On the Practicality of Integrity Attacks on Document-Level Sentiment Analysis

Andrew Newell, Rahul Potharaju, Luojie Xiang, and Cristina Nita-Rotaru
Dept. of Computer Science, Purdue University
West Lafayette, IN, USA
newella@cs.purdue.edu, rpothara@cs.purdue.edu, xiang7@cs.purdue.edu,
crisn@cs.purdue.edu

ABSTRACT

Sentiment analysis plays an important role in the way companies, organizations, or political campaigns are run, making it an attractive target for attacks. In *integrity attacks* an attacker influences the data used to train the sentiment analysis classification model in order to decrease its accuracy. Previous work did not consider practical constraints dictated by the characteristics of data generated by a sentiment analysis application and relied on synthetic or preprocessed datasets inspired by spam, intrusion detection, or handwritten digit recognition. We identify and demonstrate integrity attacks against document-level sentiment analysis that take into account such practical constraints. Our attacks, while inspired by existing work, require novel improvements to function in a realistic environment where a victim performs typical steps such as data cleaning, labeling, and feature extraction prior to training the classification model. We demonstrate the effectiveness of the attacks on three datasets – two Twitter datasets and an Android dataset.

Categories and Subject Descriptors

I.5.2 [**Computer Methodologies**]: Pattern Recognition—*Design Methodology*

Keywords

Machine Learning; Security; Sentiment Analysis

1. INTRODUCTION

Sentiment analysis has emerged as one of the most popular data analytics applications driving targeted advertising [1], product recommendations [2], sentiment extraction [3–5], and opinion mining [6]. According to a recent article in Communications of the ACM [7] "over 7,000 articles have been written on the topic, hundreds of startups are developing sentiment analysis solutions and major statistical packages such as SAS and SPSS include dedicated sentiment analysis modules". The most common form is *document-level sentiment analysis* which assumes that the entire document conveys a particular sentiment about one main object.

AISec'14, November 7, 2014, Scottsdale, Arizona, USA.

http://dx.doi.org/10.1145/2666652.2666661.

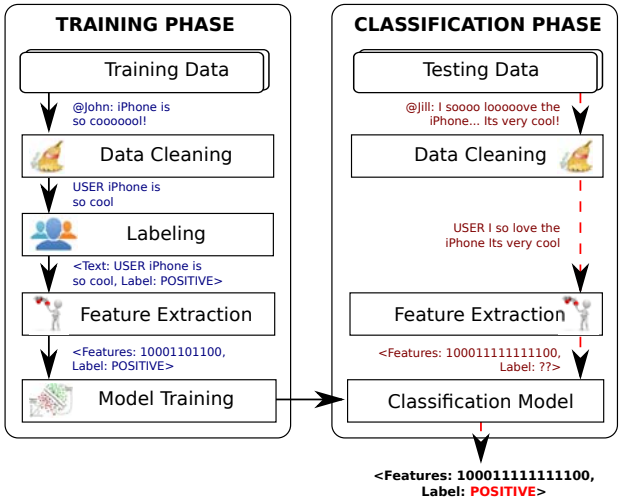


Figure 1: A typical learning system

At the core of a sentiment analysis application lies a learning system as the one shown in Fig. 1. The main goal of the system is to learn a classification model based on a set of documents referred to as *training dataset*. Prior to training the classifier each document in the training dataset must be: (1) labeled with its true class, and (2) mapped to a numerical vector through *feature* extraction [8]. By training on the feature vectors, the learning system associates each class with particular features. A higher accuracy classifier can be obtained with less training data by removing features of the data that are unimportant for classification, also known as *data cleaning*. The characteristics of the training dataset as well as the data cleaning, labeling and feature extraction procedures influence the accuracy of the obtained classification model.

Numerous techniques have been designed to deal with the noise and errors present in benign environments for sentiment analysis. However, the important role played by sentiment analysis systems provides incentives for attackers to influence them. For example, by influencing the sentiment analysis an adversary can promote low quality or untrustworthy content for online advertising or can promote his products/produce incorrect recommendations for competing products for review applications. As learning systems were not designed to work in adversarial environments, an attacker can try to compromise the confidentiality or integrity of a victim's learning system. Most of the work on attacks

against machine learning focused on integrity attacks as they provide more incentives for the attackers. Confidentiality attacks enable an attacker *to learn the classification model* by querying the classifier with crafted datapoints [9, 10]. Integrity attacks allow an attacker *to decrease the accuracy of the classification model* by modifying or inserting data at some step prior to model training. Note that integrity attacks are different from evasion attacks. An evasion attack knows a learned model and aims to create datapoints which are classified incorrectly by that model. In contrast. integrity attacks insert malicious datapoints into the training data influencing the classifier to have poor accuracy.

Sentiment analysis systems are extremely vulnerable to integrity attacks as accurate models rely on training data from public sources. These public sources (e.g., Twitter) offer a high volume of rich information. However, as the source of training data is public, an attacker may create malicious data (e.g., Tweets) that forces the learning system to learn incorrect patterns. Traditional security techniques such as cryptographic data integrity cannot prevent such an attack since an attacker is not modifying an existing training set. The attacker is tarnishing the public data before it is collected by a learning system for training. A learning system cannot easily distinguish the data created by an attacker.

Previous work on integrity attacks focused on either synthetic datasets that do not capture the characteristics of real social media data, or on different applications such as spam, intrusion-detection, or hand-written digit recognition, essentially different in the characteristics of the dataset they operate on, the cleaning, labeling, and feature vector extraction procedures used. Specifically, earlier work [11–13] focuses on integrity attacks against synthetic data from fixed distributions. By relying on the statistical properties of the considered distributions for the datasets, such works identify theoretical bounds on how much of the most erroneous data is necessary to harm classifier accuracy. However, such datasets are not from real-world applications and thus it is not clear if the identified attacks will be effective in practice against real sentiment analysis datasets. Recent works [14–19] consider real-world datasets with attacks that alter training data for spam, intrusion detection, and handwritten digit recognition. None of these applications generate data that match the characteristics of data generated by social media based sentiment analysis applications, thus their techniques can not be applied for sentiment analysis.

All of the existing work on integrity attacks against learning systems assumes modification or insertion of data at the level of numerical vectors, i.e. altering the feature vector (see Fig. 1). However, in a realistic scenario, an attacker can influence the *data collected for training and not the labeled numerical representation of this data which is a result of extraction, cleaning, and labeling.* As a result, the adversary may not know the exact details of each step taken by a learning system, and thus, cannot understand the exact numerical vectors that result from training data or how constructed malicious training data is converted into numerical vectors. Moreover, the space of possible numerical values becomes limited since certain values in the numerical vectors cannot take on arbitrary real numbers, and thus attacks that assume arbitrary values will not be effective against such systems. For example, many sentiment analysis systems use *bag-of-words* as a feature vector mechanism, where features are booleans (0 or 1) indicating the existence of particular words.

In this paper, we study integrity attacks against sentiment analysis. We apply attacks that are motivated by existing work on adversarial machine learning, but we alter these attacks to consider the real-world scenario in which such an application would be deployed. Specifically, we assume that the victim implements a realistic learning system with all the steps described in Fig. 1, consider realistic adversarial models and construct attacks that meet the constraints resulted from the data processing steps. Additionally, we consider a weaker but more realistic attacker that has partial knowledge of the victim's learning system and study the differences in crafted attack datapoints as well as their impact on a learning system. We provide a mathematical formulation to find impactful attack data points. Our attack formulation finds a feasible data point optimizing an objective which is the opposite objective of the correct model. For instance, an attacker inserts positively labeled datapoints with words of heavy negative sentiment.To our knowledge, our work is the first to demonstrate adversarial machine learning attacks in such a practical scenario. Recent work has also considered practical scenarios, but focused on evasion attacks [20, 21].

Our evaluations consider three realistic datasets (two Twitter datasets and an Android dataset), two well-known machine learning algorithms (SVM and Naive Bayes), and the bagging ensemble technique. We highlight the following findings from our evaluation:

- Our attack with the strongest adversarial assumptions is able to degrade accuracy of a learning system from 86% to 50% by adding fewer than 2% maliciously crafted datapoints to a training dataset.
- We gained insights into attack effectiveness against different aspects of a learning system: more complex learning algorithms are more susceptible to attack, larger training datasets require more attack points to corrupt accuracy, using data that has more relaxed constraints is more susceptible to attack, and the learning algorithm ensemble technique of bagging slightly reduces attack effectiveness.
- We also assume scenarios where an attacker may have partial knowledge about certain preprocessing steps of a victim's classifier. Considering such a scenario brings insight into the value for a victim to keep information of their system secret from possible attackers. The attack effectiveness varied from degrading accuracy from 86% to 50% (same as full knowledge) to degrading only to 68%. Thus, attacks that craft datapoints can still be effective even in scenarios of partial knowledge.

We organize our paper as follows. Section 2 describes the general machine learning model we aim to attack. Section 3 describes attacks against machine learning models. Section 4 describes our experimental methodology for evaluating the performance of these attacks and Section 5 presents the results of our evaluation. Section 6 presents results when the attacker has partial information about the steps used by the learning system. Section 7 lists related work, and Section 8 concludes our work.

2. MACHINE LEARNING FOR SENTIMENT ANALYSIS

We describe the learning system for document-level sentiment analysis of social media assumed in this work. We consider a system which could easily and practically be deployed today as a sentiment analysis system.

2.1 Overview

The design of a machine learning system involves making several design choices taking into account the nature of the data to be classified and the classification task:

- Select the *training data* used to train the machine learning algorithm performing the classifying task.
- Select the *data cleaning* mechanisms which remove useless characteristics for classification.
- Select a *labeling technique* for the training data to indicate what is the true class for each document.
- Select the *feature vector* that transforms each document in the training data into a numerical vector for the machine learning algorithm.
- Select a *classification algorithm* that learns a function from the processed numerical vectors.

Training and testing data: Learning algorithms are sensitive to the size of the training set. A training set (1) must be sufficiently large to represent a majority of the hypothesis space, and (2) must have sufficient datapoints from each class as an imbalanced training set can pose difficulties in learning. In addition, the training and testing data should possess the property of *data stationarity* i.e., they must be drawn from the same distribution to ensure that correlations found in the former are also true in the latter.

Preprocessing: Raw data often contains elements that are difficult to process such as special characters, web links, or text in different languages. Many algorithms are sensitive to such elements. Thus, data used for training needs to be preprocessed to ensure the best representative datapoints are used. For example, textual data is filtered by language to simplify learning language-specific textual patterns.

Labeling: The accuracy of labeling the training set has a direct impact on the overall accuracy of the learned hypothesis function f. To label a sufficiently large training set, noisy-labeling techniques can be employed for automatic labeling as long as the noise in the labeling is low.

Feature extraction: The core of a learning model is the extraction of features [8] from a preprocessed data instance that is representative for the data. Each data instance is mapped to a vector of numerical values which is a datapoint. The goal is to capture the important features of a data instance as a vector of numbers such that a learning algorithm can correctly associate datapoints with their corresponding class. For example, textual data can be transformed via a *bag-of-words* approach which results in a vector of 0's and 1's indicating which words are absent or present respectively.

Learning algorithm: Several learning approaches exist, each algorithm determines the space of possible discriminant functions h that can be learned. Learning algorithms differ in how they correlate example features with class labels, so the best algorithm may depend on the dataset.

Below we describe the design for the sentiment analysis system considered in this work. The design is largely based on the one used in [6], and is using simple techniques that were shown to be effective for document-level sentiment analysis.

2.2 Data Cleaning

While, there are numerous data cleaning steps, we selected the ones we describe below as they have been shown previously to work well in previous work [22].

Data cleaning removes features of the data that are unimportant for classification allowing higher accuracy classifiers to be created with less training data. As an example, consider the word "happy" in the training data. Such a word is typically a great indication of positive sentiment, and it is great for any machine learning algorithm to find such a correlation and use this correlation in a classifier. However, this word may show up in the training data as "Happy", "HAPPY", or "happy". Without data cleaning, the training data must have enough occurrences of each of these forms to learn such a correlation to understand each form of "happy" in a testing set can indicate positive sentiment. With data cleaning, such a correlation is learned much easier as each form of "happy" is mapped to the same feature to help build evidence of this correlation. Similar logic can be applied to the other types of data cleaning which a practical learning system for sentiment analysis may employ.

URL REMOVAL. Each URL is replaced with "url". A specific URL may indicate a negative or positive sentiment due to the time in which the training data was created, but such sentiment of a URL may not always be true for futuristic testing sets.

USERNAME REMOVAL. Each username, (e.g., "@mrsmith") is replaced with "username". Usernames are removed for the same reasoning that URLs are removed.

REPEATED LETTER REMOVAL. E.g., "happppppppy" is replaced with "happy". Limiting the number of repeated letters will help map the misspellings (with repeated letters) to the actual English.

CASE NORMALIZATION. Case normalization is performed for similar reasoning as repeated letter removal since it ensures that two different types of case usage are mapped to the same word. E.g., "Happy ToWn" is replaced with "happy town".

ENGLISH TEXT. The testing sets we use are in English text, so we did not want to consider other languages. We employed standard language detection tools to remove non-English text from our training data.

2.3 Labeling

The labeling process assigns a class to each training datapoint. We consider binary sentiment classification, i.e. the class for each datapoint is either positive or negative (sentiment). As manually labeling sufficient datapoints by a human experts is tedious and error-prone, we use automated labeling. Automated labeling uses the metadata or specific features of the data to provide a label.

Twitter dataset: For our Twitter datasets, similarly to [6] we leverage datapoints that have emoticons and use the sentiment of the emoticon to label, i.e., presence of a happy smiley indicates a positive tweet and vice versa. We strip all emoticons from the input tweets because otherwise, their presence may negatively impact the accuracies of our classifiers. This step causes the classifier to learn from the other features present in the tweet. Therefore, even if the test data contains an emoticon, it does not influence the

classifier because emoticon, as a feature was removed from the training data.

Android dataset: For our Android dataset, we leverage the rating supplied by the user to indicate the sentiment of the comment. We take comments with a rating score higher than 3 as having a positive sentiment and less than 3 as having a negative sentiment.

2.4 Feature Extraction

Feature extraction transforms a document into a numerical vector that is convenient for a learning system. We assume our system uses a common approach denoted *bag-of-words* wherein a mapping from every word in the training data to an element of a numerical vector is constructed. Note that the number of elements in a numerical vector can get quite large (≈50K in our learning system). For each datapoint, a numerical vector is created where an element has a value of 1 if the corresponding word exists in the datapoint while it has a value of 0 otherwise. Thus, the numerical vectors have element values of 0s and 1s while the vector is typically sparse (few 1s and many 0s).

2.5 Model Training

We consider model training for binary classification, i.e., a classifier h is learned which maps datapoints into two classes. Specifically, given a training set of datapoints of the form $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, the learning algorithm outputs a function $y = h(\mathbf{x})$. Each vector $\mathbf{x}_i = \langle x_1, x_2, ..., x_n \rangle$ consists of values called *features*[1] and each value y_i is a discrete value $1, ..., M$ called a *class label*. We focus on the case where $M = 2$ which represents a binary classifier.

The goal of a learning system is to achieve a high accuracy. For a learned classifier h we define *accuracy* as the proportion of datapoints such that h correctly maps a datapoint to its correct class. Specifically, given a testing data $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_t, y_t)\}$, accuracy is the proportion of these points where $h(\mathbf{x}_i) = y_i$. Below we describe the classifiers we consider in this work.

Naives Bayes: The general Bayes classifier $h(\mathbf{x})$, uses the class posterior probabilities given a datapoint, i.e., $f_i^*(\mathbf{x}) = P(C = i|X = \mathbf{x})$. Applying Bayes rule gives $P(C = i|X = \mathbf{x}) = \frac{P(X=\mathbf{x}|C=i)P(C=i)}{P(X=\mathbf{x})}$, where $P(X = \mathbf{x})$ is identical for all classes, and therefore can be ignored. This gives us the following Bayes classifier:

$$h(\mathbf{x}) = \text{argmax}_i P(X = \mathbf{x}|C = i)P(C = i) \quad (1)$$

finds the maximum *a posteriori probability hypothesis* given a datapoint $\mathbf{x}$. Because, direct estimation of $P(X = \mathbf{x}|C = i)$ from a training set is difficult when the feature space is high-dimensional, approximations are commonly used. When features are assumed to be independent, we refer to the resulting classifier as the *Naive* Bayes classifier which substitutes into Equation 1 the following:

$$P(X = \mathbf{x}|C = i) = \prod_{j=1}^{n} P(X_j = x_j|C = i) \quad (2)$$

Support Vector Machines: We describe a binary Support Vector Machine (SVM). The goal of an SVM is to search a hyperplane in the reproducing kernel hilbert space [23] that maximizes the margin between the two classes of datapoints with the smallest training error [24]. This problem is typically formulated as the following quadratic optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq \xi_i, \xi_i \geq 0 \end{aligned} \quad (3)$$

the variables are a weight vector $\mathbf{w}$, a hyperplane offset b, and slack variables ξ_i. The separating hyperplane is orthogonal to $\mathbf{w}$ and b offset from the origin. There is a single slack variable ξ_i for each training datapoint $(\mathbf{x}_i, y_i)$. A slack variable is 0 if the training datapoint is outside the margin on the correct side of the hyperplane. Alternatively, a slack variable's value increases based on the degree to which the point is misclassified, that is, the further the point is away from the hyperplane. Thus, loss is captured by a summation over these slack variables, $C\sum_{i=1}^{n}\xi_i$, where C is a constant that controls loss versus generality of the model. The regularizer term to ensure generality of the model is to maximize the margin of the hyperplane. The margin is $\frac{2}{\mathbf{w}^T\mathbf{w}}$, so minimizing the inverse in the objective is equivalent to maximizing the margin.

The classifier for SVM is the following:

$$h(\mathbf{x}) = \begin{cases} +1 & \mathbf{w}^T\mathbf{x} + b > 0 \\ -1 & else \end{cases} \quad (4)$$

where $\mathbf{w}$ and b are learned from the optimization problem above over the training data.

Ensemble: For NB, SVM, or other model training algorithms, additional ensemble techniques can be used to improve accuracy. An ensemble aggregates the results of multiple classifiers from different model trainings to increase accuracy. The ensemble of multiple classifiers is a classifier itself. In our work, we study an ensemble technique called bagging which improves the generality of the final, learned classifier. Given a set of training data S, we create k random subsets of this data $s_1, s_2, ..., s_k$, s.t., $\forall i, s_i \subset S$. Then, we can train k classifiers $h_1(\mathbf{x}), h_2(\mathbf{x}), ..., h_k(\mathbf{x})$, and the overall ensemble classifier is a majority vote of these k classifiers. Such a technique is used in practice to reduce the effect of outliers in training data.

3. INTEGRITY ATTACKS

In this section, we describe the attack model we assume, the general attacks against machine learning, and concrete ways to generate such attacks against real-world machine learning systems for sentiment analysis.

3.1 Attacker Model

The popularity and importance of sentiment analysis provides incentives for attackers to attempt to influence it. An attacker can try to compromise the confidentiality or integrity of a victim's learning system. Confidentiality attacks enable an attacker *to learn the classification model* by querying the classifier with crafted datapoints. Integrity attacks allow an attacker *to decrease the accuracy of the classification model* by modifying or inserting data at some step prior to model training. We focus on integrity attacks as they provide more incentives for the attackers.

For a realistic scenario, we assume an attacker never modifies training data already collected by a learning system.

[1]As a comment on our notation throughout this work, $\mathbf{x}_i$ refers to the ith datapoint while x_i refers the to the ith feature of a datapoint.

Such attacks can be addressed by more traditional security techniques such as firewalls, secure operating systems, or cryptographic signatures. Instead, we assume an attacker influences training data by inserting data into a data set prior to its collection by a learning system. Such influence is possible as data sets are typically generated based on content generated from online user profiles. An attacker can create numerous profiles and generate content from each profile. As shown in previous work [25] attackers can compromise Twitter accounts and use them for malicious purposes. For example, previous work [25] has studied spam based on Twitter accounts. In our work, we assume that the compromised Twitter accounts are used to generate content to influence the learning system of a sentiment analysis classifier.

We assume that the attacker transforms the training dataset by injecting a certain number of maliciously selected or crafted datapoints. Every data point added for training affects the model learned, and the maliciously created data points will strive to degrade the performance of the classifier. Such an attack subverts data stationarity as the training dataset becomes dissimilar to a testing set which the victim aims to have high accuracy on. Unlike previous work that assumes that such injection is performed on the pre-processed data or synthetic datasets, we assume that the injection happens before data is pre-processed and feature vectors are computed. This model captures realistic scenarios of sentiment analysis application deployment where the attacker does not have direct control over the feature vectors. Furthermore, we also assume scenarios where an attacker may have partial knowledge about certain preprocessing steps of a victim's classifier. Considering such a scenario brings insight into the value for a victim to keep information of their system secret from possible attackers.

Note that integrity attacks are different from evasion attacks. An evasion attack knows a learned model and aims to create datapoints which are classified incorrectly by that model. In contrast. integrity attacks insert malicious datapoints into the training data influencing the classifier to have poor accuracy.

We assume that the adversary has knowledge of various components of a victim's classifier and inserts datapoints into the training set based on this knowledge. Components that can be known by an adversary include: training data set, data cleaning, labeling, feature extraction, and learning algorithm. We define the following types of adversaries, ordered from the weakest to the strongest based on their capabilities.

Adversary-0. This is an adversary that can insert a limited amount of datapoints in the training set. Such datapoints are subject to domain-restrictions (e.g., respect the character limit). He also knows the labels associated with the training data used by the victim but cannot modify them.

Adversary-I. This is an adversary that has all the capabilities of *Adversary-0* and in addition controls the labeling of datapoints inserted into the training set. For example, the attacker knows the noisy-labeling technique.

Adversary-II. This is an adversary that has all the capabilities of *Adversary-I* and in addition knows the algorithms of the learning system, knows all training data, and can alter the contents of datapoints inserted into the training set.

Adversary-IIP. This is an adversary that has all the capabilities of *Adversary-II* except he only has a partial knowledge of the data cleaning (for instance, has knowledge of user name removal but not repeated letter removal) performed by the victim's learning system.

Adversary-III. This is an adversary that has all the capabilities of *Adversary-II* and in addition knows a limited testing set which the adversary aims to influence classification on. These assumptions are the greatest of any adversary.

Adversary-IIIP. This is an adversary that has all the capabilities of *Adversary-III* except he only has a partial knowledge of the data cleaning performed by the victim's learning system.

3.2 Attack Description

All the attacks have the ultimate goal to inject datapoints in order to influence the accuracy of the classifier. The difference between the attacks is how datapoints are selected or crafted (a summary is shown in Table 1). For a full attack, these steps would be repeated several times to increase the influence the attack has on the victim's model.

We describe four attacks conducted by attackers with increased strength: REPEAT, LABEL-FLIP, POISON-1, and POISON-2. The first two attacks consist of inserting an existing or modified datapoint, while the next two attacks consist of inserted newly crafted datapoints.

REPEAT: This is the simplest attack created by an attacker in class *Adversary-0*. The attack works as follows: the attacker selects a random datapoint of a preselected class from the training data, creates a copy of the datapoint, and then inserts the copy of datapoint into the training set. The attacker exploits a model's sensitivity to an imbalanced training set. The attacker only needs access to a set of datapoints from a given class to replicate and insert many datapoints into the training set.

LABEL-FLIP: This attack is conducted by an adversary of type *Adversary-I*. The attack selects a random datapoint, creates a copy of the datapoint, changes the label of the copied datapoint, and then inserts the copy with its changed label into the training set. By switching the label, the model uses mislabeled points for training and with enough mislabeled points starts to incorrectly associate certain features with the classes they should actually represent.

POISON-1: This attack requires significantly more information about the victim's learning system and can be conducted by an adversary of type *Adversary-II*. A datapoint is crafted by selecting a set of features and class label which are the least fitting to the victim's learned model. The creation requires the learned model which is a result of knowing the training dataset and learning system of the victim which is why an Adversary-II is required. Additionally, the creation requires the solving of an optimization problem which depends on the feature extraction and machine learning algorithm which we detail in Section 3.3. When a datapoint is crafted, it is added back into the training set to determine which features to use in the next crafted datapoint. This attack can also be performed by an adversary of type *Adversary-IIP*, the only difference is that the attack may have less impact due to partial knowledge of the victim's learning system.

POISON-2: This attack is similar to POISON-1 while requiring additional knowledge of a testing set and is conducted by an attacker from class *Adversary-III*. The datapoints are crafted like in POISON-1, but the features are

Table 1: Summary of various integrity attacks on learning systems

Name	Exploit	Data insertion technique (Target: Victim's training data)	Adversary
REPEAT	Imbalance	Replicate datapoint	Adversary-0
LABEL-FLIP	Label noise	Change label class	Adversary-I
POISON-1	Outliers	Craft datapoint that least fits the victim's learning model	Adversary-II/IIP
POISON-2	Outliers	Craft datapoint that least fits the victim's learning model based on victim's testing data	Adversary-III/IIIP

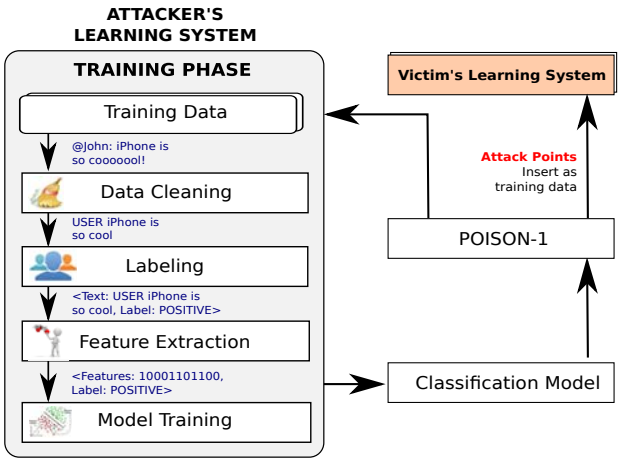


Figure 2: Poisoning attack scenario against learning systems.

weighted by their frequency in the testing set. The weighting ensures features are preferred if they occur more often in a testing set. Focusing on those features in the testing set allows higher accuracy degradation with fewer crafted datapoints. The crafted datapoints are still outliers like with POISON-1, but their features are focused on the testing set features. This attack can also be performed by an adversary of type *Adversary-IIIP*, the only difference is that the attack may have less impact due to partial knowledge of the victim's learning system.

3.3 Poisoning Real Learning Systems

The POISON-1 and POISON-2 attacks craft points which oppose the model learned by the victim's learning system (see Figure 2). We define optimization formulations which capture how to craft points against the SVM and NB learning systems. Our formulations consider the bag-of-words feature extraction which is typical for sentiment analysis and other machine learning techniques using textual data.

Previous research efforts [10, 14, 26] assume that the features present in the samples are determined randomly or ignore their distributions altogether. However, this is rarely the case in real-world datasets as there are several practical application-specific constraints such as bag-of-words feature extraction and character limit that prevent an attacker from launching attacks with arbitrary vectors. For instance, consider the case of sentiment analysis using Twitter — an adversary must use 0,1 element values and obey the constraint of 140 characters per datapoint when attempting to poison the data stream. This scenario constrains an attacker to craft only sparse binary vectors for their attack. Below, we propose techniques with these constraints to attack an SVM and an NB classifier. These same techniques are used for scenarios of partial knowledge.

SVM poisoning with sparse binary vectors. We describe techniques for POISON-1 and POISON-2 against an SVM classifier on textual sentiment analysis data. Equation 3 describes the optimization problem that minimizes loss while maximizing the margin. When a training datapoint $(\mathbf{x}_i, y_i)$ is fit well by the model, then the loss for that datapoint is captured by the variable ξ_i, which is equivalent to $1 - y_i(\mathbf{w}^T\mathbf{x} + b)$. In contrary, the goal of the attacker is to create a training datapoint $(\mathbf{x}, y)$ which maximizes loss of the SVM while obeying the constraints for a sparse binary vector:

$$\begin{aligned} \text{maximize} \quad & 1 - y(\mathbf{w}^T\mathbf{x} + b) \\ \text{subject to} \quad & x_i \in \{0, 1\}, \forall i \\ & \mathbf{c}^T\mathbf{x} - 1 \leq L, y \in \{-1, +1\} \end{aligned} \tag{5}$$

$\mathbf{c}$ is a vector containing the word length (including a space separating character) of each feature, and L is the maximum number of characters in the document. In the Twitter dataset, the minimal size to append an emoticon is 3 characters (2 characters for ":)" or ":(" along with a space), and the maximal Tweet size is 140 characters, so $L = 137$. In the Android comments dataset, the maximal size for a comment is 1200 characters, so $L = 1200$. Solving Equation 5 involves solving two knapsack problems: one sets $y = -1$ and the other sets $y = +1$. The largest value from these two knapsacks is chosen as the final result. Although optimally solving the general knapsack is NP-Hard there is a $(1+\epsilon)$-approximation algorithm [27] which we utilize that runs in time polynomial in $\frac{1}{\epsilon}$.

POISON-2 improves its impact over POISON-1 as we leverage a known test set. An additional step is included which adjusts the likelihood to select each feature based on the rate of occurrence of that feature in the test set. Let $\mathbf{t}$ be a vector with an element t_i for each feature in the testing set. The t_i value is the sum of class label values of those testing set points that contain the ith feature. A value t_i has a large positive value if it indicates strong sentiment in the testing set and vice versa for negative values. Let the operator $\odot$ be a pairwise multiplication of vector elements such that $\mathbf{w} \odot \mathbf{t} = \langle w_1 * t_1, w_2 * t_2, ... \rangle$. The POISON-2 attack is then performed as follows:

$$\begin{aligned} \text{maximize} \quad & 1 - y((\mathbf{w} \odot \mathbf{t})^T\mathbf{x} + b) \\ \text{subject to} \quad & x_i \in \{0, 1\}, \forall i \\ & \mathbf{c}^T\mathbf{x} - 1 \leq L, y \in \{-1, +1\} \end{aligned} \tag{6}$$

NB poisoning with sparse binary vectors. We now show a POISON-1[2] attack on NB. Equation 1 shows the classification for NB which selects the class label i which maximizes posteriori probability $P(X = \mathbf{x}|C = i)$. We can find a datapoint which opposes the model by finding a datapoint which maximizes the ratio of the correct and incorrect posteriori probabilities:

[2]There is a POISON-2 for NB that is similar to the version for SVM, but we omit those details in this work.

$$\begin{aligned} \text{maximize} \quad & \left(\frac{P(C=-1|X=\mathbf{x})}{P(C=+1|X=\mathbf{x})}\right)^y \\ \text{subject to} \quad & x_i \in \{0,1\}, \forall i \\ & \mathbf{c}^T\mathbf{x} - 1 \leq L, y \in \{-1,+1\} \end{aligned} \tag{7}$$

In this form it is unintuitive how to solve, but we will show this problem is a knapsack problem as well. We first transform the objective of the optimization problem by letting $P(X_i = 1|C = +1) = p_i^+$ and $P(X_i = 1|C = -1) = p_i^-$. Then, we substitute those terms that do not rely on the variable x, the substituted constant a is computed once for a model and remains the same for any value of the variable x:

$$\begin{aligned} & \left(\frac{P(C=-1|X=\mathbf{x})}{P(C=+1|X=\mathbf{x})}\right)^y \\ = & \left(\frac{P(C=-1)\prod_{i=1}^n(1-p_i^-)\prod_{\forall i:x_i=1}\frac{p_i^-}{1-p_i^-}}{P(C=+1)\prod_{i=1}^n(1-p_i^+)\prod_{\forall i:x_i=1}\frac{p_i^+}{1-p_i^+}}\right)^y \\ = & \left(a\prod_{\forall i:x_i=1}\frac{p_i^- - p_i^-p_i^+}{p_i^+ - p_i^-p_i^+}\right)^y \end{aligned} \tag{8}$$

We then take the logarithm of the objective as maximizing the logarithm also maximizes the original objective. By maximizing the logarithm of the objective from Equation 7 we are left with a knapsack problem similar to the SVM case:

$$\begin{aligned} \text{maximize} \quad & y\log(a) + y\sum_{\forall i:x_i=1}\log\left(\frac{p_i^- - p_i^-p_i^+}{p_i^+ - p_i^-p_i^+}\right) \\ \text{subject to} \quad & x_i \in \{0,1\}, \forall i \\ & \mathbf{c}^T\mathbf{x} - 1 \leq L, y \in \{-1,+1\} \end{aligned} \tag{9}$$

4. METHODOLOGY

We use three different datasets for our experiments, two Twitter datasets and an Android dataset.

Collecting the datasets. The first two datasets are tweets from Twitter users spanning two different time periods: "Egyptian", consisting of 16 million tweets collected between January 23rd and February 8th, 2011 capturing the time period of the Egyption revolution, and "Elections", consisting of 11 million tweets collected between November 5 and November 12, 2012 capturing the time period of the US presidential election. The first dataset was collected and made available by NIST [28]. We collected the second dataset using Twitter's streaming API (with IRB approval).

The third dataset, "Android", pertains to comments made on a popular smartphone market, Google Play, a marketplace for Android applications. We have developed an in-house crawler that we used to take a snapshot of the entire Google Play store on Nov 17, 2012. Our snapshot consists of 150K applications that have 625K user comments accompanying them.

Processing the datasets. To obtain English documents from our data we perform standard language filtering techniques [29] trained by an existing language corpus provided by the Natural Language Toolkit [30]. On the resulting English documents, we apply the standard data cleaning steps described in Section 2.2. The Twitter datasets are labeled by using only Tweets with emoticons. The Android dataset is labeled by their comment ratings which range from 1-5, and we let ratings of 1 and 2 represent negative sentiment while 4 and 5 represent positive sentiment (we do not use comments with ratings of 3).

Table 2: Experimental Setup

	Default value	Experimented values
Training dataset	Egyptian	Egyptian, Election, and Android (see Section 4)
Training dataset size	30K	15K, 30K, and 45K
Machine learning algorithm	SVM	SVM and NB
Ensemble	No ensemble	No ensemble and bagging ensemble

Selecting testing datasets. For the testing sets, we manually label a set of Tweets and Android comments. We use the Tweets testing set to measure accuracy of a learning system trained with Tweets, and the same for Android comments. Each testing set consists of 200 positive and 200 negative datapoints.

Selecting size of the training datasets. To understand how many datapoints were sufficient to train an accurate model we varied the size of the training dataset and observed accuracy. For Twitter, we found that a training dataset of 12K datapoints resulted in 83% accuracy while growing slightly to 86% with 30K datapoints, and the accuracy did not increase by adding more datapoints. We observed similar accuracy from the Android dataset as well. So, we use 30K datapoints (15K positive and 15K negative) as our default training dataset size. For the Election dataset, due to a limited number of tweets with emoticons, we use just 12K datapoints (6K positive and 6K negative). For the Android dataset, we chose to remain consistent with the Egyptian dataset by randomly selecting 30K positive and 30K negative datapoints as the training dataset.

5. EVALUATION

We study the effectiveness of the attacks described in Section 3 by evaluating sensitivity to different classifiers, training datasets, and training dataset sizes. We plot the accuracy of the learning system as a function of the number of attack datapoints inserted into the victim's training dataset. Table 2 shows the default values and the different parameters we vary during the experiments.

We experiment with an ensemble technique. For this, the training dataset is randomly sampled 10 times to create 10 training datasets. These new datasets are half the size of the original training dataset, and they are not disjoint from each other. A model is learned from each dataset. For classification, a datapoint is classified by each model, and the final classification is a majority vote from each of the 10 models.

5.1 Attack comparison

To evaluate the impact of the attacks on the accuracy of the classifier, we insert a varying number of attack datapoints (depicting the attacker's actions) into the victim's training set. Figure 3(a) shows the accuracy for each attack using the default dataset (Egyptian) and the default learning system (SVM). The order of increasing effectiveness is as expected since each attack requires more information than the previous attack: REPEAT, LABEL-FLIP, POISON-1, and POISON-2. To better illustrate the impact of POISON-2,

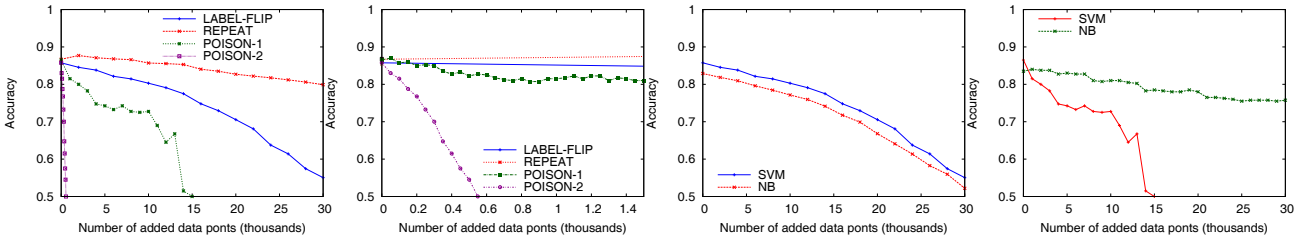


Figure 3: **[a,b] Comparison of REPEAT, LABEL-FLIP, POISON-1, and POISON-2 attacks (a) 0-30K attack datapoints and (b) 0-1.5K attack datapoints. [c,d] Comparison of attack effectiveness against different machine learning algorithms SVM and NB: (c) LABEL-FLIP and (d) POISON-1. All attacks are performed against training datasets with 30K datapoints.**

we provide Figure 3(b) which has the same results just drawn at a finer scale. Overall, we note that POISON-2, POISON-1, and LABEL-FLIP degrade accuracy to 50% with roughly 500, 15K, and 30K attacker datapoints respectively. Degrading to 50% is significant as the integrity of the classifier is fully broken at this point since the classifier has been reduced to a trivial random guessing classifier.

Creating mislabeled points is more effective than simply swapping labels as POISON-1 degrades performance twice as fast as LABEL-FLIP. When an adversary aims to degrade accuracy on a known testing set, the POISON-2 attack increases impact by an order of magnitude when comparing with the other attacks due to focusing the attack on features occurring most often in the testing set. Note that with only 500 tweets a POISON-2 attack can degrade to 50% the accuracy of a classifier. An attacker can easily generate these number of tweets, particularly if he has under his control several compromised accounts.

5.2 Sensitivity to learning systems

Learning algorithm. Figure 3(c) compares the effectiveness of the LABEL-FLIP attack for the SVM and NB classifiers. For LABEL-FLIP the NB classifier consistently performs worse than the SVM classifier by roughly 2% accuracy which is consistent as the attack occurs. The POISON-1 attack shown in Figure 3(d) has less impact on NB than on SVM as the SVM classifier is more sensitive to outliers which forces the model to adjust sharply to these outliers. The result shows that NB accuracy is only degraded by 5% at the point when the POISON-1 attack against the SVM classifier degrades accuracy to 50%.

Ensemble. Figures 4(a) and 4(b) show the accuracy achieved by the ensemble classifier as a function of the added datapoints. The ensemble reduces the effectiveness of the attacks which can be seen as a LABEL-FLIP attack requires 17K points against a victim with ensemble as opposed to 14K points against a victim without ensemble to degrade accuracy to the same point of 75%. A bagging ensemble could provide a valid mitigation strategy, but we do note that it takes more computational effort to train multiple classifiers, and the mitigation is not substantial enough to completely protect the integrity of a learning system.

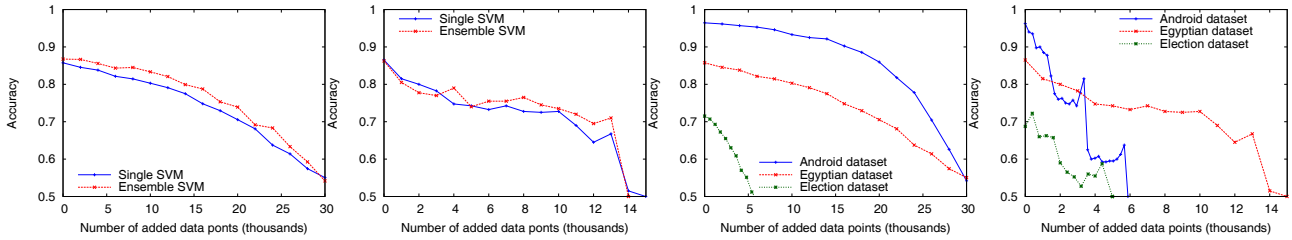
Training dataset. We show the effectiveness of LABEL-FLIP and POISON-1 against different datasets in Figures 4(c) and 4(d). Note that Egyptian and Android are trained with a training set of size 30K while Election is trained with a training set of size 12K. In the LABEL-FLIP experiment, we note that the Election dataset is more susceptible due to its smaller training set size, while for the Egyptian and Android datasets which are quite diverse, the LABEL-FLIP diminishes the accuracy to nearly 50% when the number of attack points is equal to the training set size. For the POISON-1 attack, the Android dataset allows datapoints up to 1200 characters long allowing POISON-1 to craft much stronger attacks points diminishing accuracy to 50% after 6K added datapoints instead of 15K added datapoints in the Egyptian dataset.

Training dataset size. We study the effectiveness of the LABEL-FLIP and POISON-1 attacks when the training dataset size changes, where all the training datapoints are selected from the Egyptian dataset. Figures 5(a) and 5(b). show that the attacks have less impact when the model is trained with larger training sets.

Sensitivity summary. From our attack sensitivity experiments we find the following. The POISON-1 attack had significant difference between the two classifiers. The classifier for SVM is more complex which allows SVM to exploit more structure in datapoints of the training set, but this complexity is the reason SVM suffers from the POISON-1 attack. NB considers frequencies of each feature separately, so POISON-1 attacking combinations of incorrect features does not significantly harm the model. A victim with a larger training set will force an attacker to use more effort in degrading accuracy. Additionally, a victim in a scenario where a datapoint is less constrained, such as Android comments compared with Twitter, is more vulnerable to POISON-1 attacks. A POISON-1 attacker is able to be more malicious with each datapoint due to the fewer constraints.

6. POISONING WITH PARTIAL KNOWLEDGE

So far in our evaluation we assume the attacker has full knowledge of the victim's processing steps in the learning system, and now we show cases where the attacker only has partial knowledge of these steps. We focus on partial knowledge for three typical data cleaning steps detailed in Section 2.2, Repeated Letter Removal (happppppy $->$ happy), Case Normalization (HappY $->$ happy), and Username Removal (@John $->$ username). In a real-world scenario, it may be possible that the attacker learns other aspects of the learning system, but he/she may be unable to learn *all* of these data cleaning steps. We evaluate scenarios with partial knowledge by having the attacker



(a) (b) (c) (d)

Figure 4: [a,b] Comparison of attack effectiveness against no ensemble and bagging ensemble: (a) LABEL-FLIP, (b) POISON-1. [c,d] Comparison of attack effectiveness against different datasets Egyptian, Election, and Android: (c) LABEL-FLIP and (d) POISON-1. All attacks are performed against training datasets with 30K datapoints.

Table 3: Most popular features used in attacks with partial knowledge

Full knowledge		Missing cleaning step: Repeated Letter Removal		Missing cleaning step: Case Normalization		Missing cleaning step: Username Removal	
username	1530	username	1530	username	1540	you	680
you	670	you	680	you	580	not	680
not	660	not	670	not	570	but	630
but	600	but	610	but	540	i'm	460
love	440	love	450	I'm	410	love	450
thanks	400	thanks	410	miss	370	thanks	410
good	400	i'm	440	love	370	miss	400
i'm	460	good	400	have	330	good	380
miss	400	miss	390	was	300	was	350
was	340	can't	320	don't	300	can't	330

create POISON-1 attack points when not performing one of those data cleaning steps. First, we provide some statistics which characterize how the attack points differ, and then we show the accuracy degradation by inserting these created attack points into a victim's training dataset where the victim performs all data cleaning steps.

Table 3 shows the frequency of the top 10 features used in 15K attack points. We gain two important insights from Table 3. First, the attack without the *Username Removal* cleaning step does not use username in its attacks. This is due to the attacker not mapping all usernames to the same feature, and the inclusion of a username in a tweet happens to be a strong indication of positive sentiment according to our Election dataset. Since, the *username* feature indicates positive sentiment, the poisoning attacks select the username feature to be included in attack datapoints that are labeled with negative sentiment. Second, despite this discrepancy with usernames, the top 10 most frequent features are fairly consistent across the different attacks with partial knowledge. Such similarities indicate that despite differences in the models learned by each attack with partial knowledge, the models learn the same features which have high indication of sentiment.

Table 4 shows statistics for the three scenarios of partial knowledge. The total number of features in the trained model for each scenario is shown, and for reference, the number of features in the model with all data cleaning steps is

Table 4: Statistics of attacks with partial knowledge

Feature Count	Missing cleaning step: Repeated Letter Removal	Missing cleaning step: Case Normalization	Missing cleaning step: Username Removal
Model	52034	59820	69438
Attack	5344	4596	5818
New	1746	1274	2455
Lost	1583	1859	1818
Same	3598	3322	3363

≈51K features. Data cleaning combines numerous features into more meaningful features so missing one of the steps (i.e., having only partial knowledge) increases the number of features. The "attack features" count captures the number of unique features used in the generated 15K attack datapoints for a given scenario of partial knowledge. We additionally show the number of "new", "lost", and "same" features used in the partial knowledge scenarios compared with the scenario where the attacker has full knowledge of the cleaning steps. We observed a large number of new features for the Username Removal case that were not from the additional new model features of distinct usernames. The increase in new features for this attack is actually due to the *username* feature being used less in the attack as observed in Table 3 which has both advantages and disadvantages in terms of attack effectiveness in this scenario.

Figure 5(c) shows the attack effectiveness for each scenario of partial knowledge. The scenario with full knowledge is shown as a baseline. Lacking the knowledge of the Repeated Letter Removal step has little difference from the scenario of full knowledge. Lacking the knowledge of Case Normalization step does harm the effectiveness of the attack as accuracy is degraded to 68% (as opposed to 50%) after 14K attack points. Interestingly, lacking the knowledge of Username Removal step results in a stronger attack when fewer attack datapoints are used, but the effectiveness plateaus after many attack datapoints are added.

We investigated the unexpected behavior when Username Removal step is not known to the attacker, and we first discuss the steep accuracy decrease with few attack datapoints and then the plateau behavior when numerous attack datapoints are added. The steep accuracy decrease is due to more non-username features being used in the attack since the lack of username cleaning puts less emphasis in the model on the username features. Thus, testing datapoints without the username attributes are inaccurately classified quicker. The plateau observed in the curve is due to not

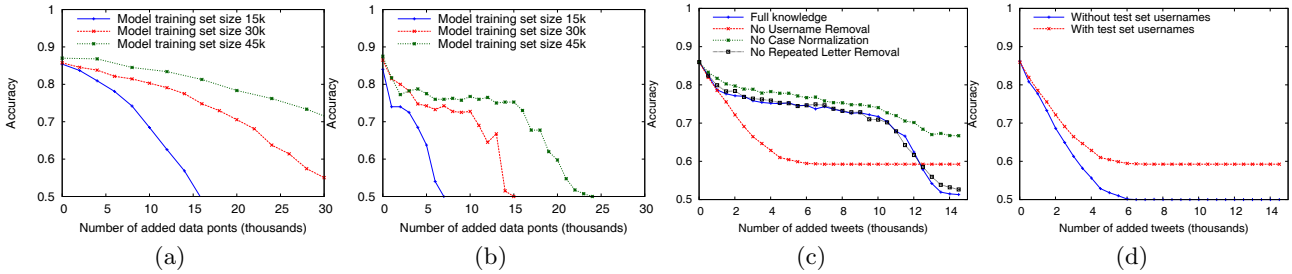


Figure 5: [a,b] Comparison of attack effectiveness against varying training set sizes 15K, 30K, and 45K: (a) LABEL-FLIP and (b) POISON-1. [c,d] Comparison of attack effectiveness for (c) all scenarios of partial knowledge and our standard testing set and (d) the username scenario of partial knowledge and a testing set with usernames removed. Attacks in [c,d] are performed against training datasets with 30K datapoints while those in [a,b] are performed on datasets of varying sizes.

having enough effect on those testing set datapoints which contain usernames. We verified that the usernames in the testing set were causing this plateau by including a comparison where we perform the same experiment with partial knowledge but remove all username features from the testing set in Figure 5(d). The case without usernames in the testing set has accuracy degradation to 50% without a plateau effect at 60% accuracy.

7. RELATED WORK

Poisoning SVMs: The closest existing work to ours is a poisoning attack against SVM classifier accuracy in the work of Biggio et al. [19]. That work formulates a non-convex optimization problem which maximizes loss of a testing set given a model learned from a training set with a poisoned data point. This optimization problem is based on complete information about the learner's data, and there are no constraints over their data points which allows them to use the technique of gradient ascent to find a good solution to this hard non-convex problem. We distinguish ourselves from that work as the attacker sometimes has partial knowledge about the learner, and we consider the domain-specific constraints on the search space. In our search space we are constrained to using only sparse binary vectors, and thus our search space itself is non-convex and causes problems. We choose to use a simple linear objective function instead that exploits SVMs sensitivity to outliers, and we employ heuristics to solve over the non-convex search space.

Adversarial machine learning: Learning in the presence of an adversary is an active area of research. The study of learning from examples with malicious errors was initiated by Valiant [11] and Littlestone et al. [31] which aimed to understand accuracy of a learning algorithm given a fixed probability of error. Barreno et al. [32], Newsome et al. [33], and Zhou et al. [18] studied the efficacy of intrusion detection systems based on machine learning when an attacker aims to subvert such a system. Biggio et al. [17] demonstrate how adversarial label flipping performs against an SVM which performs training to be robust to random label noise, and Xiao et al. [34] extend this work by improving the adversarial label changing while additionally considering the effects on SVMs with non-linear kernel functions. Our work considers a learning system system with all the practical preprocessing steps that warp the space of acceptable datapoints which can be altered. Finding attack datapoints in such a space is far more difficult due to the induced non-convexity, and we provide experimental results as opposed to analytical results shown in prior works.

Mimicry attacks: Existing work aims to understand how to craft datapoints that can bypass a classifier. Note that this is different from the attacks we consider, as the goal of the attack is not to influence the learning system, but to bypass detection. For example, the work in [26, 35, 36] studies spam, focusing on datapoint creation for getting past the spam filter. Work on malware detection in PDFs [20, 21] similarly shows how to craft datapoints (PDFs) which are classified as negative when they are actually malicious. This work considers similar practical constraints on the search space that we consider when degrading a classifier.

Opinion mining and sentiment analysis: Previous work studied opinion mining and sentiment analysis. Pang et al. [37] describe techniques and approaches for an opinion-oriented information retrieval. Yang et al. [38] use web-blogs to construct a corpora for sentiment analysis and use emoticons assigned to blog posts as indicators of users' mood. Read et al. [22] used emoticons to form a training set from Usenet newsgroup documents similar to how we have constructed training sets in our work. Lam et al. [39] provide preliminary analysis on evaluating classifiers with test data having noisy labels and analyze the bounds of the error rate for the classification. Pal et al. [40] build a probabilistic model of the classifier in the absence of a true label using a latent variable model.

8. CONCLUSION

We demonstrate how to subvert a real-world learning system for sentiment analysis, a common task that is critical in numerous data analytics applications. The attacks we present consider practical aspects of a learning system that were not considered in prior work. The practical concerns consist of the varied capabilities of an attacker, knowledge of an attacker, and constraints on feasible datapoints. We present attacks matching several classes of attackers with varied capabilities and knowledge. We formulate poisoning attacks, crafting malicious datapoints, to ensure created attack points meet constraints imposed by the set of feasible datapoints. We evaluate our attacks with three real-world datasets and two typical machine learning algorithms. As a further practical concern in our evaluation, we consider these poisoning attacks when the full learning system of the victim is unknown and the attacker must work with partial knowledge.

9. REFERENCES

[1] S. Rodgers and E. Thorson, "The interactive advertising model: How users perceive and process online ads," *JOIA*, 2000.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[3] J. Bollen, A. Pepe, and H. Mao, "Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena," in *CWSM*, 2011.

[4] M. D. Conover, J. Ratkiewicz, M. Francisco, B. Gonçalves, A. Flammini, and F. Menczer, "Political polarization on twitter," in *CWSM*, 2011.

[5] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment," in *CWSM*, 2010.

[6] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," in *Proceedings of LREC*, vol. 2010, 2010.

[7] R. Feldman, "Techniques and applications for sentiment analysis," *Communications of the ACM*, vol. 56, no. 4, pp. 82–89, 2013.

[8] I. Guyon, *Feature extraction: foundations and applications*. Springer, 2006, vol. 207.

[9] B. Nelson, B. I. Rubinstein, L. Huang, A. D. Joseph, S. J. Lee, S. Rao, and J. Tygar, "Query strategies for evading convex-inducing classifiers," *JMLR*, 2012.

[10] G. Ateniese, G. Felici, L. V. Mancini, A. Spognardi, A. Villani, and D. Vitali, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *arXiv*, 2013.

[11] L. G. Valiant, "Learning disjunctions of conjunctions," in *AI*, 1985.

[12] M. Kearns and M. Li, "Learning in the presence of malicious errors," *SIAM Journal on Computing*, vol. 22, no. 4, pp. 807–837, 1993.

[13] A. R. Klivans, P. M. Long, and R. A. Servedio, "Learning halfspaces with malicious noise," *JMLR*, 2009.

[14] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *Transactions on Knowledge and Data Engineering*, 2013.

[15] Y. Chen, C. Caramanis, and S. Mannor, "Robust sparse regression under adversarial corruption," in *Proceedings of ICML*, 2013.

[16] M. Großhans, C. Sawade, M. Brückner, and T. Scheffer, "Bayesian games for adversarial regression problems," *ICML*, 2013.

[17] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," *ACML*, 2011.

[18] Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and B. Xi, "Adversarial support vector machine learning," in *ACM SIGKDD*, 2012.

[19] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[20] N. Šrndic and P. Laskov, "Practical evasion of a learning-based classifier: A case study," in *Proceedings of S&P*, 2014.

[21] ——, "Detection of malicious pdf files based on hierarchical document structure," in *Proceedings of NDSS*. Citeseer, 2013.

[22] J. Read, "Using emoticons to reduce dependency in machine learning techniques for sentiment classification," in *ACL SRW*, 2005.

[23] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc*, vol. 68, no. 3, pp. 337–404, 1950.

[24] V. Vapnik, "The nature of 6tatistical learning theory," *Data mining and knowledge discovery*, pp. 1–47, 6.

[25] K. Thomas, C. Grier, D. Song, and V. Paxson, "Suspended accounts in retrospect: an analysis of twitter spam," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 243–258.

[26] G. L. Wittel and S. F. Wu, "On attacking statistical spam filters," in *Proceedings of the first conference on email and anti-spam (CEAS)*. California, USA, 2004.

[27] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*, 1st ed. New York, NY, USA: Cambridge University Press, 2011.

[28] "NIST Twitter Dataset for TREC 2011." [Online]. Available: {http://trec.nist.gov/data/tweets/}

[29] J. C. Schmitt, "Trigram-based method of language identification," Oct. 29 1991, uS Patent 5,062,143.

[30] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *ACL-02 NLPCL*, 2002.

[31] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *ML*, 1988.

[32] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. Tygar, "Can machine learning be secure?" in *ACM ICCS*, 2006.

[33] J. Newsome, B. Karp, and D. Song, "Paragraph: Thwarting signature learning by training maliciously," in *RAID*, 2006.

[34] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines," 2012.

[35] J. Graham-Cumming, "How to beat an adaptive spam filter," in *Presentation at the MIT Spam Conference*, 2004.

[36] D. Lowd and C. Meek, "Good word attacks on statistical spam filters," in *CEAS*, 2005.

[37] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *FTIR*, 2008.

[38] C. Yang, K. H.-Y. Lin, and H.-H. Chen, "Emotion classification using web blog corpora," in *IEEE Web Intelligence*, 2007.

[39] C. P. Lam and D. G. Stork, "Evaluating classifiers by means of test data with noisy labels," in *AI*, 2003.

[40] C. Pal, G. Mann, and R. Minerich, "Putting semantic information extraction on the map: Noisy label models for fact extraction," in *AAAI WIIW*, 2007.