Statistical Natural Language Processing
Brandeis University
Fall 2015

Analysis of YouTube Spam

Jing Zou
Huilin Gang
December 9, 2015

1 Purpose

YouTube becomes more lucrative targets for spammers hoping to attract unsuspecting users to malicious websites, where a variety of threats such as scams (phishing, e-commerce) and malware can be found. We demonstrate how such of these can be tracked over time using prime SVM and Neural Network Model. Based on both models, we will estimate how these models work with our dataset.

2 Dataset Collection

Following the lead of earlier related YouTube research, a data set was collected in order to permit the investigation of contemporary spam comment activity.These data are retrieved from YouTube API which provides access to video and user information. Data retrieval ran from October 31st, 2011 to January 17th, 2012.

Table 2.1: Data Properties

Videos	6407
Total comments	6431471
Total users	2860264
Spam comment users	177542

Table 2.2: Sample data from training dataset

published	videoId	userId	commentText	spam
1321991078000	video1473	user812400	i lost count after last night at the gay bar	false

3 Preprocessing

Our methodology requires the generation of a network to represent the comment posting activity of users to a set of videos. Initially, the focus of the evaluation is English-language spam comments.

The original dataset is way to large for us to train our model and go on with experiments. Therefore, we prepared the datasets with 100000 comments. This dataset will be our original dataset

Also, to extract the English comments in the original dataset, we included the ENCHANT package in our preprocessing. We tried two different methods to extract the English comments. For the first one, we put the whole sentence for ENCHANT to check and collect those with only exact English words. This method works poor for the reason that it only extracts the simple comment with less words like "Good!". Also, most of the comments are same. Therefore, we tried the second method to check the percentage of English words in the sentence. Before we do the extraction, we included NLTK tokenization tool to tokenize the comment first.

4 Methodologies

We will implement both Support Vector Machine and Neural Network models in our final project to classify the spam comment. The optional model comes Network Motif Analysis, which is generated in the previous research of Youtube spam analysis [Derek O Callaghan, Martin Harrigan, Joe Carthy, P adraig Cunningham]. We will implemented it if we have enough time, otherwise we will just add it into our experimentation.

1. SVM
   Our focus is how SVM can have good practical performance scaled to work with large training sets. We are going to implement algorithms use variant of primal stochastic gradient descent(SGD).

   (a) Generalization
   A learning problem mush have a measure that says how good and algorithm performs on the task. A universal goal is *generalization*. We try to minimize some function over the training set like:

   $$\hat{g}(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(x_i, y_i; \theta)$$

   where L is a loss function.

   (b) Regularization
   Regularization refers to augmenting the objective function g(θ) with an extra term that penalizes complex θ vectors. Therefore, the general form becomes

   $$\hat{g}(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(x_i, y_i; \theta) + r(\theta)$$

   where r(θ) is a regularization term.

   (c) Stochastic Gradient Descent
   Gradient method computes $\theta = \text{argmin } \hat{g}(\theta)$, where $\hat{g}(\theta)$ is the sum of empirical loss and regularization terms. Stochastic Gradient Descent(SGD) is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as Support Vector Machines. It has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification. We do not compute the gradient using the entire training set, however with respect to only a single randomly chosen example. The update rule is then:

   $$\theta_{t+1} = \theta_t - \eta_t \nabla L(x_{i(t)}, y_{i(t)}; \theta_t) - \eta_t \nabla_r(\theta_t)$$

   where $i(t)$ is an index drawn randomly from {1, 2, ..., n}. In expectation, this update is the same as gradient descent, because $E_{i(t)}[l(x_{i(t)}, y_{i(t)}; \theta_t)] = \frac{1}{n}\sum_i l(x_i, y_i; \theta)$

   (d) The primal SVM formulation
   There are two ways of treating the SVM problem. The classical method is

the hard margin SVM[Vapnik and Lerner, 1963], which assumes that the dataset is linearly separable: hence every point must be correctly classified by the maximum margin hyperplane. The soft margin SVM [Bennett and Mangasarian, 1992, Cortes and Vapnik, 1995] allows for some points to be misclassified, but penalizes these points appropriately. We implemented the latter represented with the following optimization problem.
In contrast to (batch) gradient descent, SGD approximates the true gradient of $E(w, b)$ by considering a single training example at a time.
By implementing a first-order SGD learning routine, the algorithm iterates over the training examples and for each example updates the model parameters according to the update rule given by

$$\theta \leftarrow \theta - \eta(\alpha \frac{\partial r(\theta)}{\partial \theta} + \frac{\partial L(\theta^T x_i + b, y_i)}{\partial \theta})$$

where η is the learning rate which controls the step-size in the parameter space. The intercept b is updated similarly but without regularization.
The learning rate η can be either constant or gradually decaying. We give the default learning rate to be

$$\eta^{(t)} = \frac{1}{\alpha(t_0 + t)}$$

2. Neural Network
Instead of using bag-of-word representation, we plan to use word-cluster representation. Brown clustering algorithm introduces a good way to do clustering.

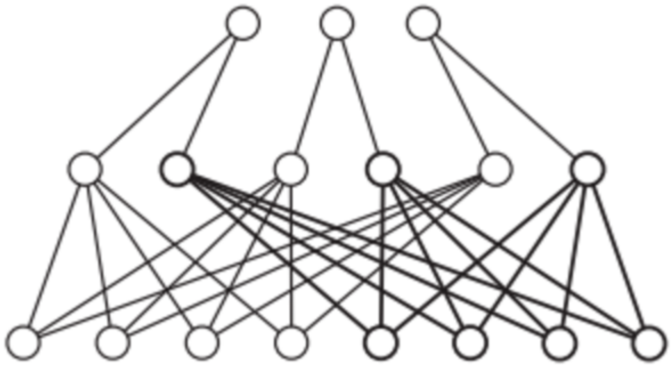
   (a) Brown Clustering
   Word embedding is very useful in deep learning area that mapped words to vectors in a low dimensional space, comparing to the vocabulary size. One of popular approaches such is Brown clustering algorithm. Brown clustering is an agglomerative clustering algorithm that clusters words based on which words precede or follow them. Each word is initially assigned to its own cluster. Then we will consider merging each pair of clusters. Two words are merged that have similar probabilities of preceding and following words. Clustering proceeds until all words are in one big cluster. This model has the same general form as a hidden Markov model. Given cluster membership indicators c_i for the tokens w_i in a text, the probability of the w_i given w_{i-1} is given by

$$P(w_i|w_{i-1}) = P(c_i|c_{i-1})P(w_i|c_i)$$

   (b) Neural Network Language Model
   Neural neworks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is output as shown in the graphic below. For example, a neural network structure for 3 classes, 8 features and 2 feature groups per class:
   Let n be the number of output units, g the number of feature groups per class and gn the number of hidden units given by the product. The output of the network for an input x is:

$$f(x) = W_o tanh(W_h x + bh) + b_o$$

   where W_o is a $n \times gn$ matrix, W_h is a $gn \times |V|$ matrix, b_o is a vector of dimension n and bn is a vector of dimension gc.



Let $w_1, ..., w_n$ be a set of weight vectors derived from a linear model. Let $\pi_1, ..., \pi_g$ be a random partition of the set of feature indices $1, 2, ..., |V|$ into g equally-sized groups. For each class c, associate g hidden units $h_1^{(c)}, ..., h_g^{(c)}$. For all $i \in [1, g]$ and $j \in [1, |\pi(i)|]$, the weight of the connection between $h_1^{(c)}$ and the input unit $x_{\pi_j}^{(i)}$ is given by $(W_c)_{\pi_j^i}$

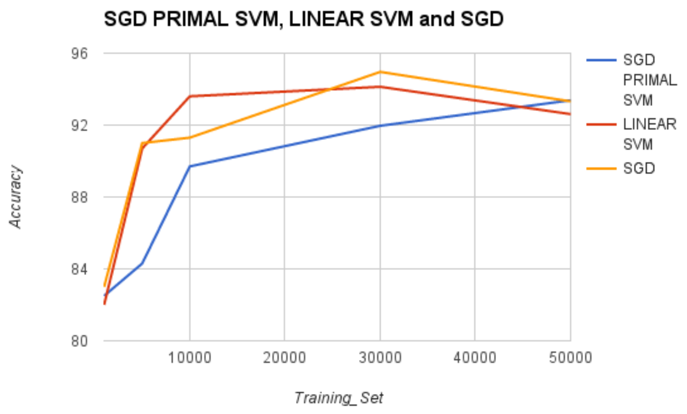
5 Experiments and Result

1. Support Vector Machines

Table 5.1: SVM accuracy with training data set 50000 and learning rate 0.0001

SGD PRIMAL SVM CLASSIFIER	93.38%
LINEAR SVM CLASSIFIER(SKLEARN)	92.61%
SGD CLASSIFIER(SKLEARN)	93.32%

Here is a figure with different training dataset associated with accuracy in the three model



2. Neural Network

6 References

O'Callaghan, Derek, et al. "Network Analysis of Recurring YouTube Spam Campaigns." ICWSM. 2012.
Menon, Aditya Krishna. "Large-scale support vector machines: algorithms and theory." Research Exam, University of California, San Diego (2009): 1-17.
Chapelle, Olivier. "Training a support vector machine in the primal." Neural Computation 19.5 (2007): 1155-1178.