and combine the merits of them, in this paper we propose the **RLANS** framework, which stands for "**R**einforcement **L**earning based **A**dversarial **N**etworks for **S**emi-supervised learning". The proposed framework formulates the predictor network P of semi-supervised learning as a reinforcement learning (RL) agent [32], where state is the input data and action is the label prediction. Thus, the primary goal of learning problem is transformed as learning a good policy of P, such that the generated predicted label can maximize the expected total reward. The predictor network P is learned via policy gradient [33]. The judge network J is used to evaluate the predicted label and provide the feedback of evaluation to guide the learning of predictor network. Adopting the feedback of J as reward can iteratively improve the predictor network P because the reward is updated dynamically. Specifically, let $D_L = \{(x_1, y_1), \cdots, (x_l, y_l)|x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$ be a set of l labeled instances and $D_U = \{x_{l+1}, \cdots, x_{l+u}|x_i \in \mathcal{X}\}$ be a set of u unlabeled instances. The overall structure of the proposed model is shown in Figure 1.

Reinforcement learning is known to be unstable or even to diverge when the action-value function is represented by a nonlinear function, e.g., neural network [22]. In the model implementation, for the sake of alleviating instability, the predictor P is pre-trained via applying maximum likelihood estimation on the D_L for several iterations, and the judge J is pre-trained with several iterations once the pre-training of predictor P is finished. In the experiment part of this paper, we demonstrate the prediction performance of the proposed **RLANS** framework using several well-known public text datasets under the semi-supervised setting. Our model attains very competitive prediction performance and outperforms some state-of-the-art related methods especially when the number of labeled instances is limited.

The main contributions of this paper can be summarized as follows:

- Inspired by self-training, in order to take into account both labeled and unlabeled instances, we novelly formulate and interpret the semi-supervised learning as a model-based reinforcement learning problem.
- We propose an adversarial networks based framework for semi-supervised learning. Unlike most of the other GAN-based semi-supervised learning approaches, the proposed framework dose not need to reconstruct input data and hence can be applied for semi-supervised text classification.
- Based on the proposed **RLANS** framework, we propose a concrete model for semi-supervised text classification. In addition, comprehensive experimental analysis is conducted on several benchmark datasets.

The rest of this paper is organized as follows. Section 2 discusses the related semi-supervised learning approaches. Section 3 presents our proposed **RLANS** framework and a concrete model for semi-supervised text classification. In section 4, we demonstrate our experimental results on several real-world benchmark datasets along with the implementation details. Finally, section 5 concludes our discussion and gives some future research directions for the proposed work.

2 RELATED WORK

Semi-supervised learning can trace back to 1970s, and it attracts extensive attention since 1990s [41]. Existing semi-supervised learning methods can be roughly categorized into the following different types, namely, self-training [25, 38], transductive learning based [14], co-training [4], graph-based [3, 42], and generative model based [23] schemes. In this section, we will first briefly describe the most important works under each category, and then highlight the relationship and primary distinctions of our proposed framework compared to the existing methods that are available in the literature.

Among all aforementioned types of semi-supervised learning schemes, the most straightforward scheme is self-training [25, 38], where the predictor model is iteratively re-trained with both the labeled instances and the most confident predictions. Transductive support vector machine (TSVM) [14] is a well-known transductive approach for semi-supervised learning. It extends support vector machine (SVM) and aims at finding the discriminative decision bounder that lies in a low density region and hence has the maximum margin on both original labeled data and unlabeled data. Co-training [4] is a special case of multi-view learning [29], which assumes the data has two views and each view is sufficient enough to train a good predictor. Initially, a separate classifier for each

view is trained with labeled instances. Then the most confident predictions of each classifier on unlabeled data are combined with labeled data to iteratively re-train the other classifier. Graph-based semi-supervised methods [1, 3, 42] build a similarity graph for both labeled and unlabeled instances, and the similar instances are assumed to be in the same class. Different methods such as Gaussian fields [42] and Hidden Markov Random Fields (HMRFs) [1] are introduced to graph-based semi-supervised learning to model the label propagation within the graph.

The generative model based semi-supervised learning scheme can be viewed as an extension of supervised learning combined with auxiliary information of the data distribution, which is learned via reconstruction of input samples. In the previous decade, some traditional generative mixture models [7, 10] have been used for semi-supervised learning. With the development of deep learning methods, deep neural networks have been employed as density estimator [26] and hence are more flexible and powerful than traditional generative models. Deep generative models such as variational autoencoder (VAE) [16] and GAN [11] have achieved impressive success in recent years. As a result, several VAE based [15, 37] and GAN based [9, 27, 30] deep semi-supervised learning methods have been proposed recently. In such reconstruction-based semi-supervised learning methods, the generator is trained to learn representations that preserve all information of the input examples to achieve perfect data reconstruction, and it is used to pre-train the classification network. However, learning a classifier is a process of mining valuable information for label prediction which is opposed to the goal of perfect data reconstruction [30], so that sometimes the learned representations may even hinder the performance of prediction model [6]. For example, in [27] users have to choose a model either to get a good data representation or to obtain a high performance classifier, but not both; moreover, in [9] the authors provide an argument that good semi-supervised learning requires a bad GAN.

Based on the reconstruction-based semi-supervised learning scheme, in [8] a two-stage approach is proposed for semi-supervised sequence learning. Firstly, in the pre-training stage, a conventional unsupervised language model [2] is built to learn a vector representation of sequence, which predicts what comes next in a sequence. Secondly, the model parameters obtained from the pre-training stage is used as a starting point for the supervised training models in text classification. Based on this scheme, in [20] the authors augment the second stage via introducing adversarial perturbations [12] in the training step to further improve the model performance. Note that, different from adversarial networks, introducing adversarial perturbations is a recent proposed regularization method for classifiers to improve model robustness [12]. In these papers, unlabeled instances are not directly involved in the second stage to train the classifier. Recently in [28] a DAN based approach is proposed for semi-supervised learning; however, labeled instances are only used to train the judge network, but not used in the predictor network.

In this paper, different from all the above mentioned deep semi-supervised learning models, we formulate the semi-supervised learning as a model-based reinforcement learning problem, and propose an adversarial networks based framework to improve the training process. The main differences between the proposed **RLANS** and the aforementioned deep semi-supervised learning approaches can be summarized as follows:

- Different from deep generative semi-supervised learning models, the proposed **RLANS** framework does not need to perform instances reconstruction for distribution approximation. Therefore, the proposed **RLANS** framework can be easily applied in the semi-supervised text classification.
- Different from the existing deep semi-supervised text classification algorithms, the proposed **RLANS** framework takes advantage of both reinforcement learning and adversarial networks. Therefore, it can be updated iteratively and both labeled and unlabeled samples are directly used to train the classifier.

3 METHODS

In this section, we introduce the proposed adversarial learning framework for semi-supervised learning in detail. We first describe the overall structure of the proposed **RLANS** framework and then describe a corresponding concrete model for semi-supervised text classification.

3.1 Model Overview

The primary goal of semi-supervised learning is to learn a predictor model P_θ parameterized by θ from both labeled data D_L and unlabeled data D_U, such that the predicted label is as close as possible to the true label. As the labeling information of unlabeled instances is unknown, it is not straightforward to train the predictor P_θ based on the commonly used maximum likelihood estimation (MLE). Instead, in this work we formulate the semi-supervised learning as a reinforcement learning problem.

3.1.1 **Training the predictor model P_θ.** We interpret the prediction problem based on reinforcement learning, where x can be viewed as the state and its corresponding predicted label $\hat{y}$ is the action. Predictor $P_\theta(\hat{y}|x)$ can be viewed as a policy model, which determines the probability that action $\hat{y}$ is chosen given the state x with the model parameter θ. The objective of policy model is to generate the proper predicted label to maximize its expected reward:

$$R(\theta) = \mathbb{E}[R|X, \theta] = \sum_{\hat{y} \in \mathcal{Y}} P_\theta(\hat{y}|x) V(\hat{y}, x), \tag{1}$$

where $\mathcal{Y}$ is the feasible action space and $V(\cdot)$ is the action-value function of choosing $\hat{y}$ as an action. In the semi-supervised learning, given the input data a good predictor should generate a predicted label that is as close as possible to the true label. Therefore, the action-value function in our problem is defined as the similarity between the predicted label $\hat{y}$ and the true label y.

Now the key problem is how to define a proper similarity function between $\hat{y}$ and y, especially for the unlabeled instances whose true labels are unknown. To address this issue, inspired by the adversarial learning framework GAN [11] we train a ϕ-parameterized discriminative model J_ϕ as the judge to provide a guidance for improving predictor P_θ, which will be discussed in detail in the following section. $J_\phi(x, \hat{y})$ is a probability that indicates how likely the $(x, \hat{y})$ is considered as true data-label pair. Therefore, in our

proposed framework we define the action-value function as:

$$V(\hat{y}, x) = \begin{cases} 1 & \text{if} \quad x \in D_L \ \& \ \hat{y} = y, \\ 0 & \text{if} \quad x \in D_L \ \& \ \hat{y} \neq y, \\ J_\phi(x, \hat{y}) & \text{if} \quad x \in D_U. \end{cases} \tag{2}$$

The key advantage of employing $J_\phi(x, \hat{y})$ in the action-value function is that J_ϕ is dynamically updated and hence it can further improve the predictor P_θ iteratively. Note that, the above defined action-value function provides an immediate reward per iteration; thus, we do not need to employ additional techniques, e.g., Monte Carlo (MC) tree search [5] and Temporal-Difference (TD) learning [34], to approximate the long-term rewards.

Maximizing the objective in Eq. (1) requires computation of its gradient w.r.t. the model parameter θ:

$$\nabla_\theta R(\theta) = \sum_{\hat{y} \in \mathcal{Y}} \nabla_\theta P_\theta(\hat{y}|x) V(\hat{y}, x). \tag{3}$$

Using likelihood ratio trick proposed in the REINFORCE algorithm [36], Eq. (3) can be further rewritten as:

$$\begin{aligned} \nabla_\theta R(\theta) &= \sum_{\hat{y} \in \mathcal{Y}} P_\theta(\hat{y}|x) \frac{\nabla_\theta P_\theta(\hat{y}|x)}{P_\theta(\hat{y}|x)} V(\hat{y}, x) \\ &= \sum_{\hat{y} \in \mathcal{Y}} P_\theta(\hat{y}|x) \ \nabla_\theta \log P_\theta(\hat{y}|x) V(\hat{y}, x) \\ &= \mathbb{E}_{P_\theta(\hat{y}|x)} \left[\nabla_\theta \log P_\theta(\hat{y}|x) V(\hat{y}, x) \right]. \end{aligned} \tag{4}$$

Eq.(4) is an unbiased estimation for Eq.(3). In practice, when we apply mini-batch training with m number of labeled instances and m number of unlabeled instances, an approximated gradient can be computed as:

$$\nabla_\theta R(\theta) \approx \frac{1}{2m} \sum_{i=1}^{2m} \left[\nabla_\theta \log P_\theta(\hat{y}_i|x_i) V(\hat{y}_i, x_i) \right], \tag{5}$$

and then the predictor's model parameter θ can be updated as:

$$\theta^+ = \theta^- + \alpha \ \nabla_\theta \ R(\theta^-), \tag{6}$$

where $\alpha \in \mathbb{R}^+$ denotes the corresponding learning rate, θ^+ and θ^- indicate the updated and current model parameters, respectively.

3.1.2 **Training the judge model J_ϕ.** As mentioned above, the judge model J_ϕ is introduced to estimate the probability that an input data-label pair is a true data-label pair. In the proposed framework, J_ϕ is trained via using a set of true labeled instances $\{(x_i, y_i) \in D_L\}$ as positive examples and a set of unlabeled instances with their corresponding predicted labels $\{(x_i, \hat{y}_i)|x_i \in D_U, \hat{y}_i = P_\theta(x_i)\}$ as negative examples. J_ϕ should discriminate the positive and negative examples as clear as possible. Therefore, the judge model is trained to minimize the cross-entropy:

$$\min_\phi -\mathbb{E}_{(x,y) \in D_L} \left[\log J_\phi(x, y) \right] - \mathbb{E}_{\substack{x \in D_U \\ \hat{y} \sim P_\theta}} \left[\log(1 - J_\phi(x, \hat{y})) \right]. \tag{7}$$

Algorithm 1 summarizes the overall learning process of the proposed framework. Before the adversarial training, in Line 1 the predictor model P_θ is pre-trained via MLE on the labeled dataset D_L, and in Line 3 the judge model J_ϕ is pre-trained via minimizing the cross entropy on both true labeled instances and fake/predicted labeled instances (generated in Line 2). After the pre-training, in each adversarial training loop, the predictor model P_θ will be applied on m number of unlabeled samples to get the predicted labels (Line 6). The judge model J_ϕ will be trained according to both the true data-label pairs, i.e., $\{(x_i, y_i) \in D_L\}$, and the predicted data-label pairs, i.e., $\{(x_i, \hat{y}_i)|x_i \in D_U\}$ (Line 8). Each time when a new judge model is obtained, we can calculate the updated action-value function in Eq.(5), and then in Line 11 the predictor model P_θ is updated via the policy gradient method.

Algorithm 1: RLANS algorithm.

Input: D_L and D_U
Output: P_θ and J_ϕ

1 Pre-train the predictor P_θ via MLE on D_L;
2 Generate the predicted labels for all instances in D_U via P_θ;
3 Pre-train the judge J_ϕ via minimizing the cross entropy ;
4 **for** *number of training iterations* **do**
5 Sample m labeled instances from D_L;
6 Sample m unlabeled instances from D_U and predict their coresponding label via P_θ;
7 **for** k *steps* **do**
8 Update the judge model J_ϕ via Eq.(7) ;
9 Update $V(\hat{y}, x)$ based on Eq. (2);
10 Calculate the gradient of expected reward via Eq. (5);
11 Update the predictor P_θ via policy gradient Eq. (6);
12 **end**
13 **end**

3.2 RLANS for Semi-supervised Text Classification

Based on the above mentioned **RLANS** framework, we propose a concrete model for semi-supervised text classification.

3.2.1 **The predictor network for text.** In this paper, we use a standard Long Short Term Memory (LSTM) network [13, 31] based model as our predictor model, shown in Figure 2.

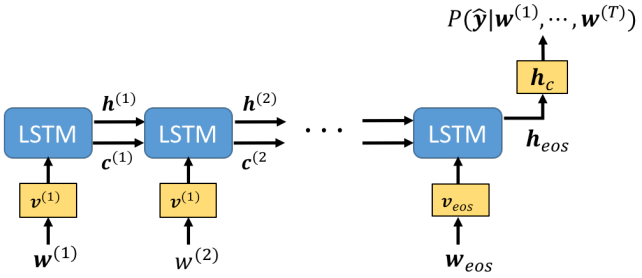


Figure 2: LSTM based text classification model.

Let an instance $\boldsymbol{x} = \{\boldsymbol{w}^{(1)}, \cdots, \boldsymbol{w}^{(T)} | \boldsymbol{w}^{(t)} \in \{0, 1\}^k\}$ be a sequence of one-hot representation vectors of T words, and its corresponding target $\boldsymbol{y} \in \{0, 1\}^c$ is encoded as the one-hot representation of the c-class classification label, where k is the number of unique words in the vocabulary. An embedding matrix $\boldsymbol{E} \in \mathbb{R}^{(k+1) \times p}$ is

used to transform the original one-hot encoding word representation into the corresponding p-dimensional continuous vector $\{\boldsymbol{v}^{(1)}, \cdots, \boldsymbol{v}^{(T)}\}$, and the $(k+1)$-th word embedding is used as an "end of sequence (eos)" indicator token, $\boldsymbol{v}_{eos}$. Given the input word $\boldsymbol{w}^{(t)}$, long term cell state $\boldsymbol{c}^{(t-1)} \in \mathbb{R}^{1\times q}$ and hidden state $\boldsymbol{h}^{(t-1)} \in \mathbb{R}^{1\times q}$ at the $(t-1)$-th step, $\boldsymbol{c}^{(t)}$ and $\boldsymbol{h}^{(t)}$ will be calculated at the t-th step. At the final step, given the final hidden state $\boldsymbol{h}_{eos}$, the model first calculates a hidden vector $\boldsymbol{h}_c \in \mathbb{R}^{1\times d}$ through fully connected layer with the rectified linear unit (ReLU) active function, and then a softmax output layer is used to calculate the corresponding estimated label distribution:

$$P_\theta(\hat{\boldsymbol{y}}_i = 1|\boldsymbol{x}) = P_\theta(\hat{\boldsymbol{y}}_i = 1|\boldsymbol{w}^{(1)}, \cdots, \boldsymbol{w}^{(T)}) = \frac{\exp(\boldsymbol{h}_c \cdot \boldsymbol{B}_{(:,i)} + \boldsymbol{b}_p^{(i)})}{\sum_{j=1}^{c} \exp(\boldsymbol{h}_c \cdot \boldsymbol{B}_{(:,j)} + \boldsymbol{b}_p^{(j)})}, \quad \forall i = 1, \cdots, c \quad (8)$$

where the weight matrix $\boldsymbol{B} \in \mathbb{R}^{d\times c}$ and bias vector $\boldsymbol{b}_p \in \mathbb{R}^{1\times c}$ are the corresponding parameters in the softmax layer. Moreover, $\boldsymbol{B}_{(:,i)}$ indicates the i-th column of $\boldsymbol{B}$ and $\boldsymbol{b}_p^{(i)}$ indicates the i-th element of $\boldsymbol{b}_p$.

3.2.2 **The judge network for text.** In this paper, we use a LSTM based model as our judge model, shown in Figure 3.

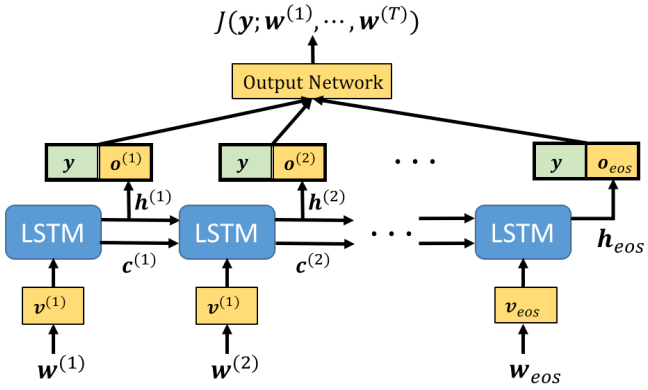


Figure 3: LSTM based judge model.

At the t-th step, a current estimated label vector $\boldsymbol{o}^{(t)} \in \mathbb{R}^{1\times c}$ is generated via a sub-network, which has a structure same as the output part of the aforementioned predictor network. We then concatenate $\boldsymbol{o}^{(t)}$ with the one-hot embedded target vector $\boldsymbol{y}$ (or with the estimated target vector $\hat{\boldsymbol{y}}$ of unlabeled instances). Once this type of concatenated vectors has been generated at all time steps, a weighted combination:

$$[\boldsymbol{o}, \boldsymbol{y}]_{\boldsymbol{\beta}} = \sum_{t=1}^{T} \beta_t \left[\boldsymbol{o}^{(t)}, \boldsymbol{y}\right]$$

is treated as the input of output network of the judge model, where $\boldsymbol{\beta} \in \mathbb{R}^T$ is the trainable weight vector.

The goal of the judge model is to estimate the probability that how likely the $(\boldsymbol{y}; \boldsymbol{w}^{(1)}, \cdots, \boldsymbol{w}^{(T)})$ is from the set of labeled instances D_L, which is the joint probability of the two components in the $[\boldsymbol{o}, \boldsymbol{y}]_{\boldsymbol{\beta}}$. A single-layer neural network treats all the input features independently and hence fails to take into account feature interaction. Therefore, the designed output network should have multiple layers. In our model, the designed output network has two layers. The first layer is:

$$\boldsymbol{o}_1 = \text{ReLu}([\boldsymbol{o}, \boldsymbol{y}]_{\boldsymbol{\beta}} \cdot \boldsymbol{W}_{o1} + \boldsymbol{b}_{o1}), \quad (9)$$

which uses ReLU as active function with the corresponding weight matrix $\boldsymbol{W}_{o1} \in \mathbb{R}^{2c\times 2c}$ and the bias vector $\boldsymbol{b}_{o1} \in \mathbb{R}^{1\times 2c}$. The second layer is a sigmoid function:

$$J(\boldsymbol{y}; \boldsymbol{w}^{(1)}, \cdots, \boldsymbol{w}^{(T)}) = \frac{1}{1 + \exp(-(\boldsymbol{o}_1 \cdot \boldsymbol{W}_{o2} + b_J))}, \quad (10)$$

which calculates how likely the $(\boldsymbol{y}; \boldsymbol{w}^{(1)}, \cdots, \boldsymbol{w}^{(T)})$ is a true labeled instance, where $\boldsymbol{W}_{o2} \in \mathbb{R}^{2c\times 1}$ is the corresponding coefficient vector and b_J is a bias scalar.

4 EXPERIMENTS

In this section, we will first describe the datasets used in our evaluation and then provide the performance results along with the implementation details.

4.1 Dataset description

To compare our proposed method with other semi-supervised text classification methods, we conduct comprehensive empirical analysis on several public benchmark datasets:

- **AG's news corpus**[1], contains news articles from more than 2,000 news sources. In the experiment, we use the dataset with 4 largest classes constructed in [40]. The original dataset contains 30,000 training samples and 1,900 testing samples in each class.
- **DBPedia ontology dataset**[1], is a dataset of Wikipedia pages for category classification. Specifically, we use the dataset constructed in [18] that picks 14 non-overlapping classes from DBpedia 2014. The original dataset contains 40,000 training samples and 5,000 testing samples in each class.
- **IMDB movie review dataset**[2], is a benchmark movie review dataset for binary sentiment classification [19]. The original dataset contains 12,500 training samples, 12,500 testing samples in each class, and additional 50,000 unlabeled samples.
- **Yelp full reviews dataset**[1], is obtained from the Yelp Dataset Challenge in 2015, which contains the reviewer texts and the corresponding rating range from 1 star to 5 stars. The original dataset contains 130,000 training samples and 10,000 testing samples in each class.

Table 1 summarizes the details of the experimental datasets. The column titled "# Classes" corresponds to the number of classes, the column titled " # Test /Class" is the number of testing samples per class. "Ave. Len" and "Max Len" correspond to the average sequence lengths and maximum sequence lengths, respectively. We conduct the experiment with different number of labeled and unlabeled instances, and the detailed information can be found in the experiment setup.

[1]Download from: http://goo.gl/JyCnZq
[2]Download from: http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz

Table 1: Summary of experimental datasets.

Dataset	#Classes	# Test /Class	Ave. Len	Max Len
AG's News	4	1,900	34	211
DBpedia	14	5,000	49	953
IMDB	2	12,500	239	2,506
Yelp-Full	5	10,000	152	1,199

4.2 Experiment setup

We compare our proposed method with several popular state-of-the-art LSTM based semi-supervised text classification methods. Moreover, these competing methods are all built based on the scheme of the generative based semi-supervised learning, which can be summarized as a two-stage framework.

In the first stage, based on both labeled text data and unlabeled text data, a LSTM based language model (shown in Figure 4) is trained to model the data distribution explicitly through reconstruction of input texts. This language model is used as a "pre-training" stage for the prediction model. More specifically, the embedding matrix and the parameters of the LSTM model obtained from the first stage are considered as a starting point of the corresponding parameters in the prediction model [8].

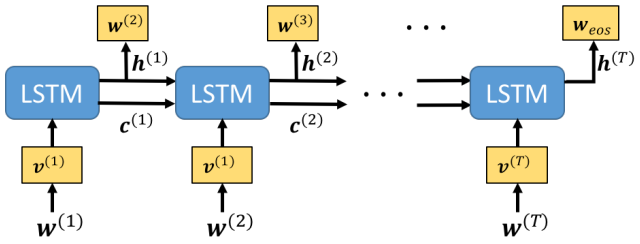


Figure 4: LSTM based language model.

In the second stage, different learning methods are used to train the text classification model, and the corresponding semi-supervised text classification algorithms are:

- Semi-supervised sequence learning (SeqSSL) [8]. The SeqSSL algorithm employs the aforementioned language model at the first stage and then trains a LSTM based classification model via a standard MLE approach.
- Training semi-supervised sequence learning with random perturbations (SeqSSL+RP). Introducing random perturbations to the input and hidden layers during training could help alleviate model over-fitting [24]; therefore, it has been used to augment the training process in the second stage of SeqSSL.
- Semi-supervised sequence learning with adversarial training (SeqSSL+AT). In [12], an adversarial training technique is used to approximate the worst case perturbations, and in [20] it has been introduced for the semi-supervised text classification.
- Semi-supervised sequence learning with virtual adversarial training (SeqSSL+VAT). Different from [12], in [21] a virtual adversarial training technique employs a Kullback-Leibler divergence (KL divergence) based regularization method to approximate the adversarial perturbation.
- Semi-supervised sequence learning with both adversarial training and virtual adversarial training (SeqSSL+ATVAT). Recently, in [20] the authors have combined aforementioned two adversarial learning methods for semi-supervised text classification.

Besides the aforementioned semi-supervised text classification methods, we also use the standard LSTM based text classification method as a baseline comparison method, which is shown in section 3.2.1. Moreover, to analyze the effect of judge network in proposed **RLANS**, a standard self-training framework with SeqSSL as base model, abbreviated as SeqSSL+SELF, is introduced as another comparison method.

In our experiment, for the sake of fairness, in a certain dataset the model hyperparameters, e.g., embedding dimension and the dimension of LSTM hidden unit, are set to be consistent in different methods. Table 2 summarizes the details of the model hyperparameters. We use the batch normalization and dropout techniques in the model implementation, and the row titled "Dropout rate" is the corresponding dropout rate of the word embedding layer. For all the methods we employ a standard single-layer LSTM as base model. Moreover, for each dataset its corresponding dimension of hidden state and cell state can be found in row titled "# LSTM hidden unit". We use the backpropagation through time (BPTT) technique [35] to speed up the training process, for each dataset the corresponding maximum BPTT is listed in the row titled "Maximun BPTT". The row titled "# hidden unit of classifier" is the dimension of the fully connected layer, i.e., the dimension of $\boldsymbol{h}_c$ in Eq.(10). All the models are trained with the mini-batch Adam Optimizer, and we set the batch size as 64 for all datasets. The row titled "# training step of LM" represents the number of training steps in the first stage of the aforementioned semi-supervised sequences classification models.

Table 2: Summary of model hyperparameters associated with each dataset.

	AG's News	Dbpedia	IMDB	Yelp-Full
Embedding dimension	256	256	256	256
Dropout rate	0.5	0.5	0.5	0.5
# LSTM layers	1	1	1	1
# LSTM hidden unit	512	1024	1024	1024
Maximun BPTT	200	400	400	400
# hidden unit of classifier	30	128	30	30
batch size	64	64	64	64
# training step of LM	20,000	100,000	100,000	100,000

4.3 Result and Analysis

In our experiments, we observe that both convergence and performance of **RLANS** are strongly affected by the pre-training of predictor and judge (refer to Line 1 and Line 3 of Algorithm 1). Note that, the pre-training stage of **RLANS** is different from the one of SeqSSL. In SeqSSL, the pre-training stage aims at training a good language model for sequence representation, while in **RLANS** the pre-training of predictor aims to find a good start point for the adversarial training steps. Moreover, the model learned from SeqSSL can be considered as a pre-trained predictor for **RLANS**. Figure 5 demonstrates the performance of **RLANS** with different pre-trained predictors, i.e., standard LSTM and SeqSSL. We can observe that a good pre-trained predictor can dramatically improve the prediction

Table 3: Performance comparison of the proposed RLANS methods and other existing related methods using accuracy (along with their standard deviations). For each dataset we vary the number of labeled training samples per class, which is shown in the column titled "# Labeled/Class", and the corresponding number of unlabeled training samples is shown in the column titled "# Unlabeled".

Dataset	# Labeled/Class	# Unabeled	LSTM	SeqSSL	SeqSSL+RP	SeqSSL+AT	SeqSSL+VAT	SeqSSL+ATVAT	SeqSSL+SELF	RLANS
AG's News	100	119,600	0.3925 (0.0046)	0.7623 (0.0009)	0.7436 (0.0009)	0.7295 (0.0007)	0.7485 (0.0011)	0.7673 (0.0010)	0.7854 (0.0235)	**0.8174 (0.0127)**
	500	118,000	0.6909 (0.0022)	0.8740 (0.0011)	0.8601 (0.0007)	0.8610 (0.0006)	0.8743 (0.0007)	0.8699 (0.0012)	0.8925 (0.0173)	**0.9152 (0.0118)**
	1000	116,000	0.7558 (0.0023)	0.8731 (0.0012)	0.8720 (0.0008)	0.8712 (0.0009)	0.8845 (0.0004)	0.8818 (0.0006)	0.9034 (0.0136)	**0.9256 (0.0109)**
Dbpedia	100	558,600	0.6871 (0.0035)	0.9611 (0.0012)	0.9761 (0.0013)	0.9666 (0.0012)	0.9778 (0.0006)	0.9684 (0.0006)	0.9813 (0.0106)	**0.9847 (0.0051)**
	500	553,000	0.9432 (0.0017)	0.9848 (0.0004)	0.9801 (0.0002)	0.9766 (0.0001)	0.9814 (0.0005)	0.9770 (0.0006)	0.9825 (0.0126)	**0.9883 (0.0036)**
	1000	546,000	0.9558 (0.0037)	0.9802 (0.0002)	0.9835 (0.0001)	0.9806 (0.0001)	**0.9918 (0.0003)**	0.9790 (0.0013)	0.9817 (0.0063)	0.9896 (0.0009)
IMDB	100	50,000	0.53875 (0.0093)	0.77355 (0.0008)	0.79495 (0.0073)	0.75645 (0.0091)	0.4314 (0.0082)	0.48635 (0.0086)	0.7974 (0.0052)	**0.8205 (0.0033)**
	500	50,000	0.6152 (0.0067)	0.87025 (0.0025)	0.8654 (0.0008)	0.8636 (0.0001)	0.65675 (0.0031)	0.7884 (0.0005)	0.8924 (0.0049)	**0.9029 (0.0023)**
	1000	50,000	0.6361 (0.0045)	0.853 (0.0014)	0.885 (0.0021)	0.8895 (0.0047)	0.81485 (0.0005)	0.85535 (0.0025)	0.8926 (0.0063)	**0.9163 (0.0032)**
Yelp-Full	100	649,500	0.2673 (0.0029)	0.4538 (0.0027)	0.4668 (0.0007)	0.4724 (0.0009)	0.4486 (0.0014)	0.4869 (0.0016)	0.4604 (0.0274)	**0.5374 (0.0265)**
	500	647,500	0.3353 (0.0023)	0.5248 (0.0003)	0.5278 (0.0005)	0.5340 (0.0004)	0.4779 (0.0031)	0.5306 (0.0005)	0.5428 (0.0236)	**0.5522 (0.0157)**
	1000	645,000	0.3979 (0.0014)	0.5489 (0.0007)	0.5463 (0.0006)	0.5677 (0.0019)	0.4952 (0.0088)	**0.5768 (0.0012)**	0.5516 (0.0335)	0.5725 (0.0163)

performance, convergence speed and stability of **RLANS**. Therefore, in the experiment we use SeqSSL as the pre-trained predictor of **RLANS**.

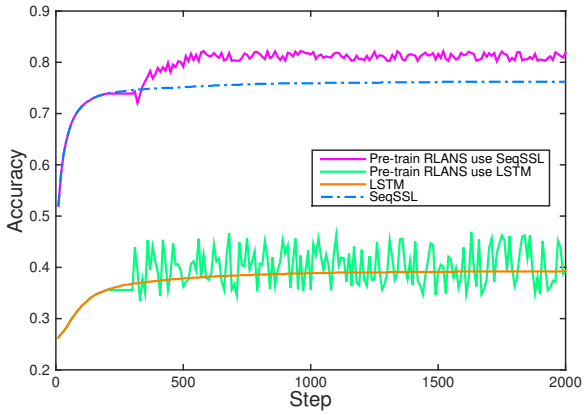


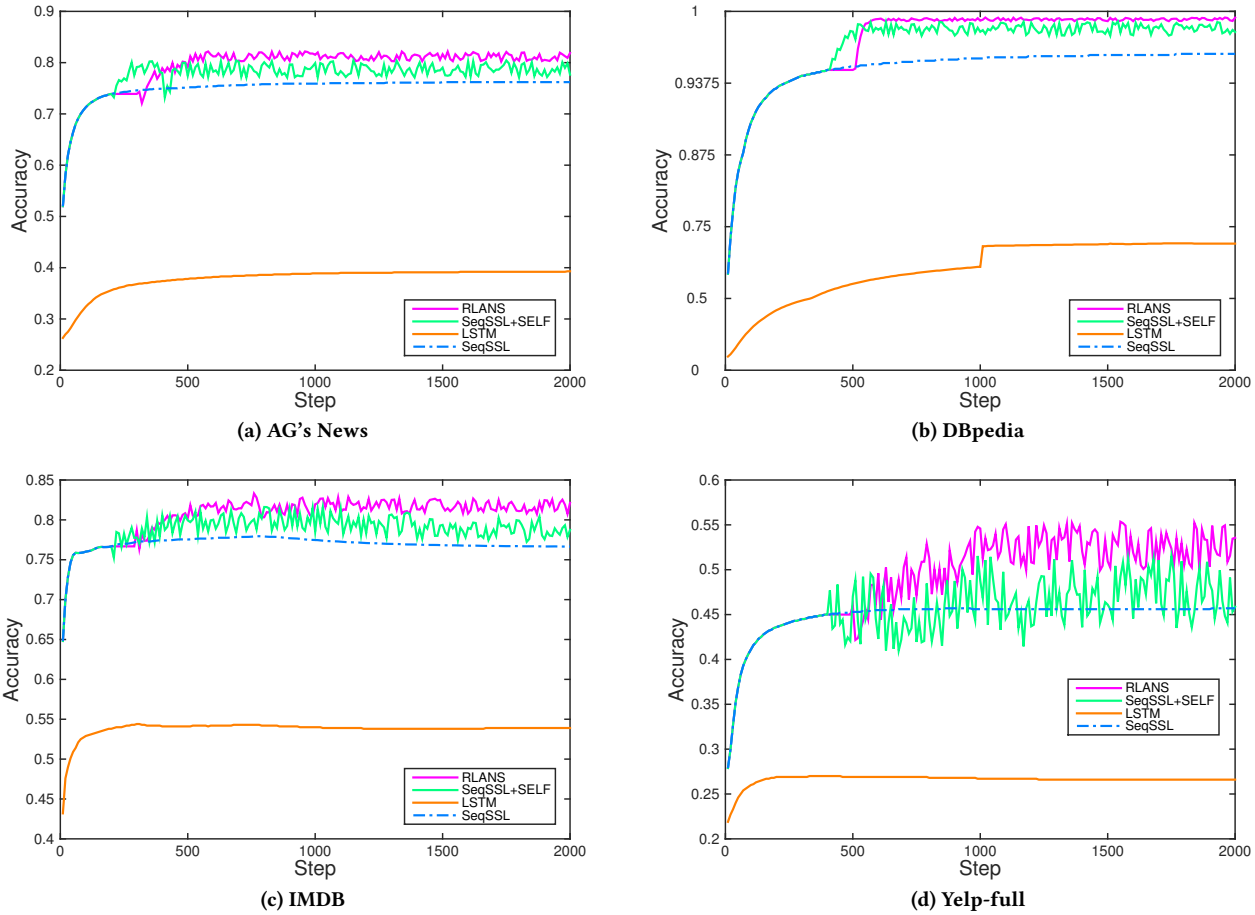
Figure 5: The performance of RLANS with different pre-trained predictors, i.e., standard LSTM and SeqSSL. The figure is drawn based on the AG's News dataset with 100 labeled instances per class.

In Table 3, we provide the performance results of accuracy on the four benchmark text datasets with various number of labeled instances in each training dataset. We report the average accuracy of five trials and the corresponding standard deviations. The best results are highlighted in bold. The results show that our proposed model outperforms the other state-of-the-art related models. Especially, the proposed **RLANS** provides significantly better prediction results compared to other semi-supervised text classification methods when the number of labeled training instances is limited.

In Figure 6, we present the convergence performance of LSTM, SeqSSL, SeqSSL+SELF and the proposed **RLANS**. Comparing with the LSTM and SeqSSL, **RLANS** is less stable, which is an inherent flaw of deep reinforcement learning. However, **RLANS** is more stable than SeqSSL+SELF, which shows the benefits of introducing judge model in **RLANS** framework. Moreover, in **RLANS**, high prediction performance is always observed along with fast convergence and good stability, it is because a good prediction model generates less incorrect predicted labels and hence alleviates bad distraction.

5 CONCLUSION

In this paper, we formulate the semi-supervised learning as a model-based reinforcement learning problem and propose a new adversarial learning framework, **RLANS**. The **RLANS** framework contains two networks: a predictor network and a judge network.



(a) AG's News (b) DBpedia

(c) IMDB (d) Yelp-full

Figure 6: Empirical convergence study of baseline LSTM, SeqSSL, SeqSSL+SELF, and RLANR w.r.t. training steps. Each subfigure is drawn based on the test result of one dataset, and we use 100 labeled samples from each class to train the aforementioned methods. Note that in RLANS, we use the first 200 training steps of SeqSSL as the pre-training of predictor model (refer to Line 1 of Algorithm 1) in the datasets AG's News and IMDB. We use first 400 training steps of SeqSSL as the pre-training of predictor model in the datasets DBpedia and Yelp-full, since SeqSSL in these two datasets converge slowly. And then we take another 100 training steps to pre-train the judge model in RLANS (refer to Line 3 of Algorithm 1) for all four datasets.

The judge network is used to dynamically evaluate the performance of the predictor network and provide a feedback as reward to dynamically guide the learning process (policy iteration) of the predictor network. The **RLANS** framework does not require data generation and hence can be easily applied to the discrete data, e.g., text. Based on this framework we propose a concrete model for semi-supervised text classification. We extensively compare the performance of the proposed algorithm with some state-of-the-art deep semi-supervised text classification algorithms using several benchmark text datasets. In the future, based on the proposed framework, we first plan to design more concrete models with more sophisticated predictor and judge networks for different types of data. We then plan to develop more advanced deep semi-supervised learning methods based on different reinforcement learning schemes, and we also plan to conduct some corresponding theoretical analysis to further improve the stability of the proposed framework.

Acknowledgments

This work was supported by the U.S. National Institutes of Health grants 1RF1AG051710-01, and National Science Foundation grants IIS-1539991.

REFERENCES

[1] Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. 2004. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 59–68.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3, Feb (2003), 1137–1155.

[3] Avrim Blum and Shuchi Chawla. 2001. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th International conference on Machine learning (ICML-01)*. 19–26.

[4] Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*. ACM, 92–100.

[5] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.

[6] Fabio Gagliardi Cozman, Ira Cohen, and M Cirelo. 2002. Unlabeled Data Can Degrade Classification Performance of Generative Classifiers.. In *Flairs conference*. 327–331.

[7] Fabio G Cozman, Ira Cohen, and Marcelo C Cirelo. 2003. Semi-supervised learning of mixture models. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 99–106.

[8] Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*. 3079–3087.

[9] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. 2017. Good semi-supervised learning that requires a bad gan. In *Advances in Neural Information Processing Systems*. 6513–6523.

[10] Akinori Fujino, Naonori Ueda, and Kazumi Saito. 2005. A hybrid generative/discriminative approach to semi-supervised classifier design. In *AAAI*. 764–769.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[14] Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International conference on Machine learning (ICML-99)*, Vol. 99. 200–209.

[15] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*. 3581–3589.

[16] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[18] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.

[19] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics, 142–150.

[20] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial Training Methods for Semi-Supervised Text Classification. *arXiv preprint arXiv:1605.07725* (2016).

[21] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2015. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677* (2015).

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[23] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine learning* 39, 2 (2000), 103–134.

[24] Ben Poole, Jascha Sohl-Dickstein, and Surya Ganguli. 2014. Analyzing noise in autoencoders and deep networks. *arXiv preprint arXiv:1406.1831* (2014).

[25] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models. (2005).

[26] Ruslan Salakhutdinov and Hugo Larochelle. 2010. Efficient learning of deep Boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 693–700.

[27] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. 2234–2242.

[28] Cicero Nogueira dos Santos, Kahini Wadhawan, and Bowen Zhou. 2017. Learning Loss Functions for Semi-supervised Learning via Discriminative Adversarial Networks. *arXiv preprint arXiv:1707.02198* (2017).

[29] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. 2005. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of ICML workshop on learning with multiple views*, Vol. 2005. Citeseer, 74–79.

[30] Jost Tobias Springenberg. 2015. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390* (2015).

[31] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*.

[32] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[33] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.

[34] Gerald Tesauro. 1992. Practical issues in temporal difference learning. In *Advances in neural information processing systems*. 259–266.

[35] Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560.

[36] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[37] Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational Autoencoder for Semi-Supervised Text Classification.. In *AAAI*. 3358–3364.

[38] David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 189–196.

[39] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.. In *AAAI*. 2852–2858.

[40] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. 649–657.

[41] Xiaojin Zhu. 2006. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison* 2, 3 (2006), 4.

[42] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 912–919.