Predicting Malicious URLs

Proofpoint, Inc.

Aidan Cheng
Kevin Herrera
Carli Lessard
Vidushi Ojha (PM)

Advisor:
Prof. Elizabeth Sweedyk

Liaisons:
Thomas Lynam
Mike Morris ‘97

Proofpoint, Inc.

- Cybersecurity firm
- Provides software as a service
- Products that protect against malware, ransomware
- Products for email, social media, mobile devices, and the cloud

Proofpoint's security suite **analyzes URLs** embedded in client emails and detects whether they **lead to malware**, blocking those it deems dangerous. Our task was to investigate various **machine learning models** to improve the accuracy of the classification process.

Project Description

Existing Solution

- Filtration technique:
    - Number of appearances in time period
    - Domains passed through
- Sandboxing: sped-up virtual environment

Data Sample


```
{ u'queue_ts': datetime.datetime(2017, 5, 4, 0, 56, 37, 485000),
  u'url': u'https://www.cs.hmc.edu/program/course-descriptions/',
  u'recv_ts': datetime.datetime(2017, 5, 4, 0, 41, 4),
  u'misc': { u'content': None,
             u'ip': u'134.173.193.149',
             …,
                u'details': None,
                u'scanid': u'3096224878164377',
                u'forensics_score': 0,
                u'result': u'malicious' },
  … }
```

Models

Support Vector Machines

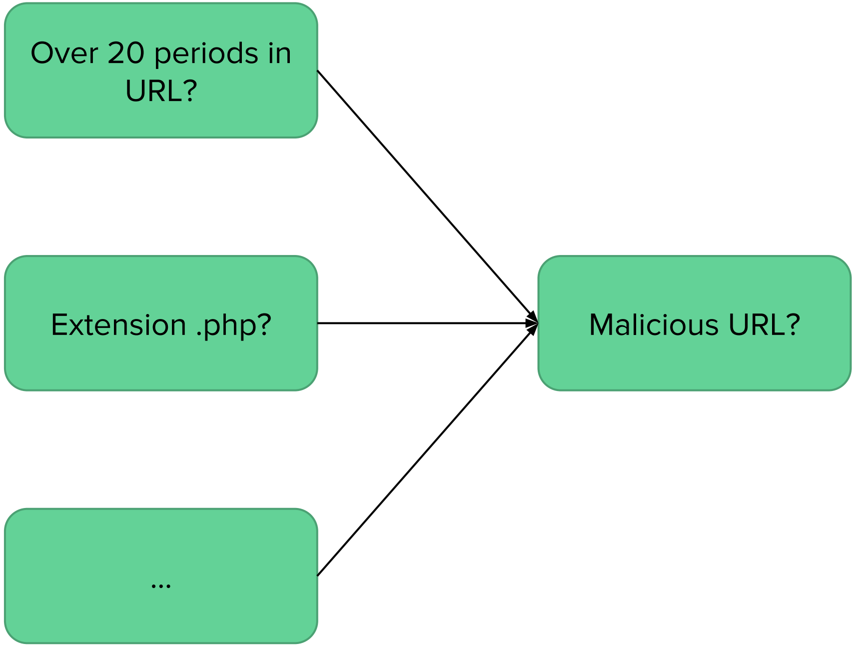
Naive Bayes

Logistic Regression

scikit learn

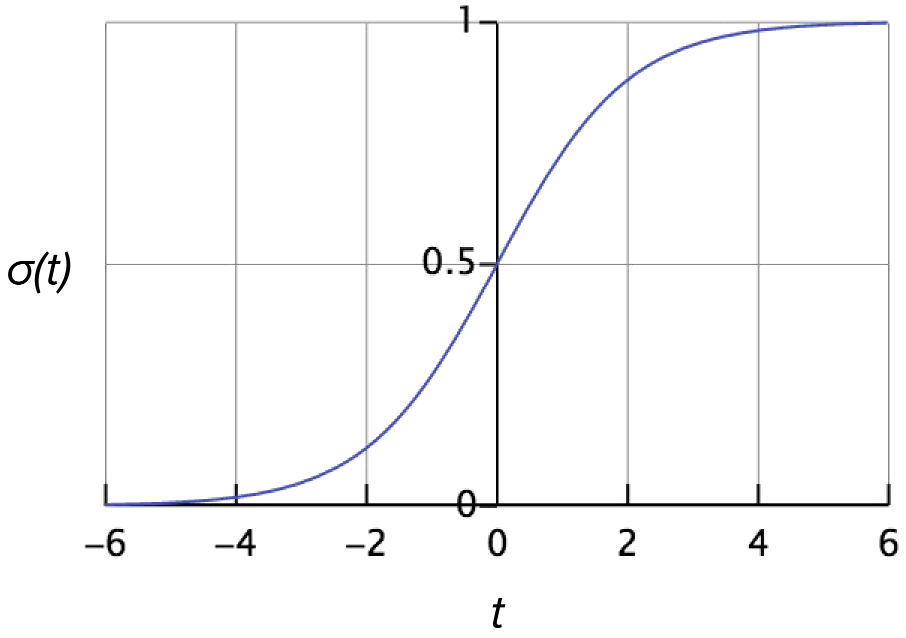
TensorFlow

Naive Bayes



- Bayes’ net encodes *conditional dependence*
- Independent features make it *naive*
- Use probability rules to compute likelihood of URL being malicious

Logistic Regression



$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

where

$$t = Wx + b$$

Use gradient descent optimization algorithm to determine the best coefficients, *W* and *b* for our model

Features

URLs

- Tokenization on all punctuation
- Character *n*-grams

`https://www.cs.hmc.edu`

‘https’	**‘http’**
‘www’	**‘ttps’**
‘cs’	**‘tps:’**
‘hmc’	**...**
‘edu’	**‘.edu’**

IP addresses

- Tokenization on ‘.’
- Word *n*-grams

`134.173.193.149`

[‘134’, ‘173’]
[‘173’, ‘193’]
[‘193’, ‘149’]

Testing

Offline

- Splits up the data set we have intro training and testing
- Learns from the training set
- Test on the testing set
- Repeat with a different split

Online

- Trains on the first 1000 chronological samples
- Tests on the next 1000 samples
- Train on the 1000 samples just tested, plus most recent malicious samples
- Repeat

Experiments

Model	Data	Testing Method	Parameters
Naive Bayes	URLs	Offline	For example: ● Iterations ● Malicious set size ● Etc.
Logistic Regression	IP addresses	Online	

Naive Bayes Results

	% of malicious URLs correctly classified	% of clean URLs correctly classified
Offline URL	37.7%	95.5%
Offline IP	75.8%	96.4%
Online URL	55.9%	92.7%
Online IP	85.3%	53.2%
Online URL & Offline IP	78.1%	95.1%

Logistic Regression Results

	% of malicious URLs correctly classified	% of clean URLs correctly classified
Online URL	78.6%	80.2%
Online IP	52.2%	94.2%
Online URL & IP	78.4%	90.6%

Combining Models

- Added Naive Bayes' predicted probability of being clean as feature to Logistic Regression

	% of malicious URLs correctly classified	% of clean URLs correctly classified
Bayes + URL + IP	78.4%	90.4%
Bayes + URL	80.1%	80.9%
Bayes + IP	65.2%	90.0%

Takeaways

- Chronological nature of data: online model
- IP addresses
- Combining models that complement one another

Future Directions

Models:

- Neural Nets
- AdaBoosting

Features:

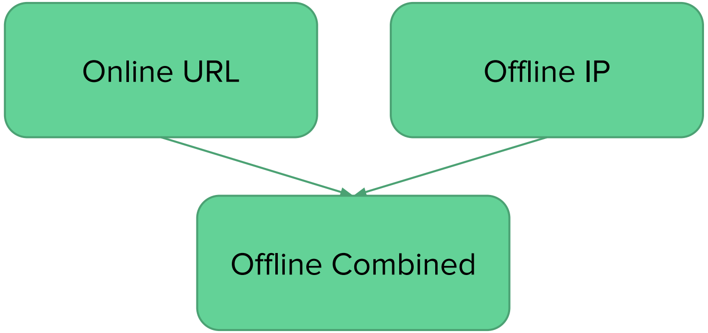
- Email subject line
- IP identities (WHOIS)

With Thanks To:

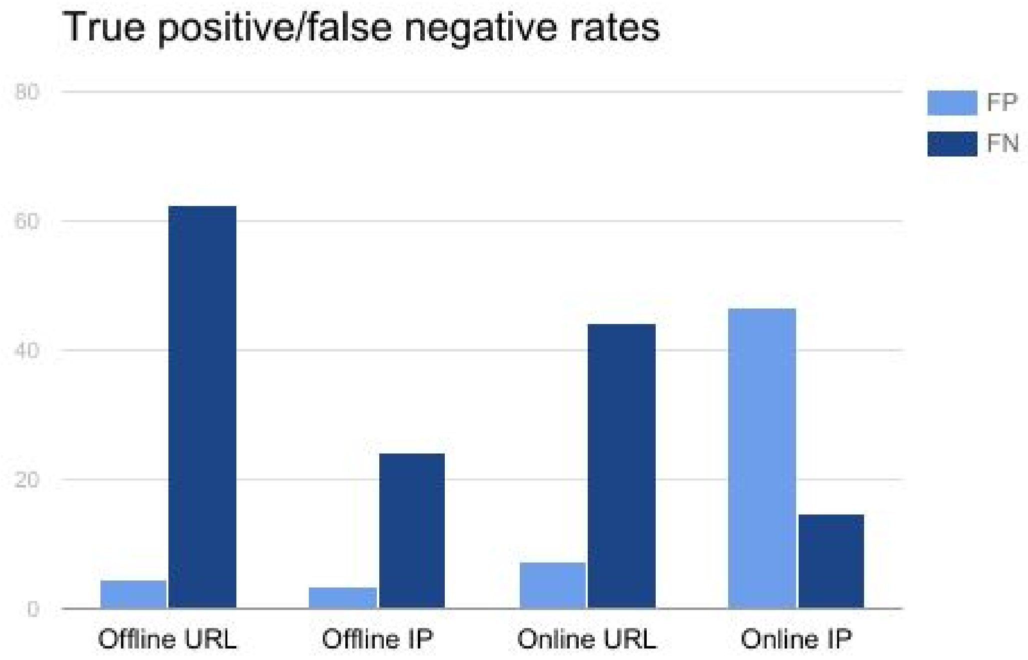
- Prof Z Sweedyk
- Mike Morris ‘97
- Thomas Lynam
- DruAnn Thomas
- Prof Geoff Kuenning

Questions?

Naive Bayes



- Our final model combined the two classifiers
- An online classifier worked best for URLs, and Offline worked best for IPs



Results for Naive Bayes

Logistic Regression

Online vs. offline

Offline:
Train on a set, test on a different set

Online:
Continuously learn from samples as they come in

Number of iterations

Count:
Frequency of features in URLs

Tf-idf:
Frequency of features in URLs times inverse document frequency

Features

N-grams:
Word or character

N-gram ranges:
Take n-grams of every size n in some range

Experiments

Online vs. offline

Offline:
Train on a set, test on a different set

Online:
Continuously learn from samples as they come in

Vectorizer type

Count:
Frequency of features in URLs

Tf-idf:
Frequency of features in URLs times inverse document frequency

Features

N-grams:
Word or character

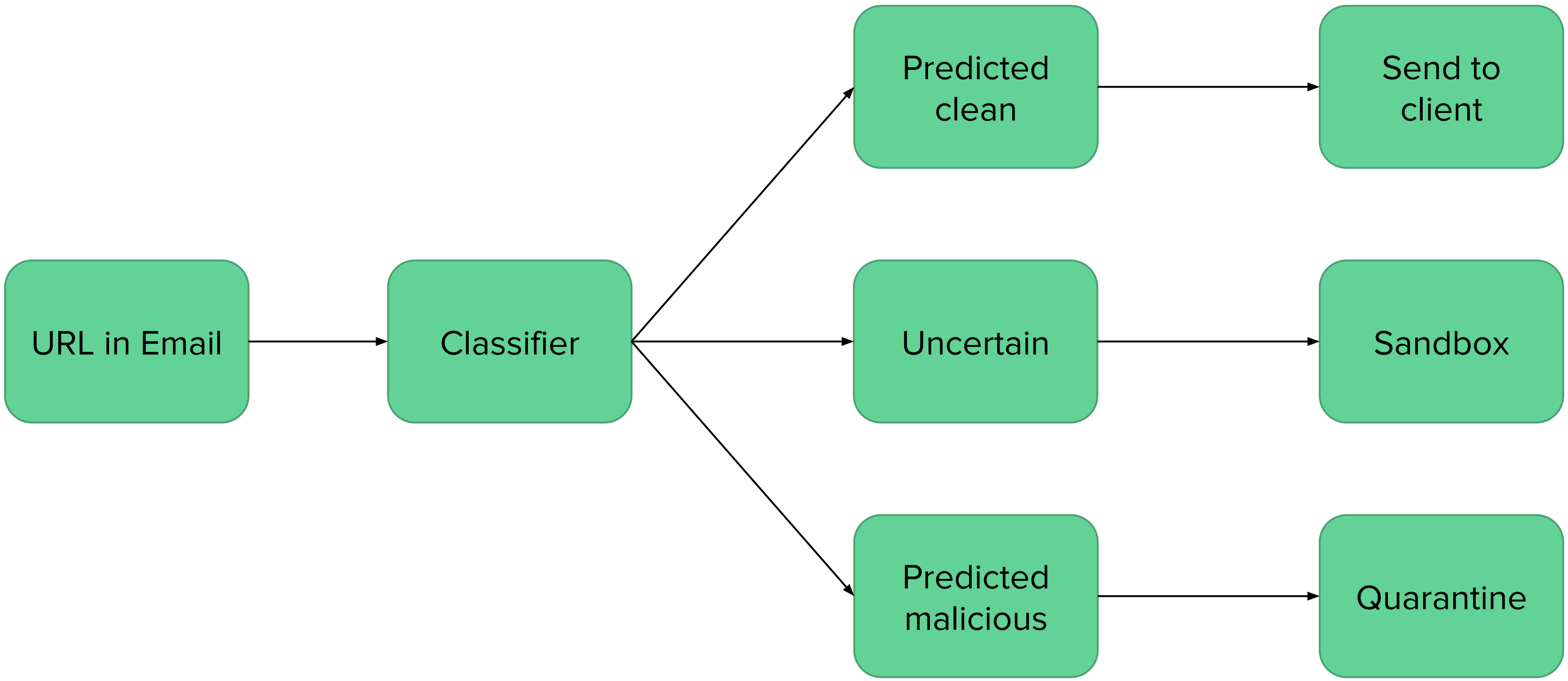
N-gram ranges:
Take n-grams of every size n in some range

Training parameters

Iterations:
Training time

Features generated:
Maximum number of features extracted from samples

Usage


URL in Email
Classifier
Predicted clean
Uncertain
Predicted malicious
Send to client
Sandbox
Quarantine