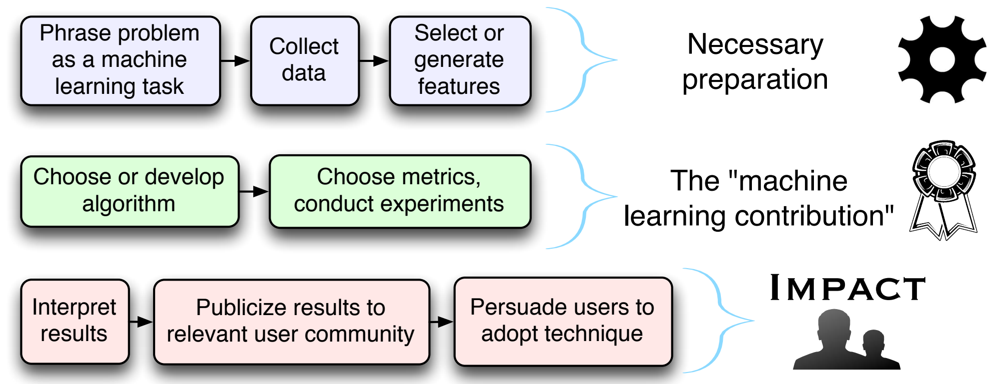
Machine Learning for Data Exploration

Charles Sutton
University of Edinburgh and the Alan Turing Institute
23 January 2017

Prediction: A small part of a big picture

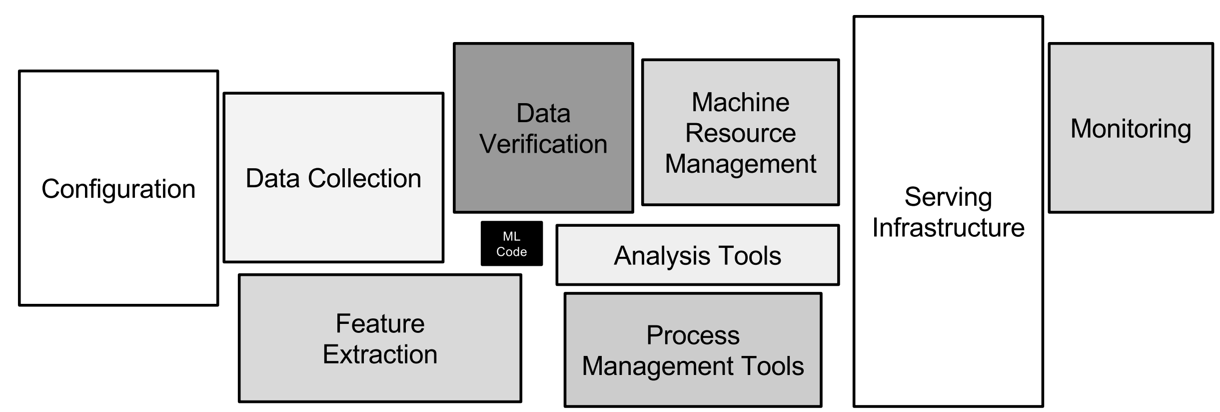
CRISP-DM

[Chapman et al 2000]



Research process

[Wagstaff, ICML 2012]



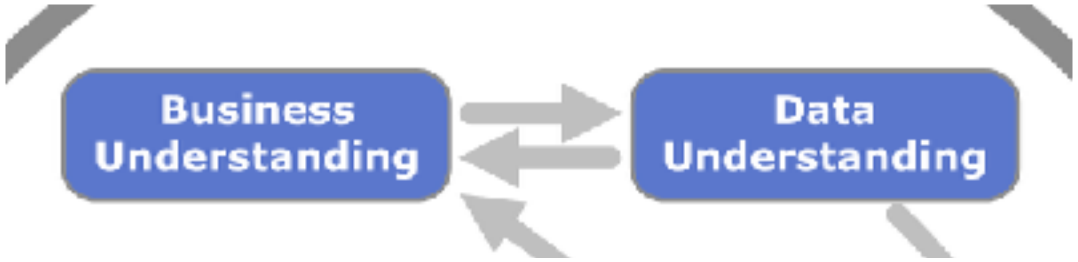
System level

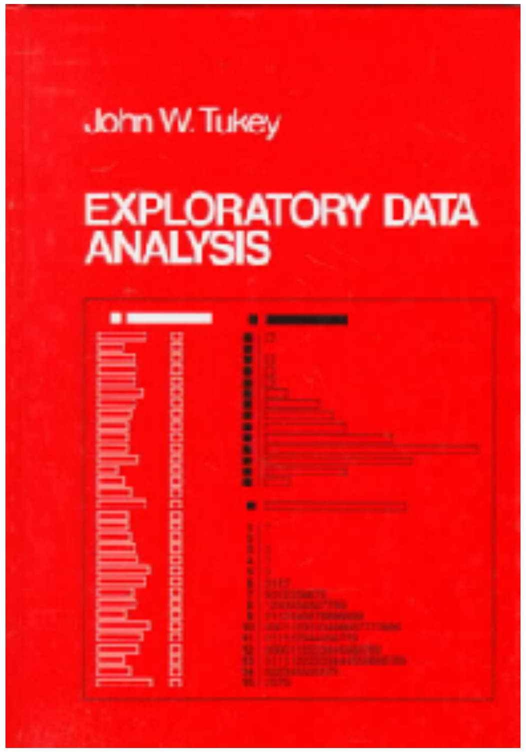
[Sculley et al, NIPS 2015]

"Making data science easier": an application area for machine learning!

Towards an Artificial Intelligence for Data Science

Data understanding





When you get a new data set….

- What's in it?
- What's wrong with it?
- What should I do with it?

Automating exploratory data analysis?

Contradiction in terms?

A task in visual analytics

Scalability a challenge

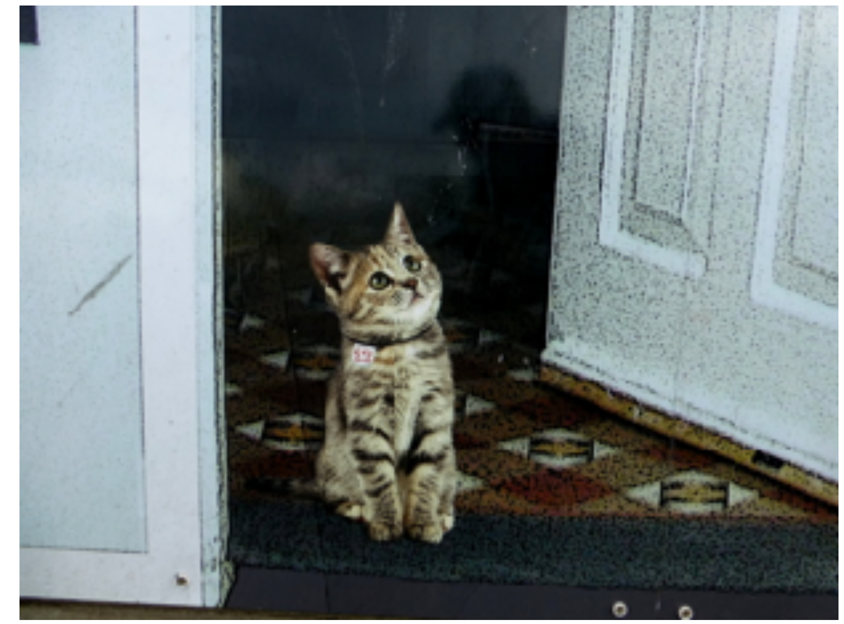
Our theme

Summarise data with probabilistic ML

Visualize resulting patterns

Patterns are "first class citizens" of model

Exploratory data analysis



Data analysts are like cats.

1. Want to explore their data
2. Don't know what they want.

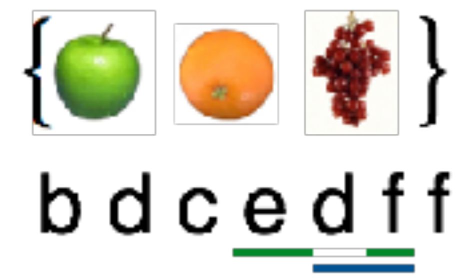
Machine learning for analysts

Whose information need is not explicit

Whose domain knowledge is difficult to encode

Explore data via learned patterns

… not just for dummies!

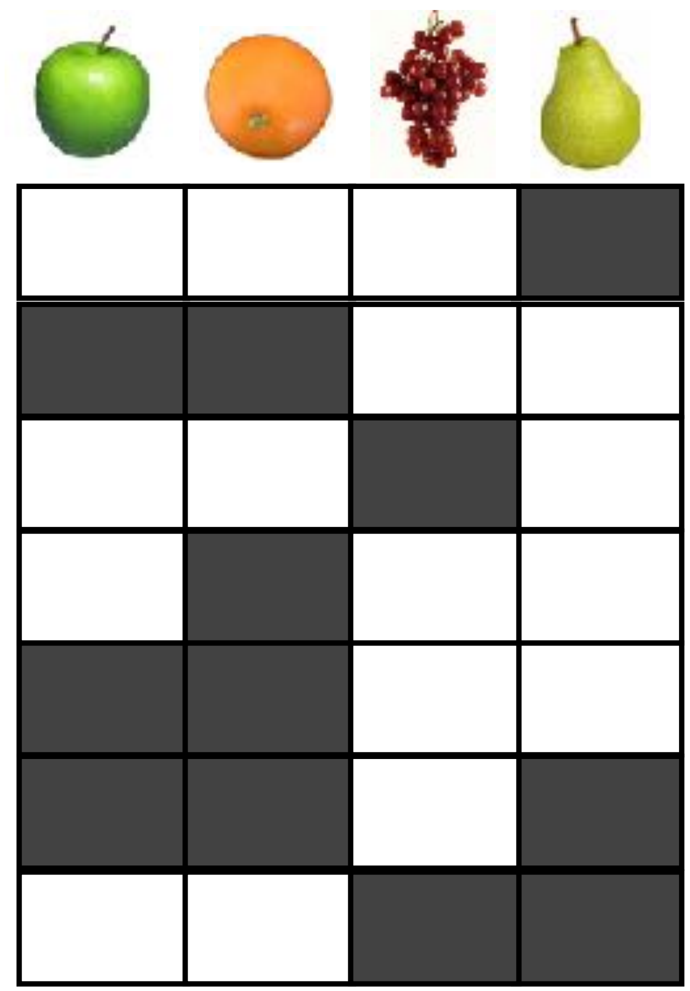


Mining Patterns

[Fowkes & Sutton, KDD 2016]

[Fowkes & Sutton, PKDD 2016]

Association Rules (def'n)



Database of transactions

Association rule mining:

Find set of all rules

that have

Prob  $\geq \alpha$

Count  $\geq M$

Frequent itemsets:

Count  $\geq M$

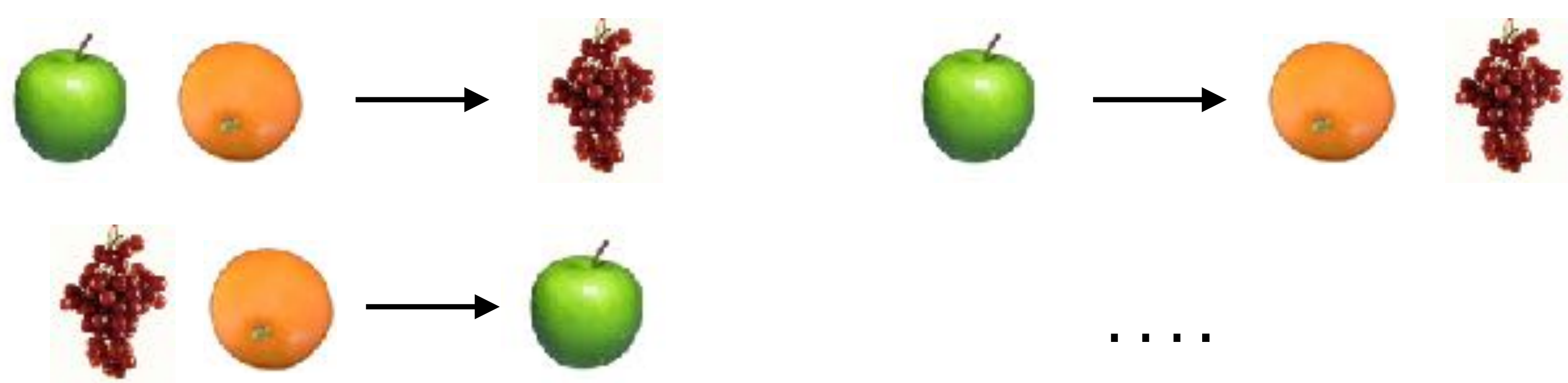
Why? Exploratory data analysis

Association Rules (alg)

1. Identify all frequent item sets

via exhaustive search (APriori, FP-Growth, etc.)

2. For each item set, consider all possible partitions



3. Rank the resulting list (e.g., by confidence) and enjoy

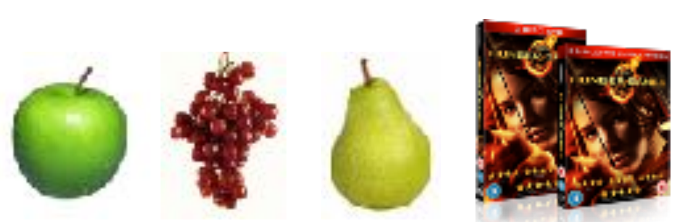
Pathologies

List of association rules unwieldy, difficult to understand

Procedure as a whole is statistically incoherent.

Essentially just repeated counting

Redundancy

If {  } frequent,

so are all 14 nontrivial subsets.
(Association rules "filter" item sets)

"Free riders"

If both of these

{  } {  }

have support >> M and **independent**

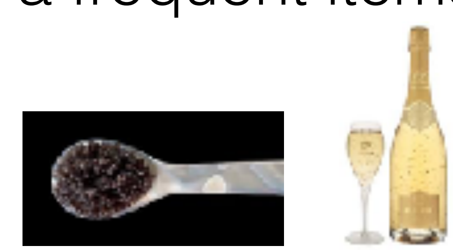
{  }

usually still support > M

(Confidence and lift do not fix this!)

Rare itemsets

Strongly associated but rare:
Not a frequent itemset

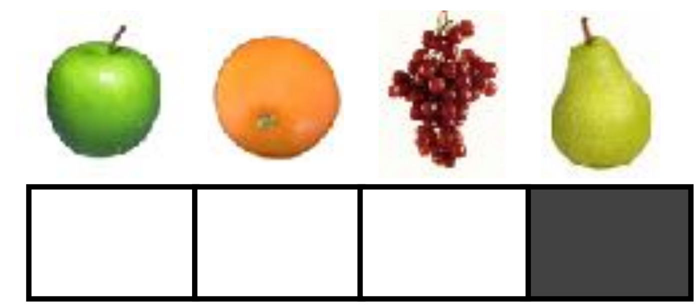
{  }

e.g., champagne, caviar

[Hastie et al., 2009]

Alternative: *Interesting* Itemsets

Optimise the collection of itemsets *as a whole,* rather than each in isolation

transaction X 

define probability model

$$p(X|\mathcal{I})$$

e.g., $\mathcal{I} = \{\{$  $\},\{$  $\}\}$

choose $\mathcal{I}$ to best fit data

$\mathcal{I}$ are the *interesting* itemsets

(unlike frequent itemsets, these are suitable for data analysis)

Interesting Sequence Mining

define a goodness measure on a *set* of patterns

Minimum description length

[Vreeken et al, 2011; Tatti and Vreeken, 2012; Lam et al 2014]

Use patterns to define a compression algorithm for database

Search for patterns that best compress

Probabilistic methods

[Fowkes and Sutton, KDD 2016, PKDD 2016]

Use patterns to define a probability distribution over database

Search for patterns that maximise database probability

(actually isomorphic; see MacKay, 2003)

Sequences more meaningful, less redundant

Also, see tiling: *[Geerts, Goethals, and Mielikäinen, 2004]*

Model

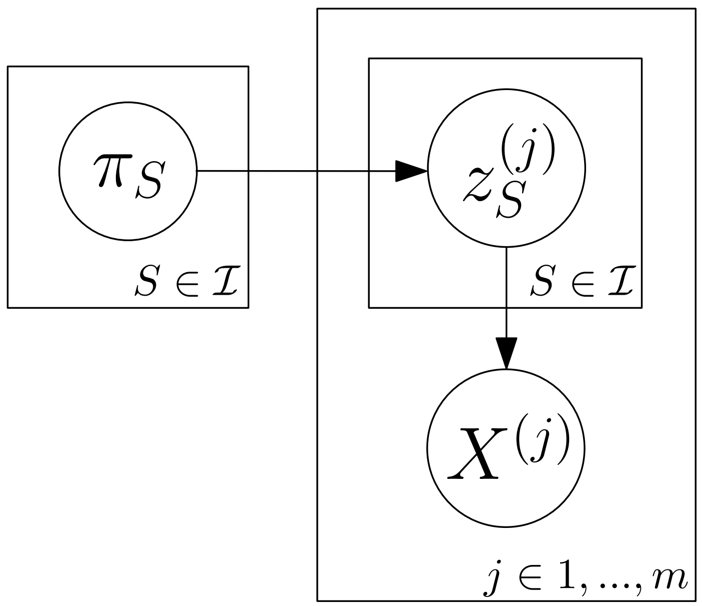
To sample a transaction,

1. For each itemset, sample

$$z_S \sim \text{Bernoulli}(\pi_S).$$

2. Deterministically set

$$X = \bigcup_{z_s = 1} S.$$



Parameters:

$\mathcal{I}$ Collection of "interesting" itemsets

$\pi_S \in [0, 1]$ for each $S \in \mathcal{I}$

probability of occurrence

Inference / Learning

Infer z from X

$$\max_{\mathbf{z}} \sum_{S \in \mathcal{I}} z_S \ln\left(\frac{\pi_S}{1-\pi_S}\right) + \ln(1-\pi_S)$$
$$\text{s.t.} \sum_{S | i \in S} z_S \geq 1 \qquad \forall\, i \in X$$
$$z_S \in \{0,1\} \qquad \forall\, S \in \mathcal{I}$$

NP-hard but submodular
(weighted set cover)
use greedy algorithm

Infer $\mathcal{I}$

Structural EM

Propose new itemset S

Add S to model

Re-infer $\mathcal{Z}$

Check if cost improved

"Implicit regularization"

Redundancy

Average distance between itemsets in one ranked list

(symmetric distance, higher is better)

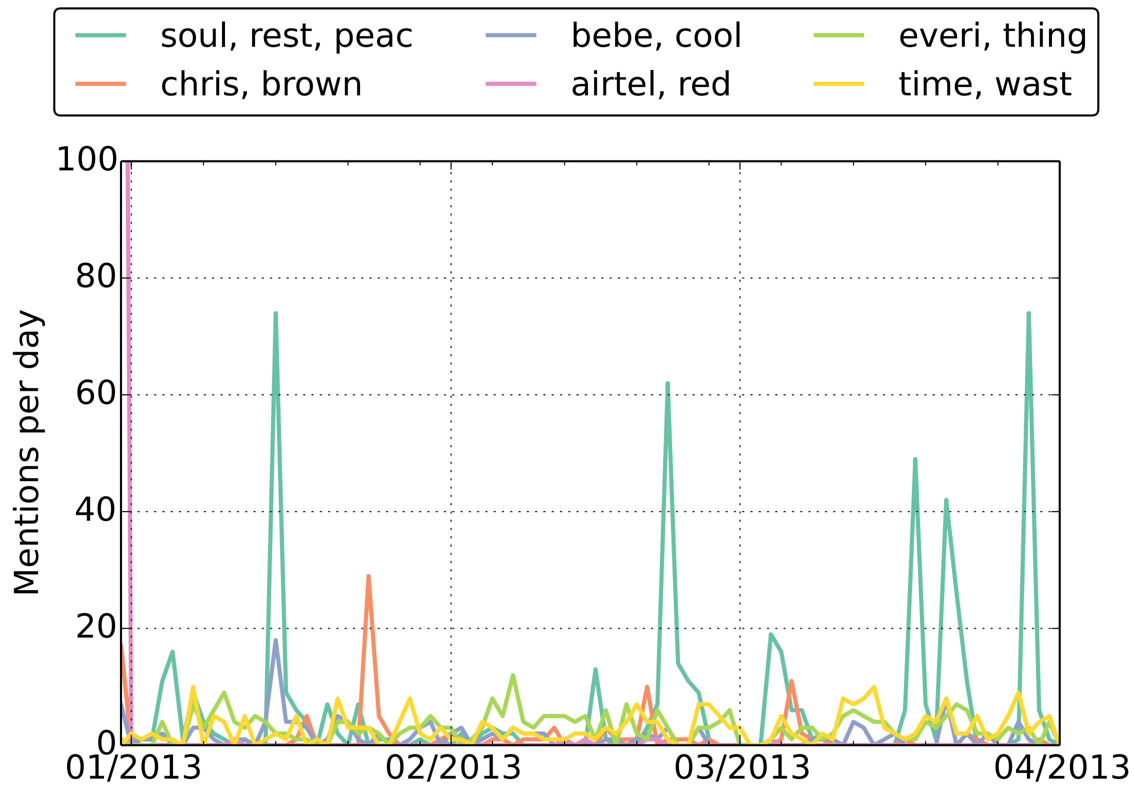
	Plants	Mammals	ICDM	Uganda
Interesting Itemsets	**3.50**	**5.30**	**3.66**	**3.72**
KRIMP	**1.53**	**2.02**	**2.22**	**2.24**
CHARM	**1.53**	**1.52**	**1.47**	**1.45**

Facebook posts

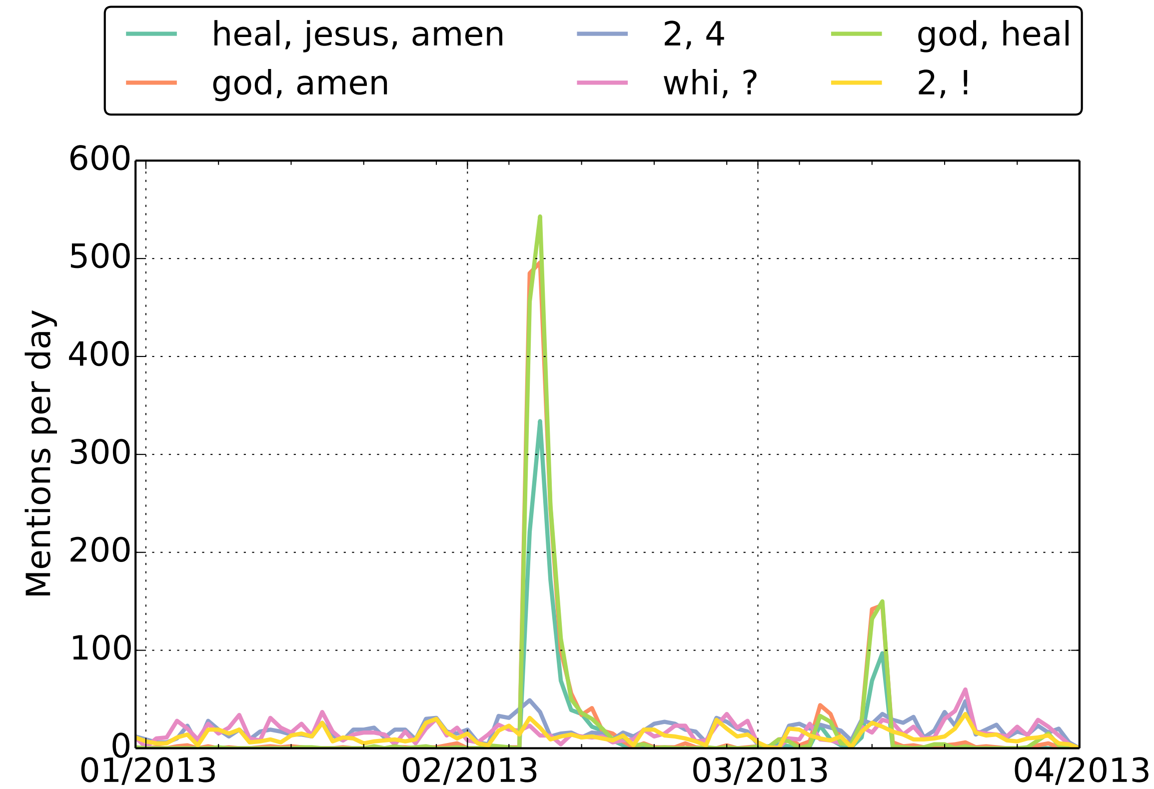
IIM	MTV	KRIMP
soul, rest, peace	heal, jesus, amen	whi, ?
chris, brown	god, amen	?, !
bebe, cool	2, 4	2, 4
airtel, red	whi, ?	wat, ?
everi, thing	god, heal	time, !
time, wast	2, !	soul, rest, peace

[courtesy John Quinn, UN Global Pulse]

Trending



IIM



MTV

Facebook posts from public Ugandan pages

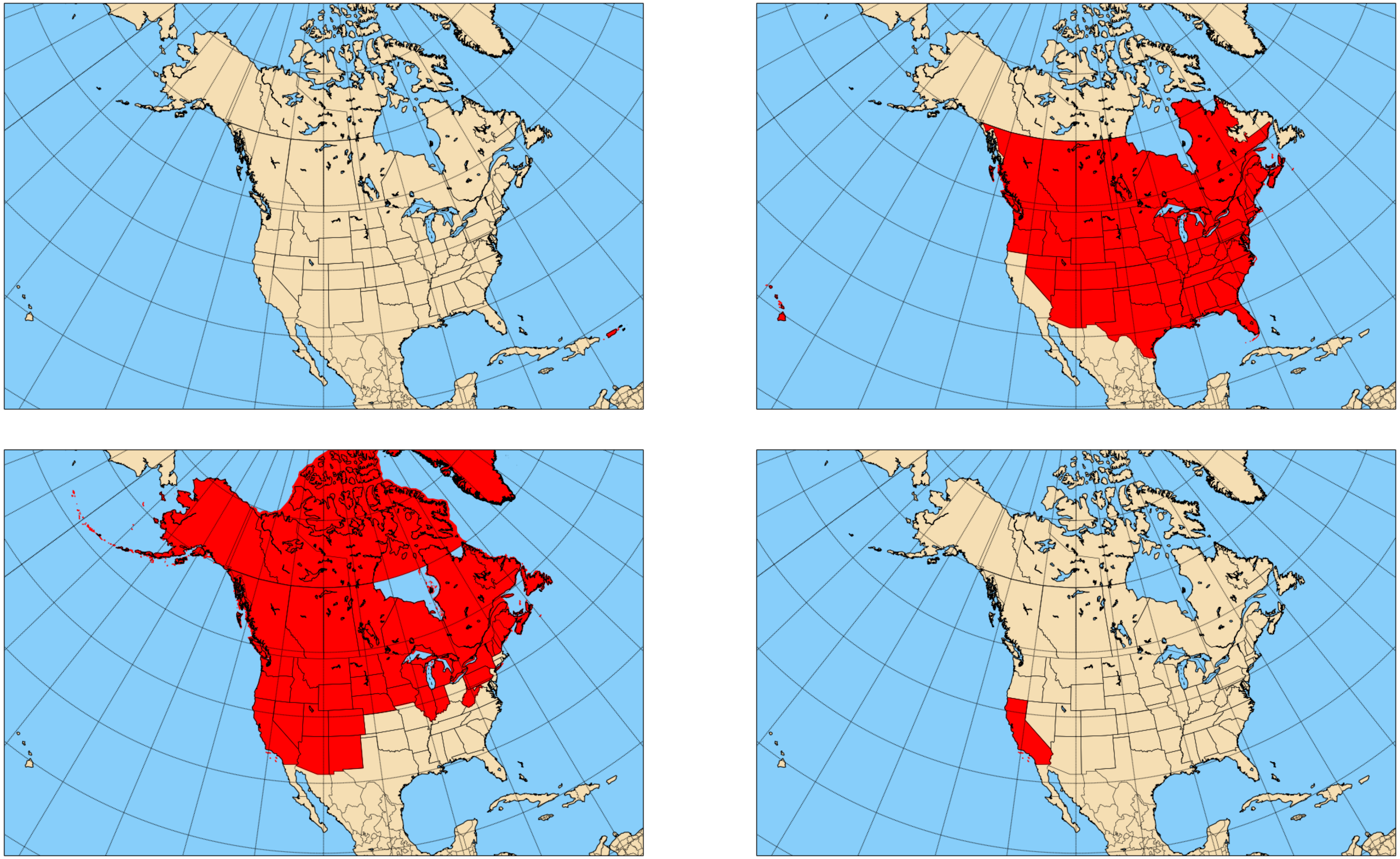
[courtesy John Quinn, UN Global Pulse]

Plants

KRIMP

Plants

IIM



Frequent Sequence Mining

Return all patterns with >= given support

Support of pattern: Number of database sequences that contain it

[Agrawal and Srikant, 1995; Wang and Han, 2004]

b d b a f e c
b c e a
e d a f c
a e f b
b d a e f c

Database of sequences

d a f c
b a f c
a e
b e
e c
...

Sequence patterns

(*e.g.* minimum support = 3)

Problem: Frequent can be trivial!

Fundamental Pathologies

Truncation

d a f c → a c

Real pattern

Could be returned (more frequent!)

Spurious correlation

Support(a) = 90%

Support(d) = 90%

… but independent …

→ d a

Pattern at 81% min_support

Effect: Redundant list of patterns

Freerider

a f c real pattern

Support(d) = 90%

… but independent …

→ a d f c

for high enough min_support

Probabilistic Sequence Mining

[Fowkes and Sutton, KDD 2016]

Define a distribution P(database | patterns)

$\mathcal{I}$

[b c e] : 0.1, **0.6**
[d f] : 0.7, **0.3**
[d f] : 0.8, **0.2**
[e f] : **0.8**, 0.1

Sequence patterns
(with probabilities)

Sample

Inclusion variables:

z_1: **1**
z_2: **1**
z_3: **1**
z_4: **0**

Interleave randomly

X b d c e d f f

Sampled database sequence

$P(X, z|\mathcal{I})$

probability of generating X, z from this process

Probabilistic Sequence Mining

[Fowkes and Sutton, KDD 2016]

Model:

$$p(X, \mathbf{z}|\boldsymbol{\Pi}) = \frac{1}{|\mathcal{P}|} \prod_{S\in\mathcal{I}} \prod_{m=0}^{|\boldsymbol{\pi}_S|-1} \pi_{S_m}^{[z_S=m]}$$

Inference: Determine $z|X, \mathcal{I}$

X b d c e d f f

cover these?

or these?

$\mathcal{I}$

[b c e] : 0.1, 0.6
[d f] : 0.7, 0.3
[d f] : 0.8, 0.2
[e f] : 0.8, 0.1

Use greedy algorithm to

$$\max_z \log p(z|X, \mathcal{I})$$

(extension of weighted set cover)

Probabilistic Sequence Mining

[Fowkes and Sutton, KDD 2016]

Output of inference

	z [b c e]	[d f]	[d f]	[e f]
b d c e d f f	1	1	1	0
e e d f f f	0	1	0	1
d f d d f f	0	1	1	1

Learning step: Infer $\mathcal{I}$

Update probabilities
(average of z)

Propose new patterns
Add to model
See if probability increases

$\mathcal{I}$

[b c e] : 0.3, 0.7
[d f] : 0.0, 1.0
[d f] : 0.7, 0.3
[e f] : 0.3, 0.7

Formally: Structural Expectation Maximization

Application to Software Engineering

[Fowkes & Sutton, FSE 2016]

Modern development is layers of libraries

Average Java file on Github:

Imports from **2.1** packages outside project

45% of files import an external package

(Not counting `java.*` `javax.*` `sun.*`)

Github Java corpus (Allamanis and Sutton, 2013)

13000+ projects with at least one fork, 2M+ Java files

http://groups.inf.ed.ac.uk/cup/javaGithub/

(heuristic analysis)

API Mining

[Zhong et al, 2009; Dang et al 2013]

Library

e.g. TWITTER4J

Coding

```
private FinchTwitterFactory(Context context) {
    mContext = context;

    installHttpResponseCache();

    ConfigurationBuilder configurationBuilder = new ConfigurationBuilder();
    configurationBuilder.setOAuthConsumerKey(ConsumerKey.CONSUMER_KEY);
    configurationBuilder.setOAuthConsumerSecret(ConsumerKey.CONSUMER_SECRET);
    configurationBuilder.setUseSSL(true);
    Configuration configuration = configurationBuilder.build();
    mTwitter = new TwitterFactory(configuration).getInstance();
}
```

brk3 / finch

```
public Twitter getTwitterInstance() {
    ConfigurationBuilder cb = new ConfigurationBuilder();
    cb.setOAuthConsumerKey(Keys.consumerKey);
    cb.setOAuthConsumerSecret(Keys.consumerSecret);
    cb.setOAuthAccessToken(mSettings.getString("accessToken", null));
    cb.setOAuthAccessTokenSecret(mSettings.getString("accessSecret", null));
    TwitterFactory tf = new TwitterFactory(cb.build());
    return tf.getInstance();
}
```

jrupac/CleanTwitter

```
private void startOAuth() {
    ConfigurationBuilder configurationBuilder = new ConfigurationBuilder();
    configurationBuilder.setOAuthConsumerKey(Const.CONSUMER_KEY);
    configurationBuilder.setOAuthConsumerSecret(Const.CONSUMER_SECRET);
    twitter = new TwitterFactory(configurationBuilder.build()).getInstance();

    try {
        requestToken = twitter.getOAuthRequestToken(Const.CALLBACK_URL);
        Toast.makeText(this, "Please authorize this app!", Toast.LENGTH_LONG).show();
        this.startActivity(new Intent(Intent.ACTION_VIEW,
            Uri.parse(requestToken.getAuthenticationURL() + "&force_login=true")));
    } catch (TwitterException e) {
        e.printStackTrace();
    }
}
```

katahirado/tsubunomi

Corpus of client code

API Mining

```
ConfigurationBuilder.<init>
ConfigurationBuilder.setOAuthConsumerKey
ConfigurationBuilder.setOAuthConsumerSecret
ConfigurationBuilder.build
TwitterFactory.<init>
TwitterFactory.getInstance
```

```
TwitterFactory.<init>
TwitterFactory.getInstance
Twitter.setOAuthConsumer
Twitter.setOAuthAccessToken
```

API patterns

Documentation Suggestion

Frequent Sequence Mining

Each transaction: client method

Each element: a method call to an API method

b d b a f e c

b c e a

e d a f c

a e f b

b d a e f c

Database of sequences

d a f c

b a f c

a e

b e

e c

...

Sequence patterns

(*e.g.* minimum support = 3*)*

For API Mining…

Top 10 API patterns from pure sequence mining (BIDE)

```
TwitterFactory.<init>
TwitterFactory.getInstance
```

```
TwitterFactory.<init>
Twitter.setOAuthConsumer
```

```
Status.getUser
Status.getText
```

```
auth.AccessToken.<init>
Twitter.setOAuthAccessToken
```

```
TwitterFactory.<init>
TwitterFactory.getInstance
Twitter.setOAuthConsumer
Twitter.setOAuthAccessToken
```

```
TwitterFactory.getInstance
Twitter.setOAuthConsumer
```

```
TwitterFactory.<init>
TwitterFactory.getInstance
Twitter.setOAuthConsumer
```

```
TwitterFactory.<init>
Twitter.setOAuthAccessToken
```

```
TwitterFactory.<init>
TwitterFactory.getInstance
Twitter.setOAuthAccessToken
```

```
TwitterFactory.getInstance
Twitter.setOAuthAccessToken
```

```
TwitterFactory.<init>
Twitter.setOAuthConsumer
Twitter.setOAuthAccessToken
```

Previous Approach: Cluster before/after

[Zhong et al, 2009; Dang et al 2013]

Probabilistic API Miner (PAM)

Interesting sequence mining for API mining

```
ConfigurationBuilder.<init>
ConfigurationBuilder.setOAuthConsumerKey
ConfigurationBuilder.setOAuthConsumerSecret
ConfigurationBuilder.setUseSSL
ConfigurationBuilder.build
TwitterFactory.<init>
TwitterFactory.getInstance
```

```
ConfigurationBuilder.<init>
ConfigurationBuilder.setOAuthConsumerKey
ConfigurationBuilder.setOAuthConsumerSecret
ConfigurationBuilder.setOAuthAccessToken
ConfigurationBuilder.setOAuthAccessTokenSecret
ConfigurationBuilder.build
TwitterFactory.<init>
TwitterFactory.getInstance
```

```
ConfigurationBuilder.<init>
ConfigurationBuilder.setOAuthConsumerKey
ConfigurationBuilder.setOAuthConsumerSecret
ConfigurationBuilder.build
TwitterFactory.<init>
TwitterFactory.getInstance
TwitterFactory.getOAuthRequestToken
RequestToken.getAuthenticationURL
```

Sequence database

```java
private FinchTwitterFactory(Context context) {
    mContext = context;

    installHttpResponseCache();

    ConfigurationBuilder configurationBuilder = new ConfigurationBuilder();
    configurationBuilder.setOAuthConsumerKey(ConsumerKey.CONSUMER_KEY);
    configurationBuilder.setOAuthConsumerSecret(ConsumerKey.CONSUMER_SECRET);
    configurationBuilder.setUseSSL(true);
    Configuration configuration = configurationBuilder.build();
    mTwitter = new TwitterFactory(configuration).getInstance();
}
```

brk3 / finch

```java
public Twitter getTwitterInstance() {
    ConfigurationBuilder cb = new ConfigurationBuilder();
    cb.setOAuthConsumerKey(Keys.consumerKey);
    cb.setOAuthConsumerSecret(Keys.consumerSecret);
    cb.setOAuthAccessToken(mSettings.getString("accessToken", null));
    cb.setOAuthAccessTokenSecret(mSettings.getString("accessSecret", null));
    TwitterFactory tf = new TwitterFactory(cb.build());
    return tf.getInstance();
}
```

jrupac/CleanTwitter

```java
private void startOAuth() {
    ConfigurationBuilder configurationBuilder = new ConfigurationBuilder();
    configurationBuilder.setOAuthConsumerKey(Const.CONSUMER_KEY);
    configurationBuilder.setOAuthConsumerSecret(Const.CONSUMER_SECRET);
    twitter = new TwitterFactory(configurationBuilder.build()).getInstance();

    try {
        requestToken = twitter.getOAuthRequestToken(Const.CALLBACK_URL);
        Toast.makeText(this, "Please authorize this app!", Toast.LENGTH_LONG).show();
        this.startActivity(new Intent(Intent.ACTION_VIEW,
            Uri.parse(requestToken.getAuthenticationURL() + "&force_login=true")));
    } catch (TwitterException e) {
        e.printStackTrace();
    }
}
```

katahirado/tsubunomi

Corpus

Probabilistic sequence mining

Data

Target projects: 17 Java libraries, all that:

- Library source on Github
- Library in top 1000 Github projects
- Called by >50 other methods on Github
- At least 10k lines of `example/` code
- Total: Over 300k lines of example code

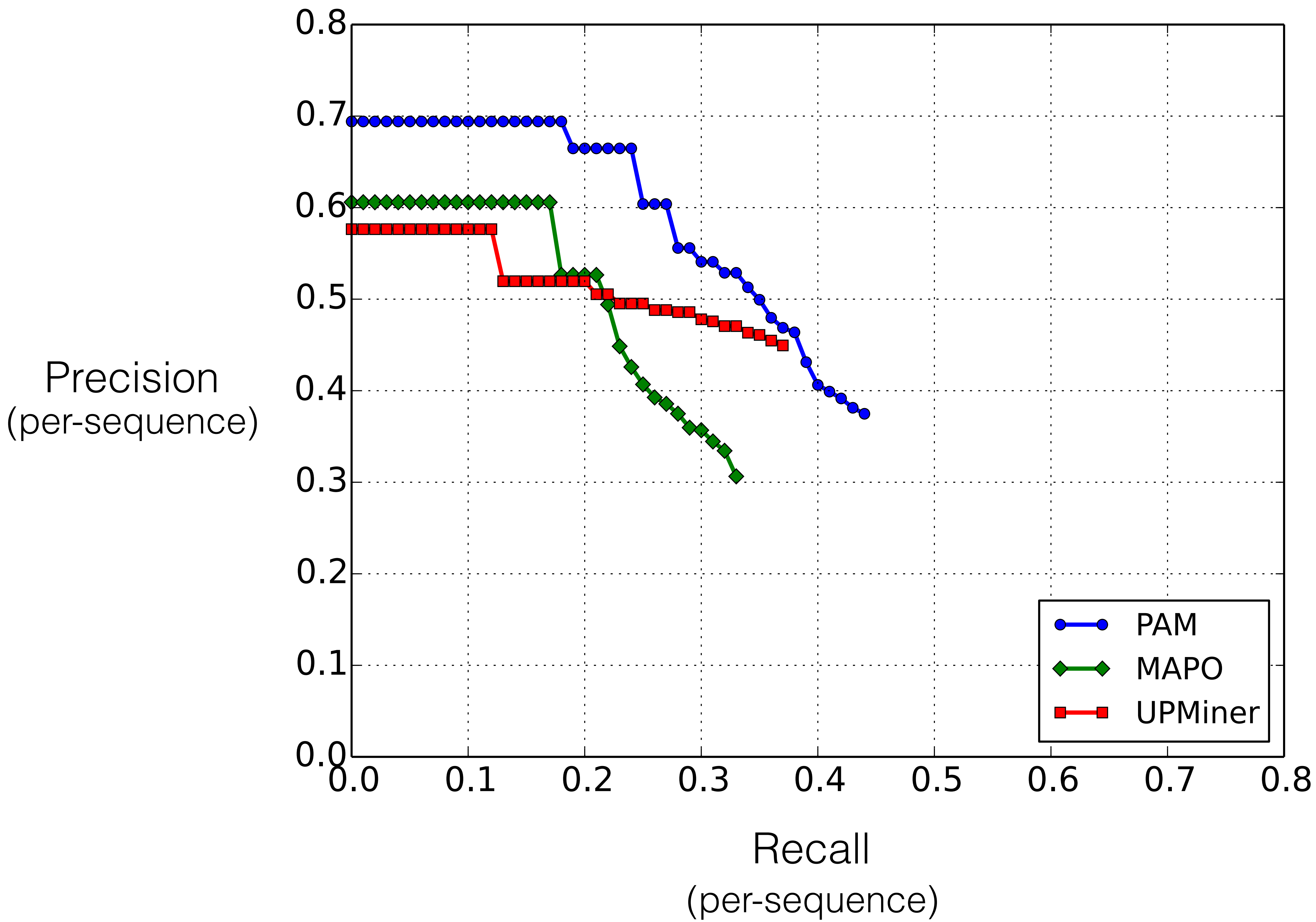
Client methods: all that called any targets

- 967 client projects
- Total: Over 4M lines of client code

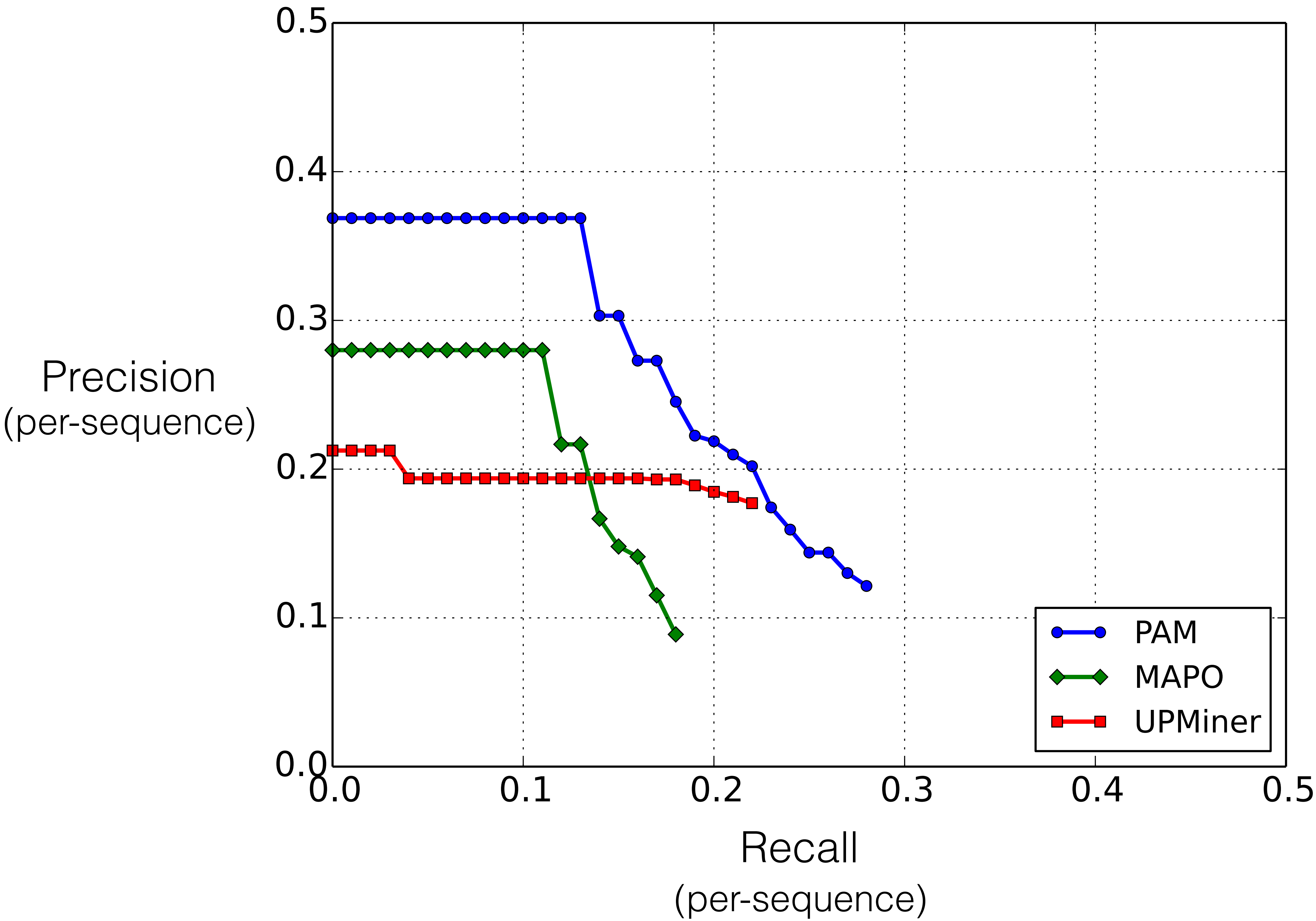
Experimental Questions

- Quality
  - Match to “held-out” client code
  - Match to examples from library developers
  - Measure: sequence overlap, precision, recall
- Redundancy
  - Why? Ease of use, diversity
  - Measure: number of containing sequences
- All results averaged over the 17 libraries

Prevalence in Client Code



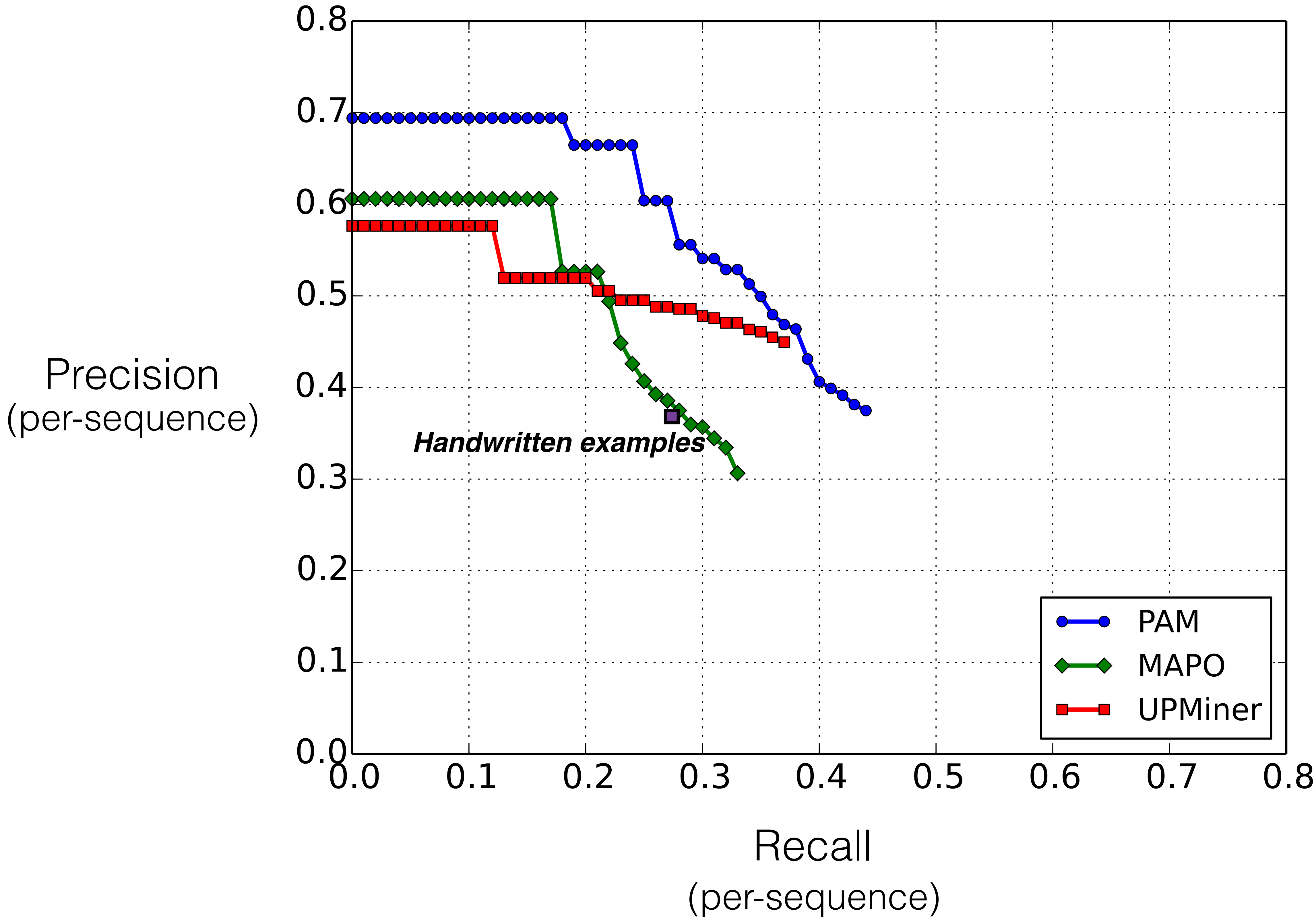
Handwritten Examples



Why Low Recall?

- API mining bad, or examples incomplete?
  - Match test set to examples: is test set covered?
  - 73% of client API sequences not covered
  - 36% of examples used in client code
- Manual error analysis
  - 3 random projects, top 5 unmatching patterns
  - 7 referred to API method not in examples
  - 3 referred to API class not in examples

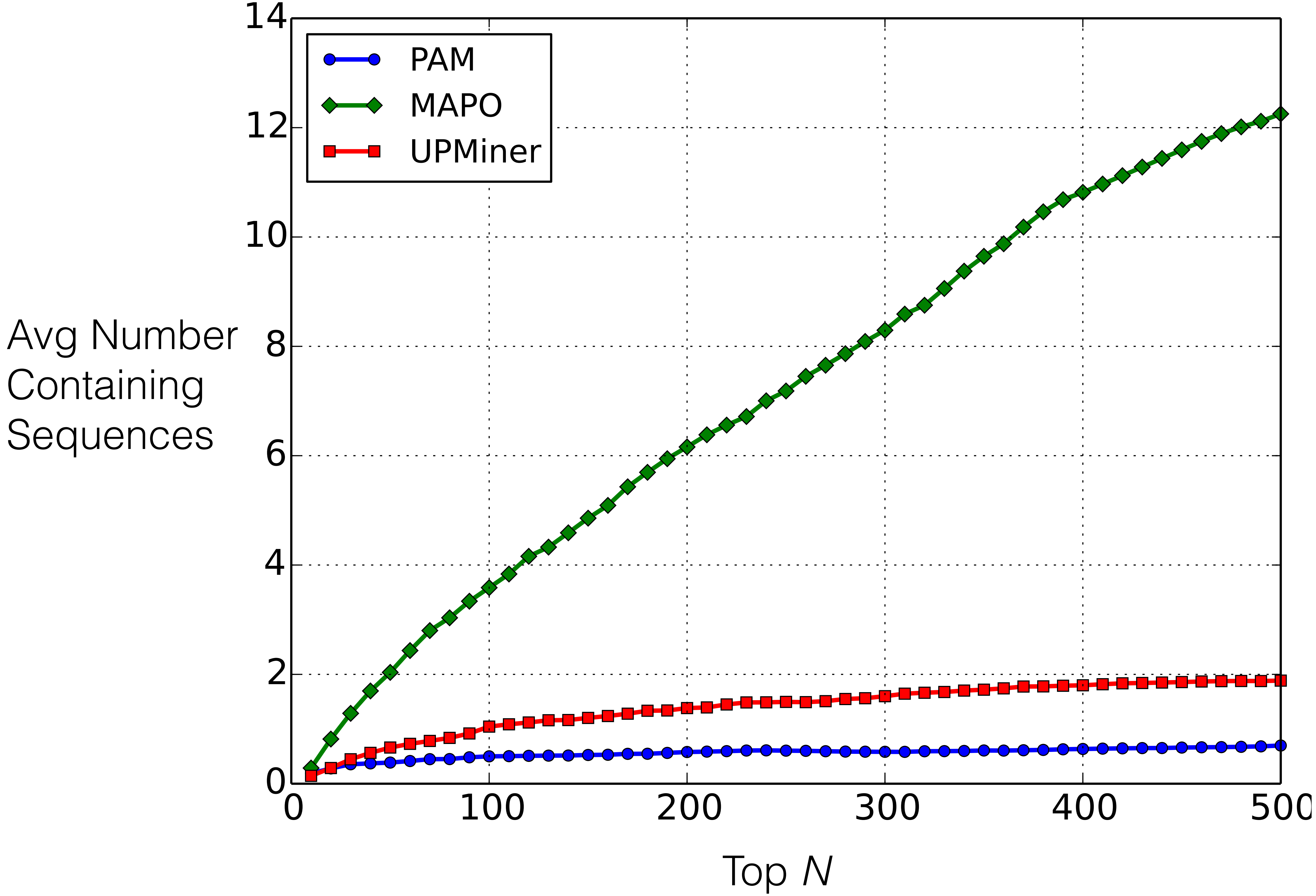
Prevalence in Client Code



Experimental Questions

- Quality
  - Match to “held-out” client code
  - Match to examples from library developers
  - Measure: sequence overlap, precision, recall
- Redundancy
  - Why? Ease of use, diversity
  - Measure: number of containing sequences
- All results averaged over the 17 libraries

Redundancy



Avg Number Containing Sequences
Top N
PAM
MAPO
UPMiner
0
2
4
6
8
10
12
14
0
100
200
300
400
500

Example: twitter4j

PAM	MAPO *[Zhong et al, '09]*	UPMiner *[Wang et al, '13]*
TwitterFactory.<init> TwitterFactory.getInstance	TwitterFactory.<init> TwitterFactory.getInstance	TwitterFactory.<init> TwitterFactory.getInstance
TwitterFactory.<init> TwitterFactory.getInstance Twitter.setOAuthConsumer Twitter.setOAuthAccessToken	Status.getUser Status.getText	TwitterFactory.getInstance Twitter.setOAuthConsumer
Status.getUser Status.getText	ConfigurationBuilder.<init> ConfigurationBuilder.build	TwitterFactory.<init> TwitterFactory.getInstance Twitter.setOAuthConsumer
AccessToken.getToken AccessToken.getTokenSecret	ConfigurationBuilder.<init> TwitterFactory.<init>	Status.getUserStatus.getText
ConfigurationBuilder.<init> ConfigurationBuilder.build TwitterFactory.<init> TwitterFactory.getInstance	ConfigurationBuilder.<init> ConfigurationBuilder.setOAuthConsu merKey	Twitter.setOAuthConsumer Twitter.setOAuthAccessToken

: two main types of twitter initialization call

Interactive Machine Learning

[Srivastava, Zou, and Sutton 2016]

Per clustering accept / reject

13	M	Red
17	M	Red
25	F	Blue
23	M	Yellow
19	F	Yellow
35	F	Yellow
57	M	Red
60	M	Red
61	F	Blue
31	F	Red

Data

clustering

Clustering:
Partition of data

13	M	Red
17	M	Red
25	F	Blue
23	M	Yellow
19	F	Yellow
35	F	Yellow
57	M	Red
60	M	Red
61	F	Blue
31	F	Red

Clustered data

Reject

Accept

Reject

User feedback

TINDER

Technique for INteractive Data Exploration via Rejection

13	M	Red
17	M	Red
25	F	Blue
23	M	Yellow
19	F	Yellow
35	F	Yellow
57	M	Red
60	M	Red
61	F	Blue
31	F	Red

Re-clustered data

Related work

- Other settings of interactive machine learning
  - Crayons *[Fails and Olsen, 2003]*
  - Overview “Power to the people”
    [Amershi et al, 2016]

- Clustering
  - Alternative clustering
    [Caruana et al, 2006] *[Bae and Bailey et al, 2006]*
    [Jain, Meka, Dhillon 2008] *[Dang and Bailey 2010]*

  - Other feedback types
    - Split/merge *[Cutting et al., 1992; Balcan & Blum, 2008]*
    - Must-link / Cannot-link *[Wagstaff et al., 2001; Basu et al., 2004]*
    - Feature-level *[Bekkerman et al., 2007; Dasgupta & Ng, 2010]*

Model

View as Bayesian prior elicitation. Revise prior based on feedback.

Prior based on mutual information based penalty.

Intuition: New clusters to not be predictable from old.

$$f_s(\theta, \theta_s) = I(H; H_s) = \sum_{h=1}^{K} \sum_{h_s=1}^{K} p_{\theta,\theta_s}(h, h_s) \log \frac{p_{\theta,\theta_s}(h, h_s)}{p_\theta(h) p_{\theta_s}(h_s)}.$$

new cluster labels

old cluster labels

Distribution over old and new clusters

$$p_{\theta,\theta_s}(h, h_s, x) = p(h|x, \theta) p(h_s|x, \theta_s) \tilde{p}(x),$$

input data distribution

new distribution over clusters

old distribution over clusters

Mutual information:
Label invariance

Reject all version: Yields CAMI *[Dang and Bailey 2010]*

Optimisation

Interactivity means we don't want to sweep the whole data set

$$f_s(\theta, \theta_s) = I(H; H_s) = \sum_{h=1}^{K} \sum_{h_s=1}^{K} p_{\theta,\theta_s}(h, h_s) \log \frac{p_{\theta,\theta_s}(h, h_s)}{p_\theta(h) p_{\theta_s}(h_s)}.$$

sum over data points within a log

EM can't help us

Instead: Lagrangian-relaxation type algorithm

Define a distribution

$$p(h, h_s) = N^{-1} \sum_i q_i(h) p(h_s | x_i, \theta_s).$$

$q_i(h)$ free parameter, replaces $p(h|x, \theta)$

(like the EM auxiliary distribution)

Then:

Use $p(h, h_s)$ within $f_s(\theta, \theta_s)$

Add a penalty to encourage $q_i(h) = p(h|x, \theta)$

Optimize via coordinate descent wrt q_i and θ

Algorithm

“E”-Step:

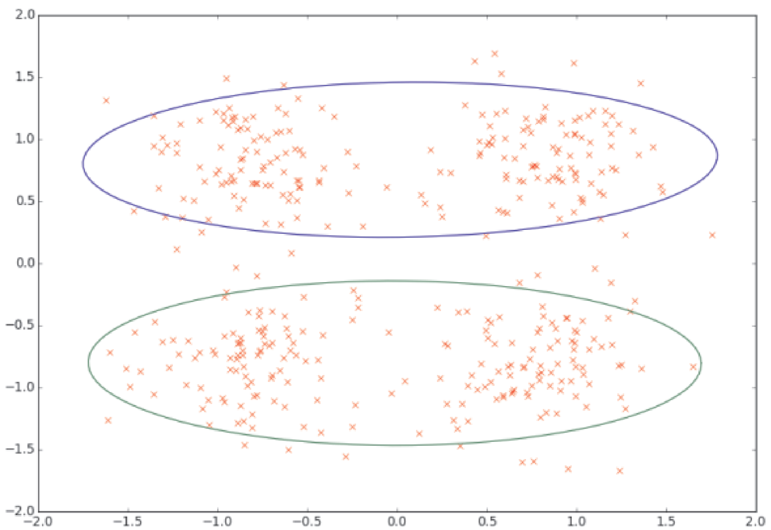
$$q \leftarrow \max_q -\beta \sum_s I(H; H_s) - \alpha \mathrm{KL}(q; p_\theta)$$

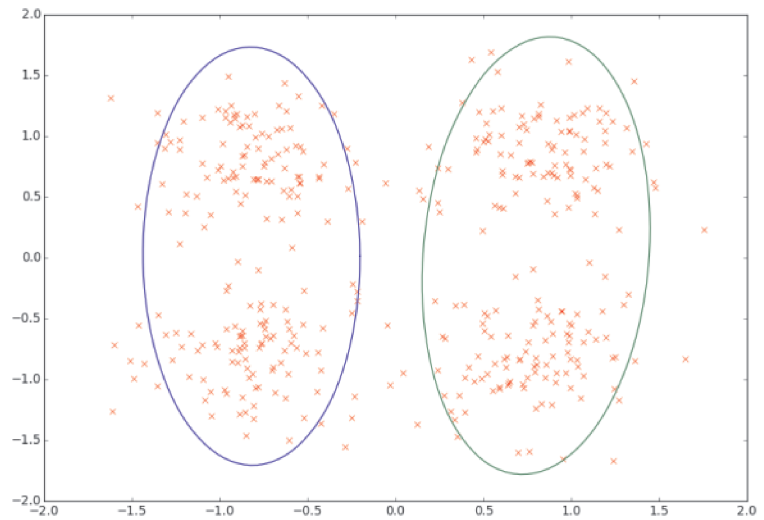
“M”-Step:

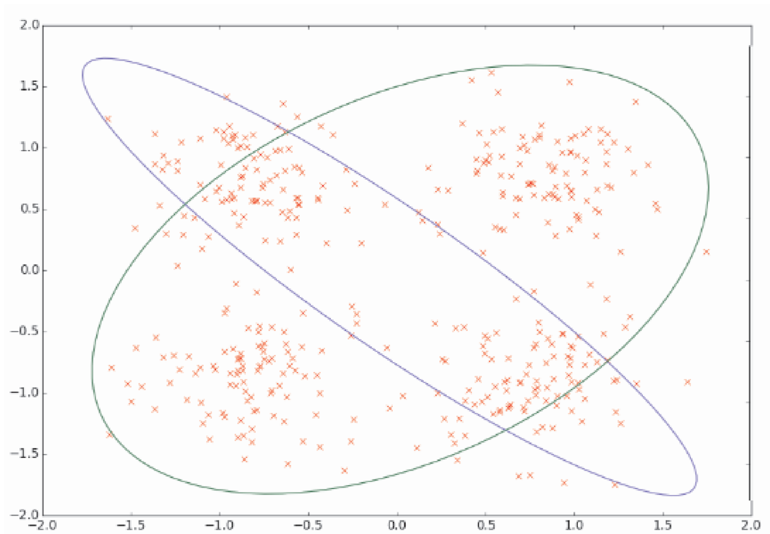
$$\theta \leftarrow \max_\theta E[\log p_\theta(v, h)]_q$$

- This is not EM! No lower bound
- But now E-step can be incremental
- Finds local optimum if at end $q_i(h) = p(h|x, \theta)$

Illustrative example


(a) Initial clustering


(b) After one "reject all"

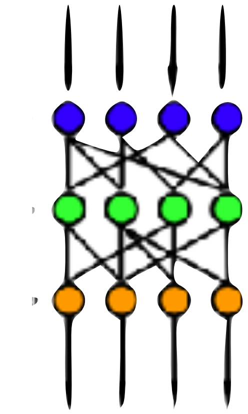

(c) After two "reject all"

Clustering quality (purity)

	CIFAR-10	CMU Face: Person	CMU Face: Gender	CMU Face: Pose
Random Restarts	0.89	0.37	0.87	**0.44**
dec-KMeans *[Jain, Meka, Dillon 2008]*	0.90	0.37	0.86	0.42
TINDER: Global	0.89	0.37	0.89	0.40
TINDER: Local	**0.93**	**0.39**	**0.93**	**0.44**

Clustering diversity

	CIFAR-10	CMU Face
Random Restarts	0.56	0.55
dec-KMeans *[Jain, Meka, Dillon 2008]*	0.90	0.37
TINDER: Global	0.89	0.37
TINDER: Local	**0.93**	**0.39**



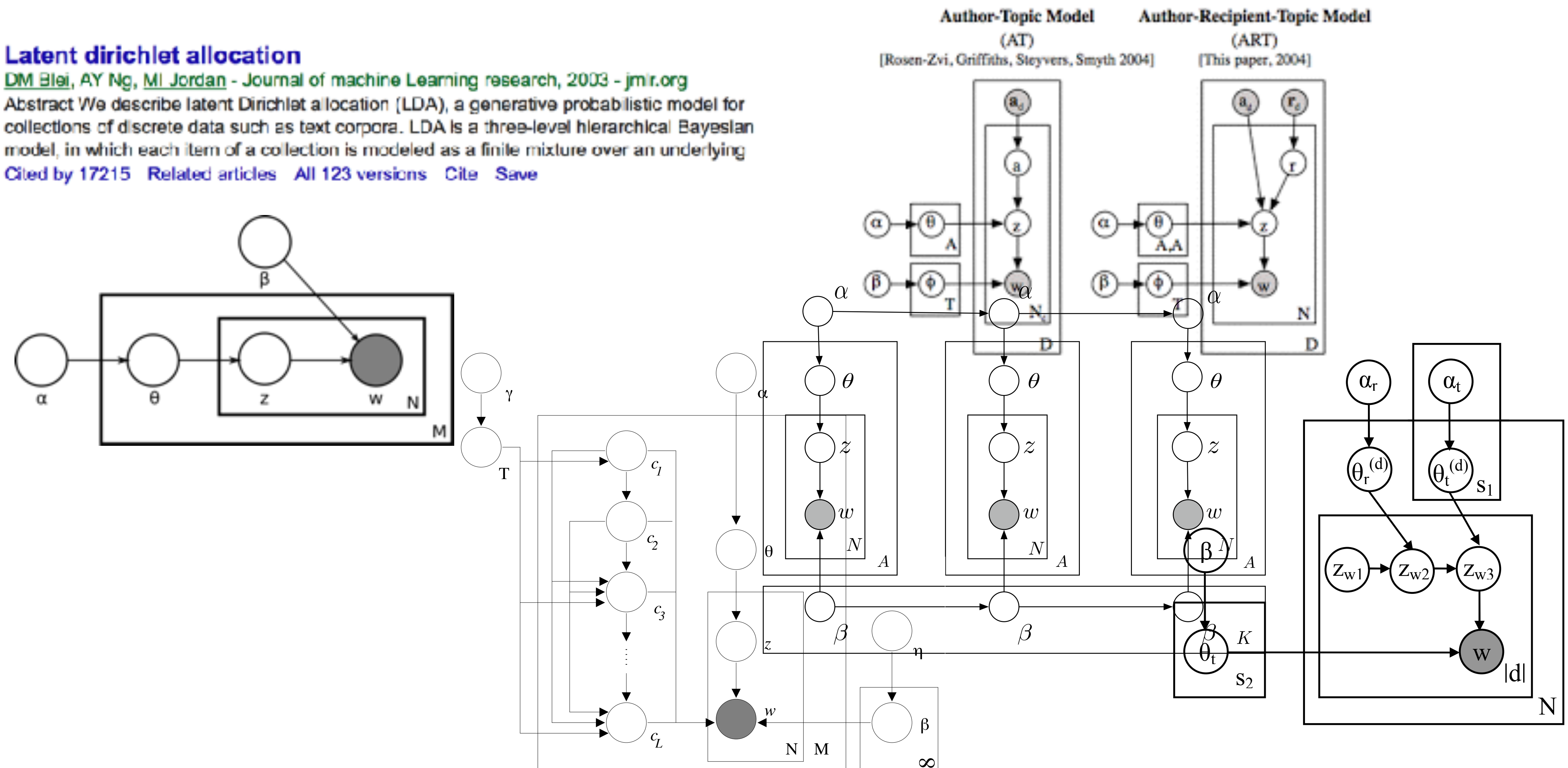
Super-Fast Neural Network Topic Modelling

[Srivastava and Sutton 2016]

Topic Models

motherboard meg printer quadra hd windows processor vga mhz connector
armenian genocide turks turkish muslim massacre turkey armenians armenia greek
voltage nec outlet circuit cable wiring wire panel motor install
season nhl team hockey playoff puck league flyers defensive player
israel israeli lebanese arab lebanon arabs civilian territory palestinian militia

Topic Models: The Industry



But…

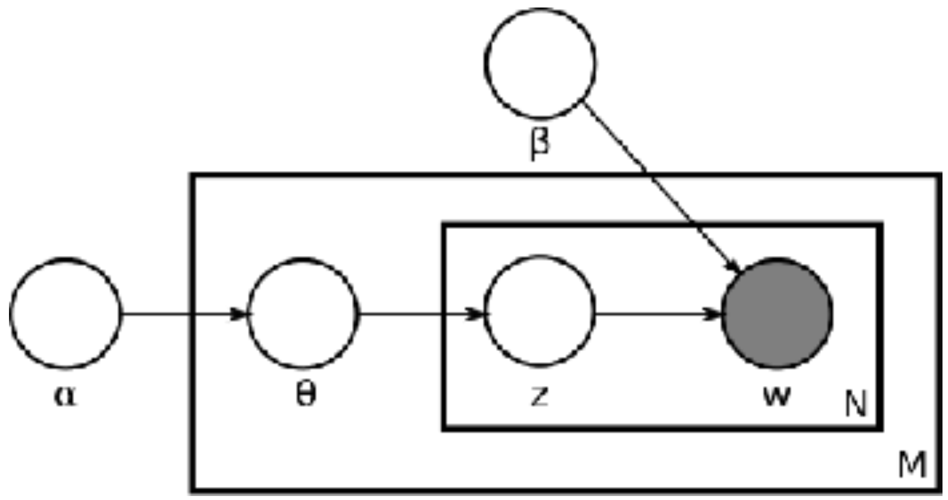
- New topic model means new inference algorithm
- MCMC general but slow; variational fast but not as general, and lower quality topics

Recognition Networks

[Hinton et al, 1995]

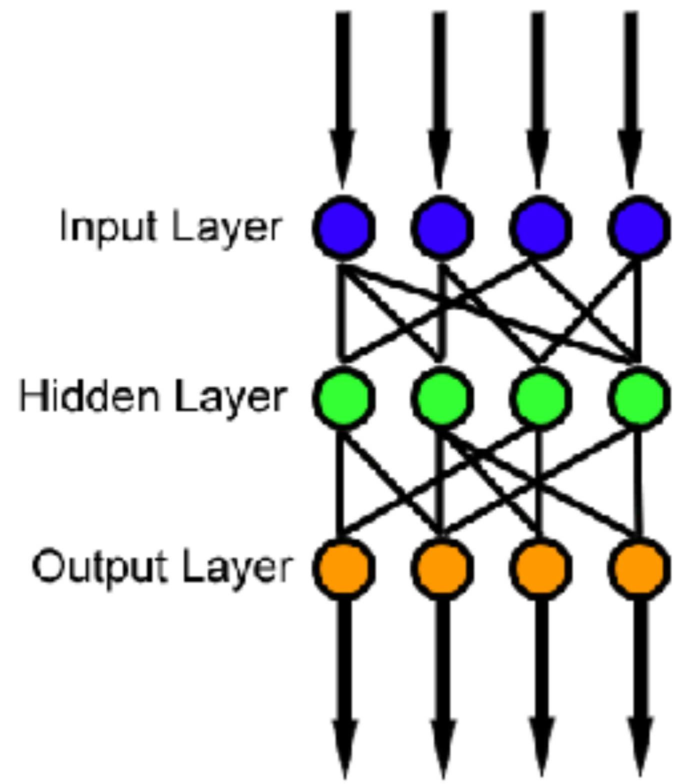
Inference network

Generator



$$p(\mathbf{w}, \theta, z | \alpha)$$

LDA



$$\theta$$

$$q(\theta | \gamma, \phi)$$

Appx posterior

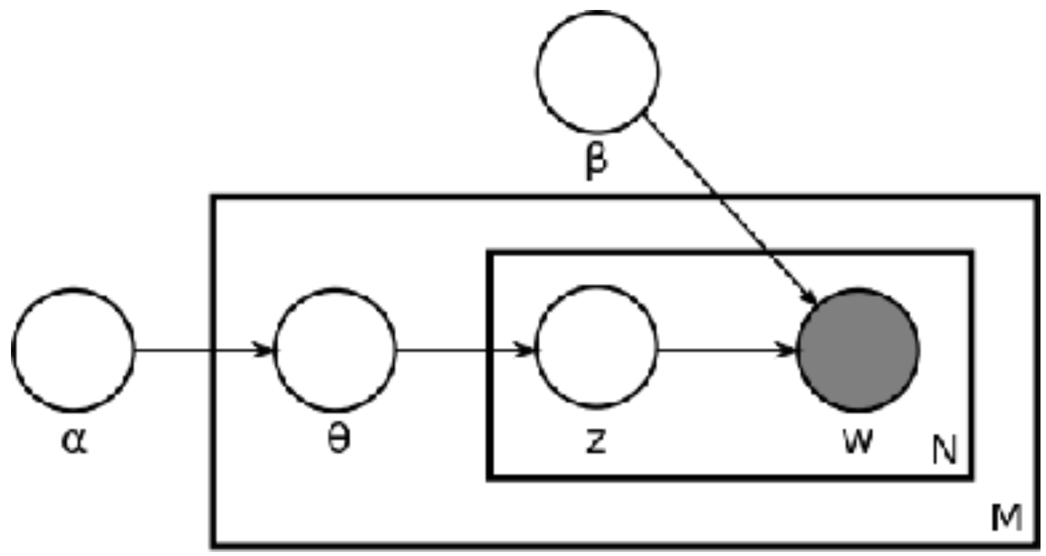
Variational autoencoder *[Kingma and Welling, 2013]*

$$L(\gamma, \phi \mid \alpha, \beta) = -D_{KL}\left[q(\theta, z|\gamma, \phi) || p(\theta, z|\alpha)\right] + \mathbb{E}_{q(\theta, z|\gamma,\phi)}[\log p(\mathbf{w}|z, \theta, \alpha, \beta)]$$

How to make it work

- Discrete variables
  - Marginalize them out
- Dirichlet difficult to reparameterize
  - Use Laplace approximation *[MacKay, 1998]*
- Topic collapsing (a bad local minimum)
  - High momentum in ADAM *[Kingma and Ba, 2014]*
  - Batch normalization *[Ioffe and Szegeti, 2015]*

Deriving new models



LDA

$$p(w_n|\theta,\beta) = \sum_k \theta_k p(w_n|z_n = k, \beta)$$

ProdLDA

$$p(w_n|\theta,\beta) \propto \prod_k p(w_n|z_n = k, \beta)^{\theta_k}$$

Very simple change to LDA, but hasn't been seen before.

Why?

- Before, I would have derived a variational inference algorithm
- Now, change one line of code

Evaluation

# topics	ProdLDA	LDA NVLDA	LDA DMFVI	LDA Collapsed Gibbs	NVDM
50	**0.24**	0.20	0.11	0.17	0.08
200	**0.19**	0.12	0.06	0.14	0.06

Topic coherence (20 Newsgroups)

# topics	ProdLDA	LDA NVLDA	LDA DMFVI	LDA Collapsed Gibbs	NVDM
50	**0.14**	0.07	-	0.04	0.07
200	**0.12**	0.05	-	0.06	0.05

Topic coherence (RCV1)

Look Ma, no inference!

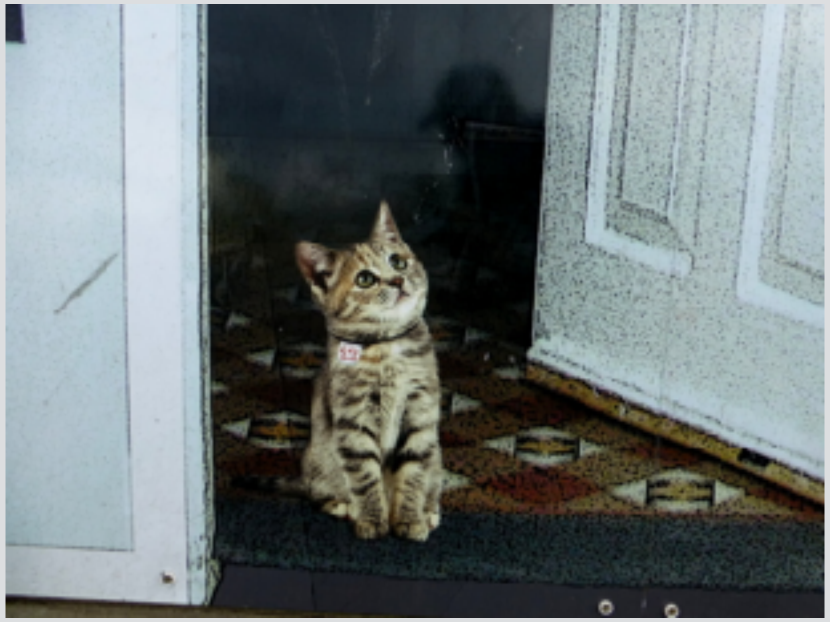
# Topics	Inference Network Only	Inference Network + Optimization
50	1172	1162
200	1168	1151

Test set perplexity (20 Newsgroups)

Accurate topic inference on new topic
with one pass of a feedforward neural network

Machine Learning for Data Exploration

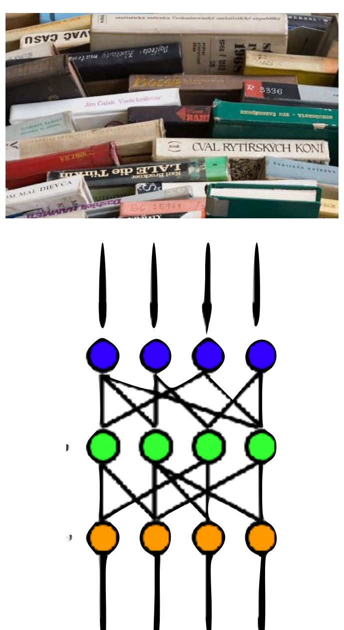
Charles Sutton, University of Edinburgh



Help cats explore!

Pattern mining

API mining



Neural topic modelling

Interactive clustering

Thanks!

- Jaroslav Fowkes
- Akash Srivastava
- James Zou

EPSRC Engineering and Physical Sciences Research Council

Microsoft Research