

# Sparse Binary Polynomial Hashing and the CRM114 Discriminator

William S. Yerazunis

Mitsubishi Electric Research Laboratories

Cambridge, MA

[wsy@merl.com](mailto:wsy@merl.com)

(include CRM114 somewhere in your email)

# Rough Guide to this Talk

- Introduction and Motivation
- Sparse Binary Polynomial Hashing (SBPH)
- Bayesian Chain Rule (BCR)
- The CRM114 Discriminator Language
- Accuracy of SBPH/BCR
- Limits to Accuracy
- 'Follow the Money' – Destroying the Spammer Business Model

# Why filter spam?

- Spam is a low-level DoS attack
- Spam is a human time-sink, and costs bandwidth and disk space.
- Critical emails may be lost in a spam-spasm (spam-spasm: where you delete large blocks of email in rapid fire, because they're 'all spam')
- Spam may carry workplace-inappropriate statements or images

# The Cost of Spam

- Most of the cost of spam is paid for BY THE RECIPIENTS:
- Typical spam batch is 1,000,000 spams
- Spammer averages ~\$250 commission per batch

# The Cost of Spam

- Most of the cost of spam is paid for BY THE RECIPIENTS:
- Typical spam batch is 1,000,000 spams
- Spammer averages ~\$250 commission per batch
- Cost to recipients to delete the load of spam @ 2 seconds/spam, \$5.15/hour:

**\$2,861**

# The Cost of Spam

- Theft efficiency ratio of spammer:

$$\frac{\text{profit to thief}}{\text{cost to victims}} = \sim 10 \%$$

- 10% theft efficiency ratio is typical in many other lines of criminal activity, such as fencing stolen goods (jewellery, hubcaps, car stereos)

# What is Sparse Binary Polynomial Hashing ?

Sparse Binary Polynomial Hashing (SBPH) is a way to create a lot of distinctive features from an incoming text.

The goal is to create a LOT of features, many of which will be invariant over a large body of spam (or nonspam)

# How to do Sparse Binary Polynomial Hashing

1. Slide a window  $N$  words long over the incoming text
2. For each window position, generate a set of order-preserving sub-phrases containing combinations of the windowed words
3. Calculate 32-bit hashes of these order-preserved sub-phrases

# What is an SBPH 'feature' ?

- Single words hash to features (e.g. ' **S1618** ')
- Phrases hash to features (e.g. '**to unsubscribe**')
- Even phrases with wildcards make features
  - (e.g. '**BUY <foo> ONLINE NOW**' )
- Each word in a text affects  $2^{N-1}$  features, we use  $N=5$  (a five-word sliding window) in CRM114
- Far more features than words than there were words in the original text!

# SBPH Example

Step 1: slide a window N words long over the incoming text. ex:

You can Click here to buy viagra online NOW!!!

Yields:

You can Click here to buy viagra online NOW!!!

You can Click here to buy viagra online NOW!!!

You can Click here to buy viagra online NOW!!!

You can Click here to buy viagra online NOW!!!


... and so on... (on to step 2)

# SBPH Example

Step 2: generate order-preserving sub-phrases from the words in each of the sliding windows

Sliding Window Text :  
'Click here to buy viagra'

...yields all these  
feature sub-phrases



Click  
Click here  
Click to  
Click here to  
Click buy  
Click here buy  
Click to buy  
Click here to buy  
Click viagra  
Click here viagra  
Click to viagra  
Click here to viagra  
Click buy viagra  
Click here buy viagra  
Click to buy viagra  
Click here to buy viagra

Note the binary counting pattern-  
this is the 'binary' in 'sparse binary  
polynomial hashing'

# SBPH Example

Step 3: make 32-bit hash value “features” from the sub-phrases

Click  
Click here  
Click to  
Click here to  
Click buy  
Click here buy  
Click to buy  
Click here to buy  
Click viagra  
Click here viagra  
Click to viagra  
Click here to viagra  
Click buy viagra  
Click here buy viagra  
Click to buy viagra  
Click here to buy viagra

32-bit hash

E06BF8AA  
12FAD10F  
7B37C4F9  
113936CF  
1821F0E8  
46B99AAD  
B7EE69BF  
19A78B4D  
56626838  
AE1B0B61  
5710DE73  
33094DBB  
..... and so on

# Evaluate these features with the Naive Bayesian Chain Rule

- **learning:** each feature is bucketed into one of one million buckets in one of two bucket files (one spam, one nonspam)
- **Classifying:** the comparable bucket counts of the two files generate rough estimates of each feature's 'spamminess'

$$P(F|C) = 0.5 + ( |F_c| - |F_{\sim c}| ) / ( 2 * \text{MaxF} )$$

# Quick review: the Bayesian Chain Rule (BCR)

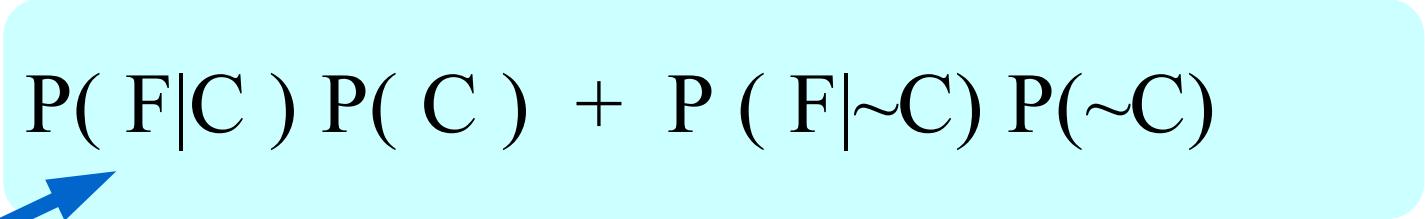
$$P(C|F) = \frac{P(F|C) P(C)}{P(F|C) P(C) + P(F|\sim C) P(\sim C)}$$

# Quick review: the Bayesian Chain Rule (BCR)

Conditional probability of a class C given Feature F


$$P(F|C) P(C)$$

$$P(C|F) = \frac{\quad}{\quad}$$


$$P(F|C) P(C) + P(F|\sim C) P(\sim C)$$

Renormalize for both possibilities of class C and not-class-C

# Naive Bayesian Parameters

- a-priori probability:  $P(C) = P(\sim C) = 0.5$
- Decision threshold:  $P = 0.5$
- SBPH/BCR has a VERY steep ROC curve: >99 % of messages evaluate to the top or bottom .  
0000000001'th of the probability range 0.0 – 1.0 (the ROC curve is basically a step function)
- SBPR/BCH is essentially *insensitive* to parameters

# So it's Naive Bayesian underneath?

- Yes, CRM114 uses a Naïve Bayesian classifier.
- The better feature set created by the SBPH feature hash gives better performance.
- Hashing and bucketing keeps all evidence from the training phase for use during classification
- Phrases in colloquial English are much more standardized than words alone- this makes filter evasion much harder

# That gives us probabilities...how do we use those for sorting mail?

- One option: make this engine into a PERL module.
- But frankly, *PERL creeps me out...*
- The obvious remaining logical option: make up a new language to express these kinds of matchings naturally.

# The CRM114 Language

- Interpreted **generic** mutilating filter language
- Based on regex matching as the primitive bind & test operator, and overlaid strings as the basic data structure.
- Data windowing for operation on truly infinite data streams (eg. Syslog streams )
- ANSI C implementation language

# CRM114 primitives

- Input/output with native data windowing
- regex test, bind, block-structured iterators (approximate match “in testing”)
- hash, learn, classify, union, and intersection
- surgical and nonsurgical string-alter operations
- subprocess management (both synchronous and asynchronous)

# CRM114 Example

'Shrouding" a message (remove the words, but keep the word positions and statistics, so test messages can be shared between multiple spamfilter developers without overtly violating privacy or copyright)

# CRM114 Example

'Shrouding" a message (remove the words, but keep the word positions and statistics, so test messages can be shared between multiple spamfilter developers without overtly violating privacy or copyright)

(yes, this algorithm is easily crackable- it's just a toy example. Work with me on this, OK?)

# CRM114 Example

'Shrouding" a message:

```
#!/usr/bin/crm
{
    {
        match < fromend > ( :word: ) /[[:graph:]]+/  
hash ( :word: ) /:*:word:/  
liaf          # Loop Iterate Awaiting Failure of match  
    }  
accept      # so we accept it to stdout and we're done.  
}
```

# CRM114 Example

'Shrouding" a message:

```
#!/usr/bin/crm
```

```
{
```

```
{
```

```
match < fromend > ( :word: ) /[[[:graph:]]+/  
hash ( :word: ) /:*:word:/  
liaf # Loop Iterate Awaiting Failure of match
```

```
accept
```

```
}  
accept # so we accept it to stdout and we're done.
```

```
}
```

```
}
```

# CRM114 Example

running this program on itself gives:

99E05DBF BF1C3F05

8BE19BD9

8BE19BD9

6CAF79C0 01E801E0 3DD0C452 11F911F1  
5945513D CBF25052 D14DD945 38EBC052

9B9E5993 5945513D CBF25052 D14DD945  
6A05F277

20938F26 A11AA91A 01525647  
A4A53CB6 9F22D2A8 0971BCE9 97319BEB 6CAF79C0

9BF2ABEA

A3704CCE A11AA91A EF986160 9ED42033  
A3704CCE 192B4BF4 EE580C1C 8ED91AF9 AA8D73B8 30C3C80A  
0D709357

# CRM114 and Mailfilter

- Mailfilter is a filter program, written in CRM114, to separate spam from nonspam.
- CRM114 provides the primitives; Mailfilter uses those primitives to create a moderately usable antispam email filter

# CRM114 and Mailfilter

- All access to Mailfilter training, whitelists, blacklists, etc. is done by mailing yourself a message with a command, a password, and arguments
- You can reprogram your Mailfilter remotely
- No special user mail agent is required- **any mail reading program will work.**

# CRM114 and Mailfilter

- Mailfilter is inserted into the mail delivery chain either via Procmail or via the '.forward' hook of Sendmail
- This step of installation requires 'root' privs and some system knowledge.

# Training CRM114 Mailfilter

- Just read your email. If you get a misclassified message, mail it back to yourself, prefixed with the proper command and password
- Train Only Errors ( the TOE strategy) works well for SBPH/BCR; corpuses of under 100Kbytes are useful at the >98% accuracy level

# How good can it get ?

- A bigger corpus of example text is better
- With 400Kbytes selected spams, 300Kbytes selected nonspams trained in, no blacklists, whitelists, or other shenanigans:

# How good can it get ?

- A bigger corpus of example text is better
- With 400Kbytes selected spams, 300Kbytes selected nonspams trained in, no blacklists, whitelists, or other shenanigans:

>99.915 %

How good can it get ?

>99.915 % (N+1 accuracy)

- This is the *actual performance* of CRM114 Mailfilter from Nov 1 to Dec 1, 2002.
- 5849 messages, (1935 spam, 3914 nonspam)
- 4 false accepts, ZERO false rejects, (and 2 messages I couldn't make head nor tail of).
- All messages were incoming mail 'fresh from the wild'. No canned spam...

How good can it get ?

>99.915 % (N+1 accuracy)

Filtering speed: classification: about 20Kbytes per second, learning time: about 10Kbytes per second (on a Transmeta 666 Mhz laptop)

Memory required: about 5 megabytes

404K spam features, 322K nonspam features

How good can it get ?

>99.915 % (N+1 accuracy)

For comparison, a human\* is only about 99.84% accurate in classifying spam v. nonspam in a “rapid classification” environment.

\* in this case, the author, because as even a grad student won't classify 3900 spams twice.

# Downsides?

The bad news:

**SPAM MUTATES**

Even a perfectly trained Bayesian filter will slowly deteriorate

New spams appear, with new topics, as well as old topics with creative twists to evade antispam filters.

# How fast does spam mutate?

A VERY ROUGH\* empirical estimate of the rate of spam evolution:

$$\text{Newspams} = \text{Totalspams} \times 0.001 \text{ x days}$$

In realistic terms, this is really about 1 to 3 truly new spams or spam methods per month.

\* pronounced “by casual observation, wholly unsubstantiated”

We're shooting at a target that's not just moving, but one that's actively making evasive maneuvers!

Sometimes these new spam technologies require a structural change to deal with (such as the *Spammus Interruptus* spams that started appearing around Pearl Harbor day, 2002)

Cases like these require significant human intervention, and/or classifying the email in 'eye-space' rather than 'ascii-space'..

How good does it need to be?

*FOLLOW THE MONEY*

- Spamming **is a business**
- Our target is not the spammer
- Our target is not the spammer's ISP

# How good does it need to be?

## *FOLLOW THE MONEY*

- Spamming is a business
- Our target is not the spammer
- Our target is not the spammer's ISP
- Our target is the

**SPAMMER'S BUSINESS MODEL**

How good does it need to be?

*FOLLOW THE MONEY*

- The typical spammer commission is between \$250 and \$500 per million spams, and gets a .0001 response rate.
- Paper junk mail costs about 25 cents per unit, and gets a .05 response rate.
- We need to drive the amortized cost of a spam above that of junk mail.

## *FOLLOW THE MONEY.....*

- Amortized cost of one email spam response: roughly 2.5 cents
- Amortized cost of one junk mail response: roughly \$5.00
- We need widely implemented spam filters at 99.5% or better to drive the spammers out of business.

# *Conclusions*

Bayesian spamfilters (especially SBPH/BCR ) filters are more than adequate to destroy the spammer's business model.

# *Conclusions*

Bayesian spamfilters (especially SBPH/BCR ) filters are more than adequate to destroy the spammer's business model.

Other high accuracy filter techniques exist- we should **deploy as many different methods as possible** to avoid monoculture-induced wide-area failures whenever spam evolves in an unforeseen way.

# *Concerns*

Full SBPH/BCR may be too computationally expensive for widespread (at-ISP) implementation.

# *Concerns*

Full SBPH/BCR may be too computationally expensive for widespread (at-ISP) implementation.

Bayesian and other high-accuracy filters may become a tool for censorship or governmental monitoring

# *Concerns*

Full SBPH/BCR may be too computationally expensive for widespread (at-ISP) implementation.

Bayesian and other high-accuracy filters may become a tool for censorship or governmental monitoring

“It's a bad idea to implement good ideas that make it easy to do bad things”

# *Future Work*

- ♦ Integrate an 8-bit-safe approximate matching regex engine (probably TRElib, by Ville Laurikari)
- ♦ Better classifier (steal Spambayes chi-square evaluator)
- ♦ Advanced mail sorting (e.g. “parties”, “rants”, ”humor”, “flames”, “blather”, “spam”)
- ♦ Speed improvements
- ♦ Ease-of-installation issues

*Thank You All !!*

**Questions?**

CRM114 is licensed under the GPL and available for free download at the URL

<http://crm114.sourceforge.net>

it's still alpha-quality, but it works.