Using Naive Bayes to Detect Spammy Names in Social Networks

David Mandell Freeman
LinkedIn Corporation

AISec 2013
Berlin, Germany
4 November 2013

Social Networks:
You're Supposed to be You

Terms of Service of popular social networks:

4. **Registration and Account Security**

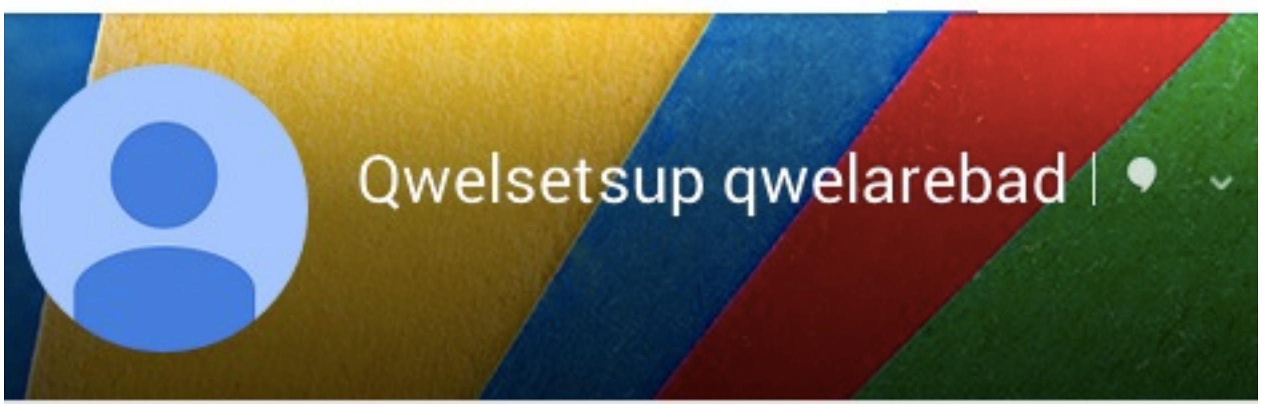
Facebook users provide their real names and information,

e; (5) will use your real name

power and authority to

use the name your friends, family or co-workers
usually call you when creating a Google+ profile. For exa
but you normally use Chuck Jones or Junior Jones, eithe

But Not Everyone Follows the Rules...



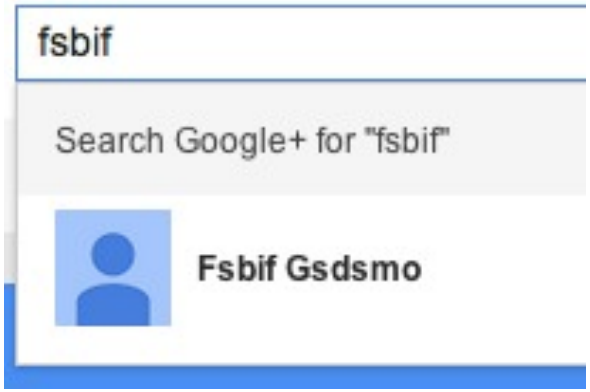
Why Sign up with a Fake Name?

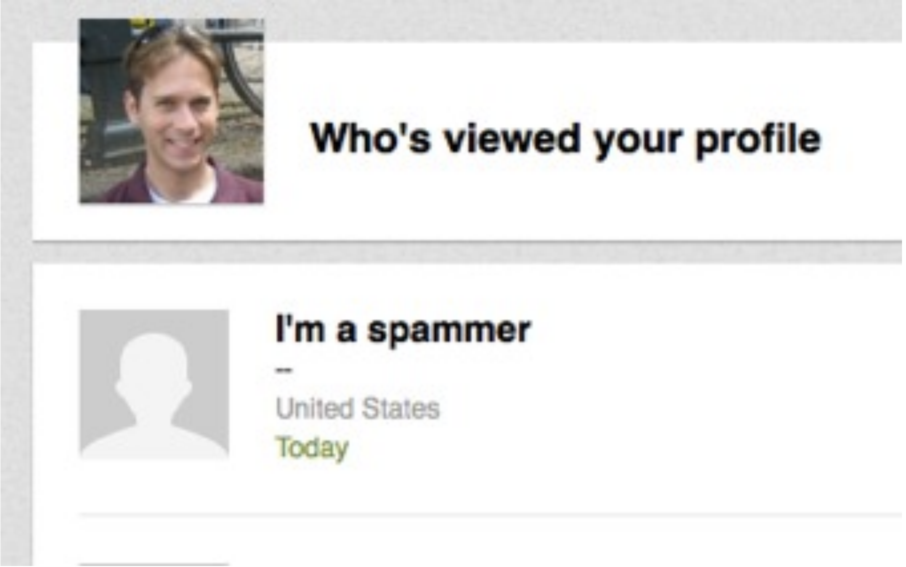
- **Malicious (human or automated):**
  - Scrapers/spammers
    - Dictionary of names
    - Random text generator
    - “Hack on the keyboard”
  - SEO
- **Non-malicious human**:
  - Lazy/secretive
    - Just type something to get through registration
  - Company name on personal page
  - Phone number or email in name field

Downstream Effects

Who cares if I enter a fake name if I'm not actively spamming?

Mistyped search for "david":





Conclusion: Fake names degrade the value of the site to real people.

Detecting Social Network Spam: Prior Work

LinkedIn

- **Clickstream patterns:**
  - Zhang-Paxson ’11: analyze timing of clicks
  - Wang *et al* ’13: cluster based on timing and page label
- **Message activity and content**:
  - Benvenuto *et al* ’10: statistics on URLs, spam words, hashtags
  - Gao *et al* ’10: scan content of Facebook wall posts
- **Social graph properties:**
  - Cao *et al* ’12: random walk on graph
  - Cao-Yang ’13: propagate negative feedback through graph

Our Contribution

- **Naive Bayes classifier to detect spam names from name text only**
  - Features: *n*-grams of letters
  - Extend feature set using phantom start/end chars
  - Several methods to handle missing features
- **Advantages**:
  - Can detect spammers at registration time
    - activity history and social graph are empty
  - Can classify names never seen before
    - large % of names are unique
  - Detects automated and human abusers
  - Detects malicious and non-malicious fakes

Multinomial Naive Bayes

- Supervised classification algorithm
- Assume features (usually words) chosen independently from multinomial distribution.
  - Feature random variable *X*, label random variable $Y \in \{0,1\}$
  - θ_{wy}= probability that word *w* appears in a sample from class *y*
  - f_w = multiplicity of word *w* in sample *x*

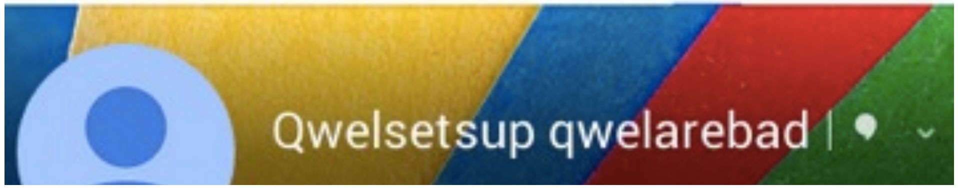
$$p(Y=1|X=\vec{x}) = \frac{1}{1+\frac{p(Y=0)}{p(Y=1)}e^{-R(\vec{x})}}, \quad \text{where} \quad R(x) = \sum_w f_w \log\left(\frac{\theta_{w1}}{\theta_{w0}}\right)$$

- To get probability estimate, need class priors $p(Y=y)$ and feature probabilities θ_{wy}.
- Use training data to estimate

$$\theta_{wy} = \frac{N_{w,y}+\alpha_{w,y}}{N_y+\sum_w \alpha_{w,y}} \qquad (N = \text{count},\ \alpha = \text{smoothing})$$

- Interpret probability estimate as a score.

Features: *n*-grams of letters



- Basic feature set (*n*=3):

  (*Qwe, wel, els, lse, set, ets, tsu, sup, qwe, wel, ela, lar, are, reb, eba, bad*)

- For better performance, consider first and last names independently:

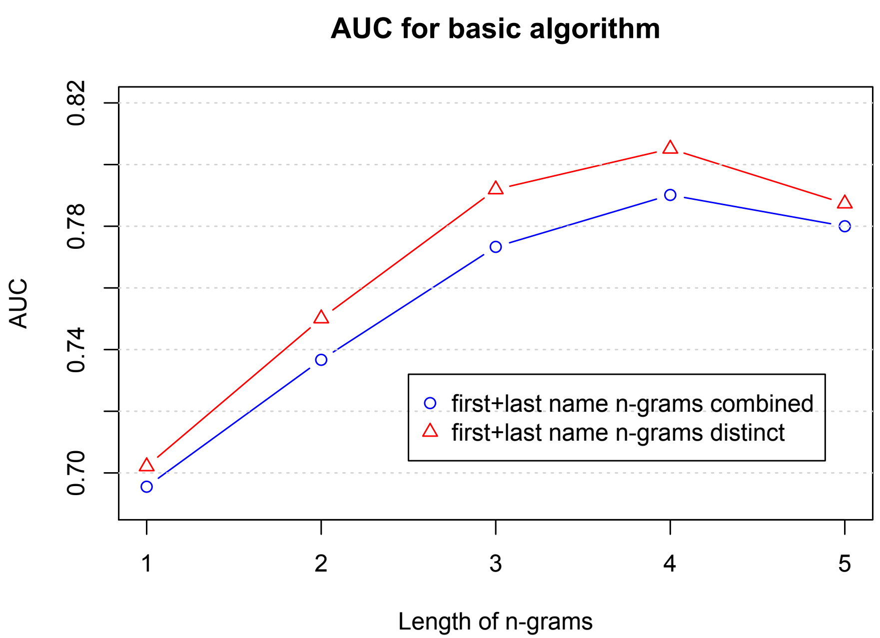
  (*Qwe, wel, els, lse, set, ets, tsu, sup*, *qwe, wel, ela, lar, are, reb, eba, bad*)

- Precompute *n*-gram frequencies for training set
  - Use entire Unicode alphabet
  - Ignore *n*-grams appearing only once in 60M accounts

	first/last distinct		first/last combined	
n	n-grams	memory	n-grams	memory
1	15,598	25 MB	8,235	24 MB
2	136,952	52 MB	86,224	45 MB
3	321,273	110 MB	252,626	108 MB
4	1,177,675	354 MB	799,985	335 MB
5	3,252,407	974 MB	2,289,191	803 MB

Training and Test Data

- **Training data:**
  - Unbiased sample of 60M LinkedIn accounts
  - Labels: 0 – flagged as fake/abusive by Security team
    1 – everyone else
- **Validation/test data**:
  - Sampled 200K accounts outside of training set
  - Biased to contain roughly equal numbers of good/bad accounts
- **Evaluation metric: AUC**
  - Doesn't require setting a classification threshold
  - Insensitive to bias in validation set



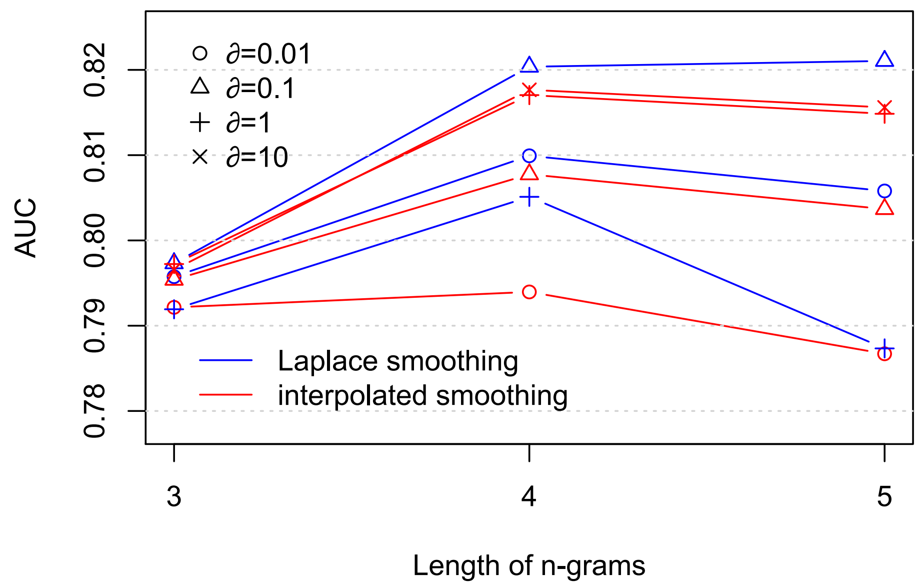
Adjusting the Smoothing Parameter

$$\theta_{wy} = \frac{N_{w,y} + \alpha_{w,y}}{N_y + \sum_w \alpha_{w,y}} \qquad (w = n\text{-gram},\ y = \text{class})$$

- *Smoothing parameter* $\alpha_{w,y}$ biases towards uniform
  - – prevent zero estimates in classes with no data
  - ○ *Laplace smoothing*: $\alpha_{w,y} = \delta$ (often $\delta = 1$)
  - ○ *Interpolated smoothing*: $\alpha_{w,y} = \delta / N_{w,y}$

- Tried $\delta \in (0.01, 0.1, 1, 10, 100)$ for both variants
- Little effect for $n \leq 3$
- Laplace smoothing works better for our dataset

AUC for various smoothing parameters



Using *n*-gram position

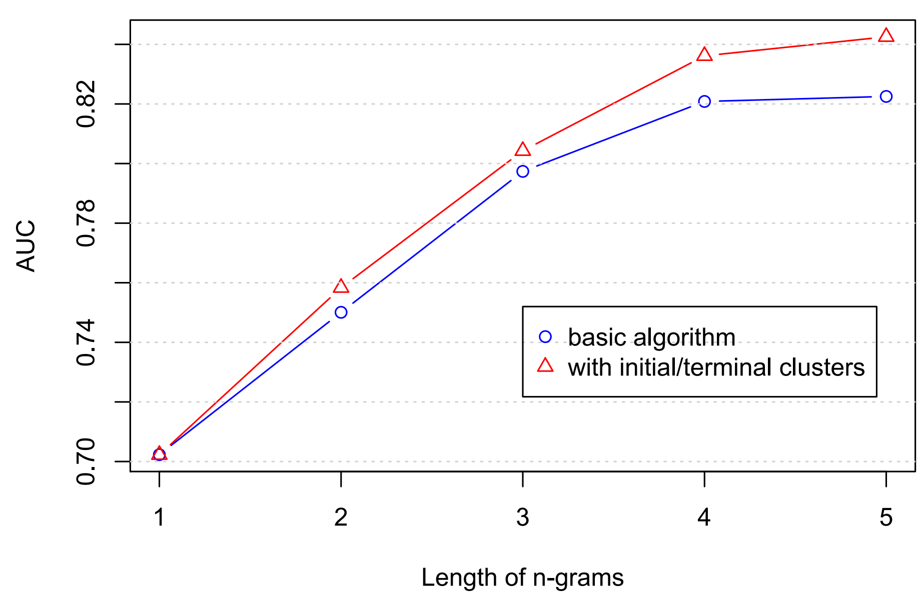
- Some *n*-grams are more or less likely to belong to spammers when at the start or end of a name
  - Capital letters, consonant clusters

    e.g.: ‘zz’ 13x more likely to be spammer if at start of name

- Insert “start-of-word” and “end-of-word” characters before parsing into *n*-grams:

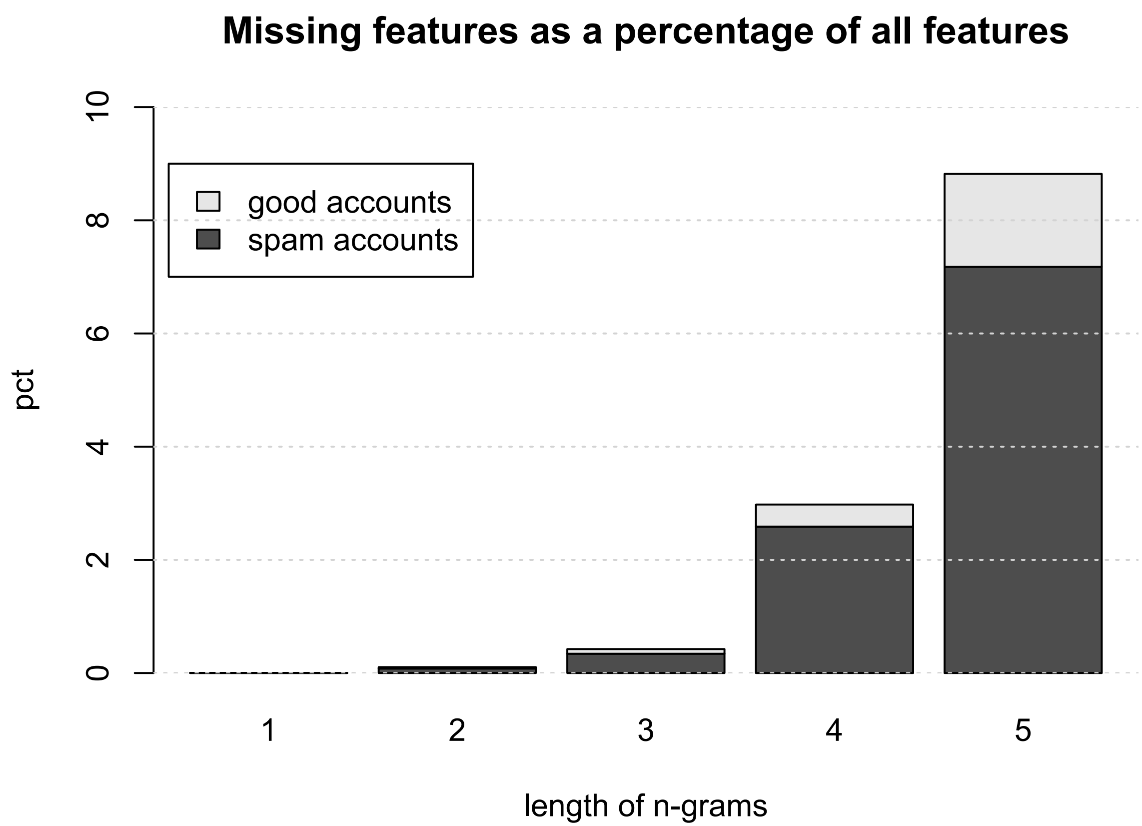
  (\^Qw, Qwe, wel, els, lse, set, ets, tsu, sup, up\$, \^qw, qwe, wel, ela, lar, are, reb, eba, bad, ad\$)

AUC for algorithm with initial/terminal clusters



Missing Features

- **Long tail of names:**
  Even with 60M training examples, many features in validation set are not present in training set.



- Explains lack of improvement from *n*=4 to *n*=5

Dealing with Missing Features (I)

Linkedin

n-gram	Qwe	wel	els	lse	set	ets	tsu	sup	qwe	wel	ela	lar	$\cdots$
$\log(\theta_{w1}/\theta_{w0})^*$	1.4	-0.6	0.8	$???$	-0.7	-0.5	0.6	-2.7	$???$	-3.1	-1.5	2.5	$\cdots$

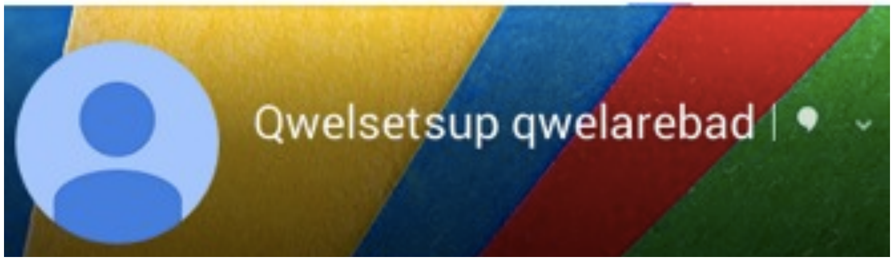
- Option 1: Ignore missing features
- Option 2: Compute parameter for "missing feature" feature (technique from NLP):
  1. Split data in two halves, $\mathcal{A} \cup \mathcal{B}$
  2. Label features that appear in only one half as "missing"
  3. Aggregate "missing feature" data to compute parameter

feature	$\mathcal{A}$ freqs	$\mathcal{B}$ freqs
v	$(8,3)$	$(3,4)$
w	$(2,1)$	
x	$(3,2)$	$(7,9)$
y	$(5,0)$	$(4,3)$
z		$(0,3)$
$miss$	$(2,4)$	

- Option 2 improves AUC for *n*=5 from 0.843 to 0.849
  - "missing feature" suggests spam

*not real data

Dealing with Missing Features (II)



Option 3: Use (*n*–1)-grams when *n*-gram data is missing:

n-gram	Qwe	wel	els	lse	set	ets	tsu	sup	qwe	wel	ela	lar	$\cdots$
$\log(\theta_{w1}/\theta_{w0})$	1.4	-0.6	0.8	???	-0.7	-0.5	0.6	-2.7	???	-3.1	-1.5	2.5	$\cdots$

ls	se
-0.6	-1.2

qw	we
???	-0.9

Iterate recursively:

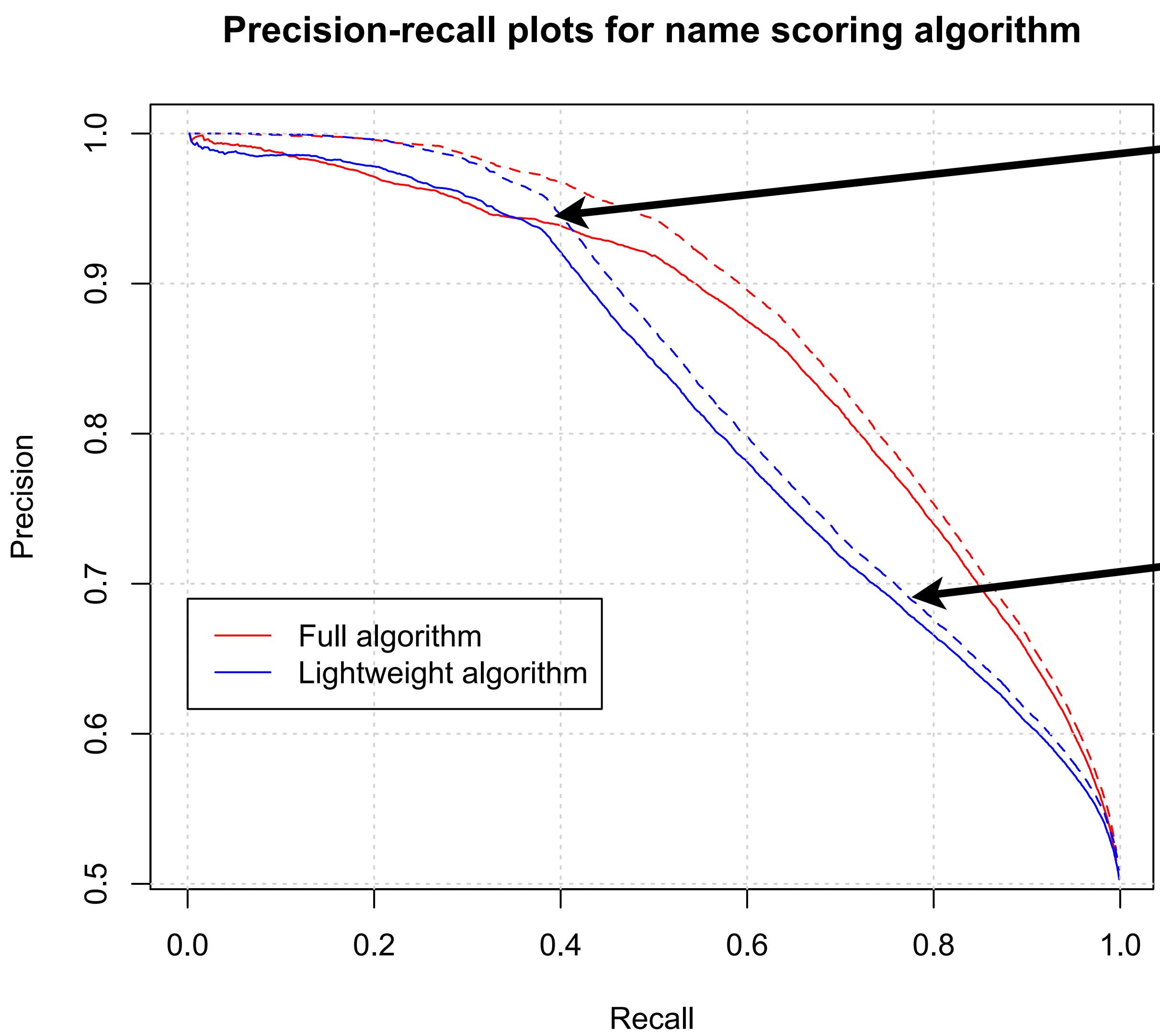
q	w
-0.4	0.5

- Recursive iteration on (*n*–1)-grams improves AUC for *n*=5 from 0.849 to 0.854

Evaluating Performance

	"Full" Algorithm	"Lightweight" Algorithm
n	5	3
smoothing	Laplace, ∂=0.1	Laplace, ∂=0.1
initial/terminal *n*-grams	yes	yes
missing *n*-grams	recursive (*n*–1)-grams	fixed estimate
AUC on test set	**0.852**	**0.803**

Evaluating Performance



Score cutoff ≈0.1 for full algorithm, ≈0.5 for lightweight algorithm

Dashed lines: test set corrected for mislabeled samples

False Positives

Manual review of test set accts with label 1 and score < 0.05

- 59% of “false positives” were incorrectly labeled.
- Precision increases from 95% to 98%.

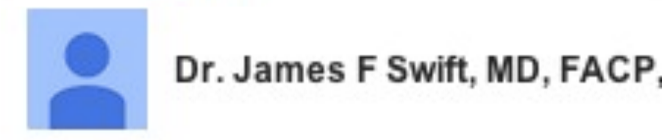
Patterns observed in false positives:

- Mixed-language names

- First/last name fields interchanged

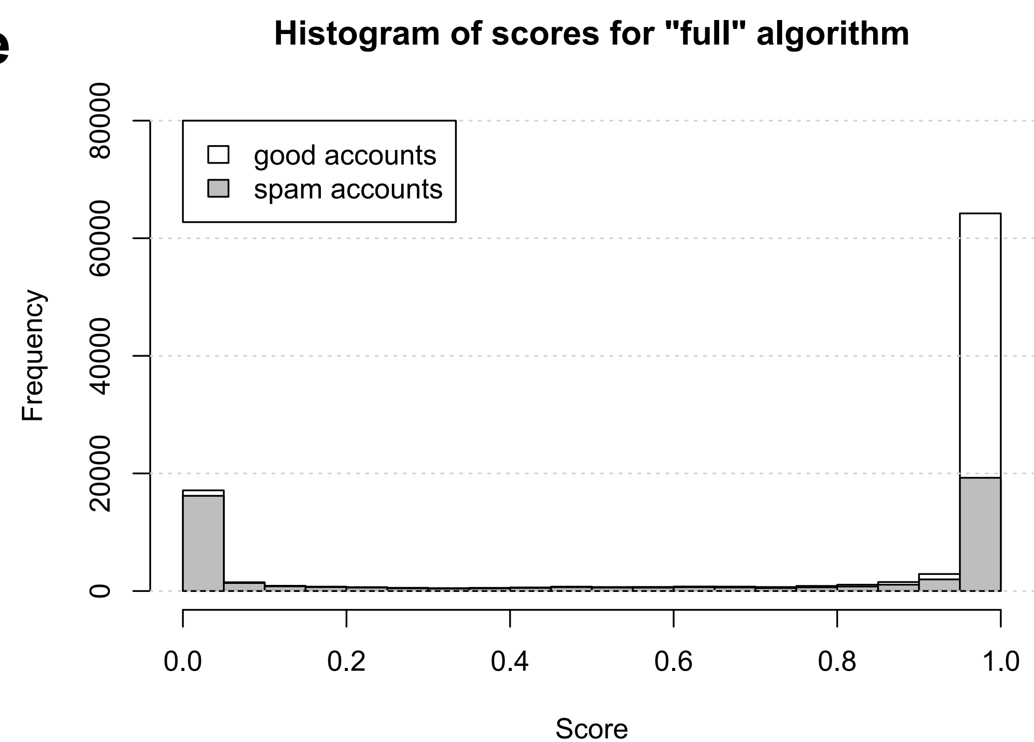
- Strange (but readable) characters

- Non-name information



False Negatives

- **Label 0 assigned to accounts marked as abusive for *any* reason — not just spam name**
  - Many spammers use real-looking names!
  - 40% of spam accounts, 91% of good accounts have scores > 0.95
- **Manually reviewed sample of accounts with label 0 and score > 0.65**
  - 93% did not have spammy names
  - Extrapolating this false negative rate to the whole test set doubles recall.

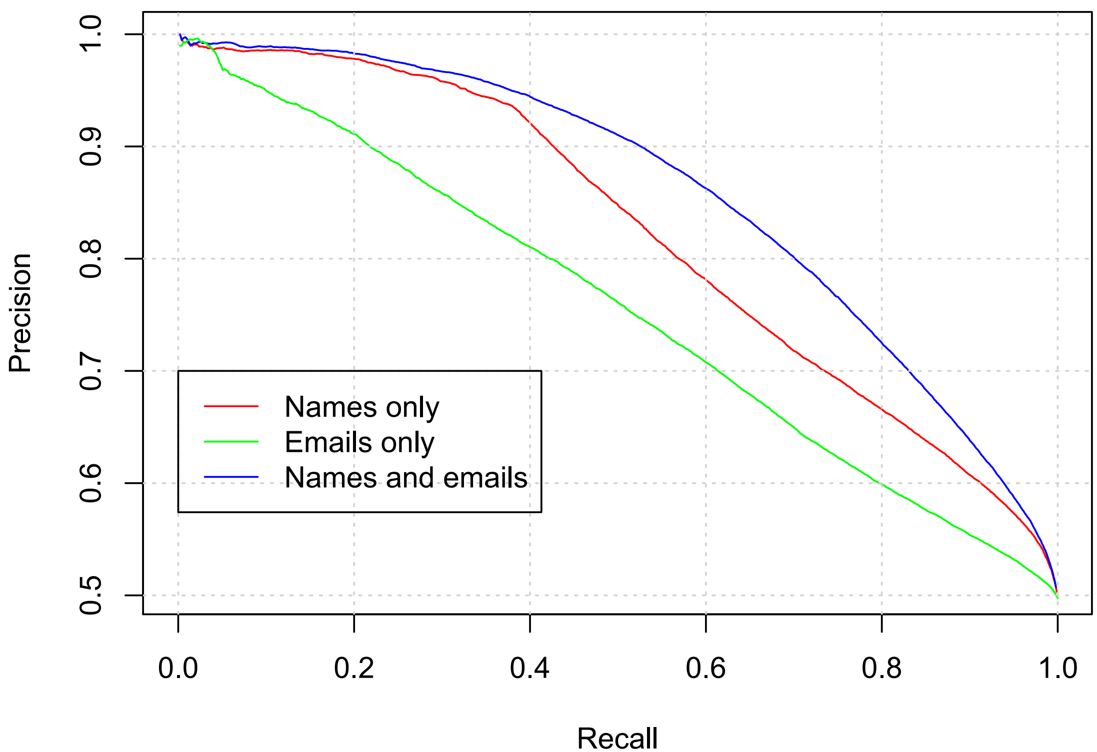


Scoring Email Addresses

- **Email usernames can also be scored using our algorithm**
  - Short texts with even greater diversity than names.
  - Spammers make less effort to have non-spammy email address.
  - Lazy user may type in gibberish to get past registration screen.

- **Scored emails alone and emails along with names**
  - Emails help distinguish spammers in border-line cases

Precision-recall plots for name/email scores



Further Directions

- **Reduce false positive rate**
  - Mixed-language names: parse and score separately
  - Switched name fields: score on alternate permutation; use weighted score.
  - Unusual characters: map to a “reduced” character set.
  - Non-name information: match to a list or improve UI.
- **Strengthen adversarial model**
  - Continuous training
- **Other ideas?**
  - Work with the LinkedIn Security Data Science team!

    — full-time, internships, collaborations
  - email **dfreeman@linkedin.com**

Thank you!