

# Semi-Supervised Learning on Email Characteristics for Novel Worm Detection

Steve Martin and Anil Sewani  
 { steve0, anil }@eecs.berkeley.edu

University of California at Berkeley

## Improving Classifiers

- Previous work: novelty detection in general is often not enough by itself.
  - False negatives extremely undesirable.
  - Common solution: make the novelty detector model very sensitive.
  - This introduces additional false positives.
- Idea: leverage knowledge gained by supervisor feedback
  - Use feedback to partially label data corpus
- Filter novelty detection results using a classifier trained on *semi-supervised* data.

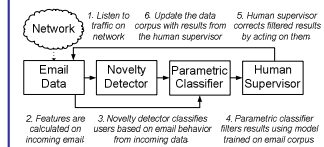
## Application: Email Worms

- Email worms have caused billions of dollars of damage
  - MyDoom, Sobig, LoveLetter, etc all spread by email.
- Signature-based methods are effective against known worms only.
  - Human element slows reaction times
  - Need an adaptive first line of defense.
- Employ novelty detection on network behavior.
  - Basic idea: leverage behavioral invariant among infected machines.
  - By definition, a worm seeks to propagate itself over a network.

## Our Approach

- Novelty detection by itself is not enough.
  - False negatives not acceptable, worm attack will succeed.
  - Too many false positives overwhelm network admins.
- Solution: two-layer approach to filter novelty detector results.
  - Novelty detector minimizes false negatives.
  - Use secondary classifier to filter out false positives.
- Use human reactions to improve secondary classifier.
  - Introduces partial labeling for semi-supervised learning.

## Improved Worm Detection



## Classifier Overview

- Exploit distinct feature distributions using a *generative graphical model*.
  - Feature distributions of worm and non-worm behavior differ
- Current implementation uses a multi-layer naive Bayes approach.
  - Assumes all features are independent.
  - Allows us to fit specific distributions to feature data.
- Classifier initially trains on supervised test set.
  - Over time, classifier retrain over *semi-supervised* data.

## Features

- Distinguished between *per-email* features and *per-user* features.
- User features capture elements of behavior over a window of time.
  - Examples: mean number of words in emails, ratio of emails sent with/without attachments, etc.
- Email features examine individual snapshots of behavior.
  - These are then aggregated over a window of time.
  - Examples: number of characters in the email subject, presence of html in the email, etc.

## Feature Distributions

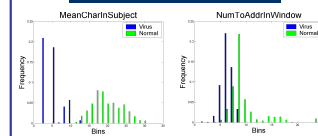


Figure 1. Distributions of the average number of characters in an email subject in Bagle.A worm vs. normal email activity.

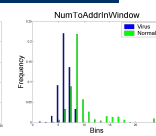


Figure 2. Distributions of the number of address sent over time in LoveLetter.C worm and normal email activity.

## Feature Distributions II

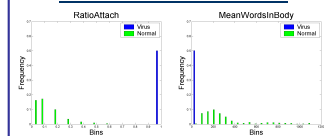


Figure 3. Distributions of the ratio of attachments between LoveLetter.C worm and normal email activity.

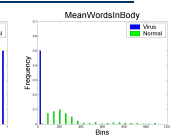
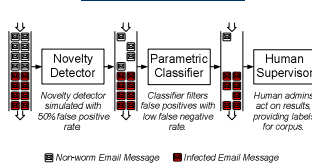


Figure 4. Distributions of the average number of words per email in Bagle.A worm vs. normal email activity.

## Preliminary Evaluation

- Instrumented a mail server to calculate feature data on live email.
  - Conducted a user study to collect data and learn distributions of our features.
- Collected virus data from three real worms using mail server and virtual machines.
  - Klez, LoveLetter.C, and Bagle.A.
- Constructed training set of real email traffic artificially 'infected' with two viruses.
- Tested on sets of email traffic infected with the third virus as the 'novel' virus.

## Current Process



## Preliminary Results

Table 1. Parametric Filter Classification Results

Worm Name	Total Emails	# Worm Emails	# Clean Emails	False Positives	False Negatives	Correctly Classified
Bagle	1090	789	301	6 (0.76%)	6 (1.99%)	1078 (98.90%)
Klez	1090	789	301	4 (0.50%)	15 (4.98%)	1071 (98.26%)
LoveLetter	1090	787	303	9 (1.14%)	5 (1.65%)	1076 (98.72%)

## Future Work

- Examine additional features and refine model
  - Incorporate dependencies between features
- Examine possible attacks worms could use to defeat our methods
- Consider other methods of filtering.
  - Our model is parametric; non-parametric methods may work better (decision trees, etc).
- Implement our algorithms into a deployable system.
  - Current implementation primarily Matlab and Perl.