Templates

for scalable data analysis

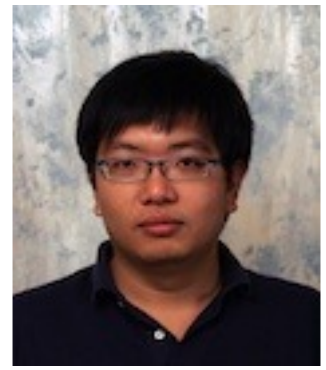
1 Introduction to Big Learning

Amr Ahmed, Alexander J Smola, Markus Weimer
Yahoo! Research & UC Berkeley & ANU

Thanks


Mohamed Aly


Joey Gonzalez


Yucheng Low


Qirong Ho


Shravan Narayanamurthy


Amr Ahmed


Choon Hui Teo


Eric Xing


James Petterson

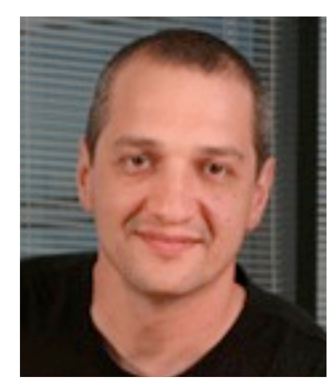

Sergiy Matyusevich


Jake Eisenstein

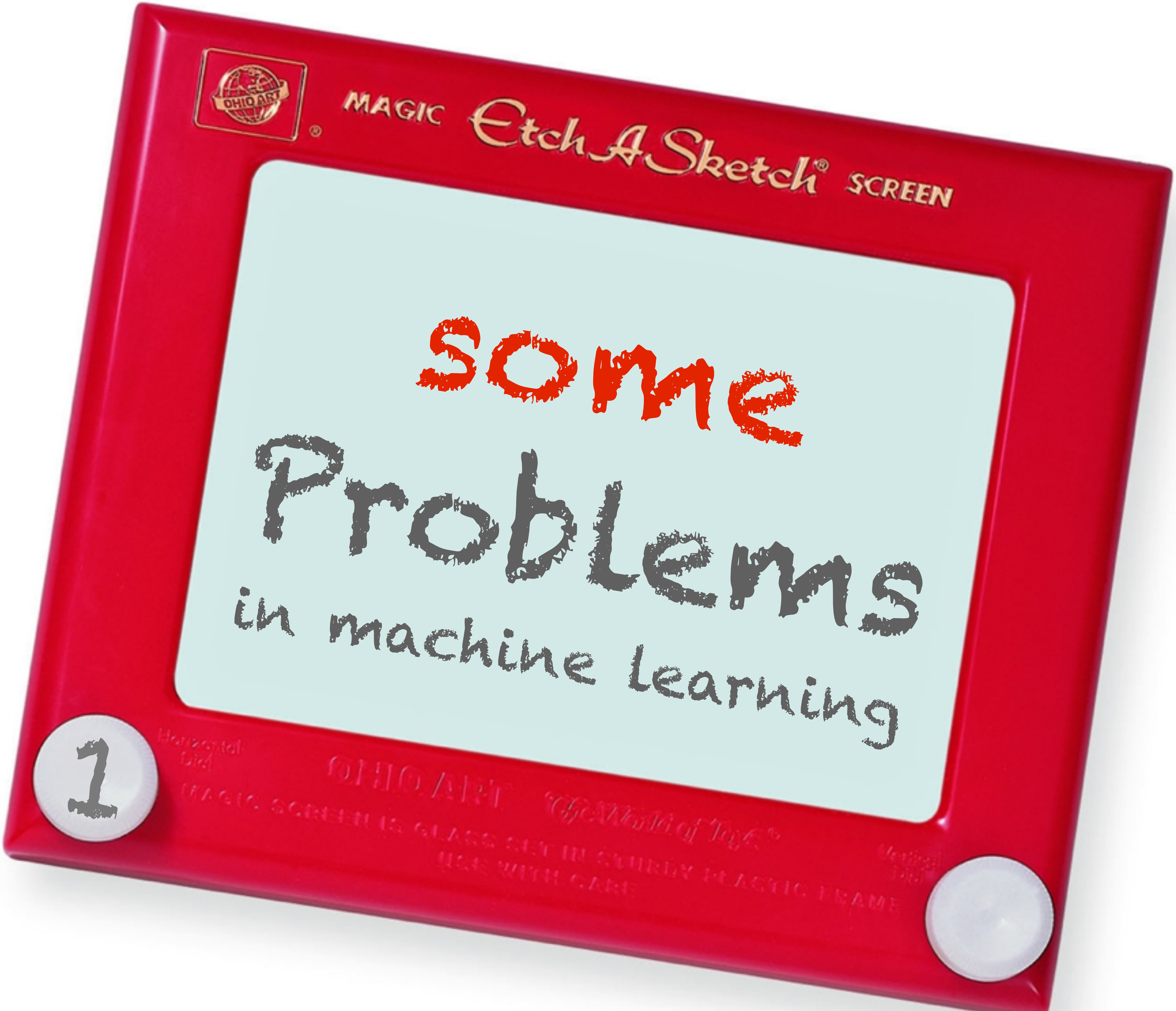

Shuang Hong Yang


Vishy Vishwanathan

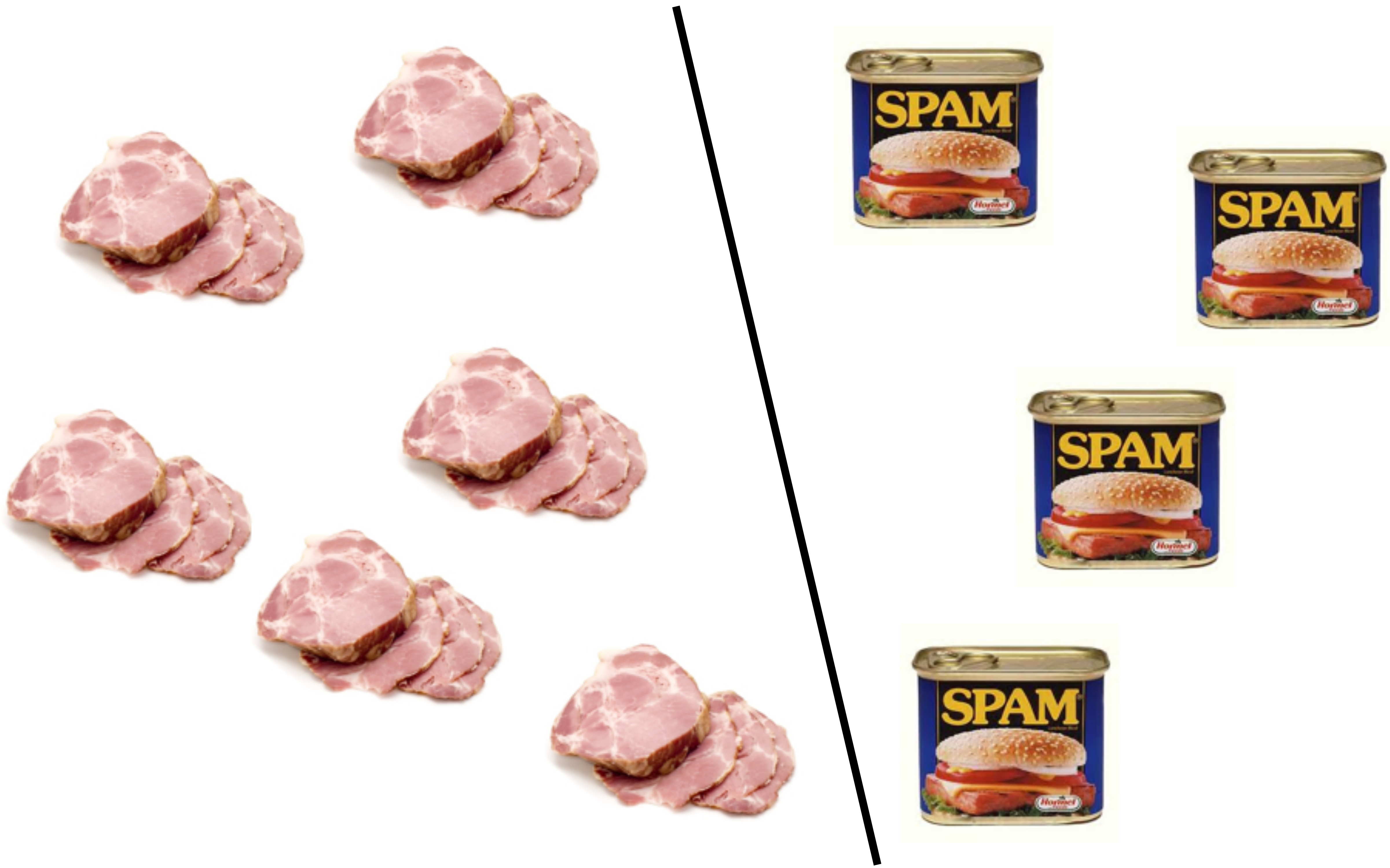

Markus Weimer


Vanja Josifovski

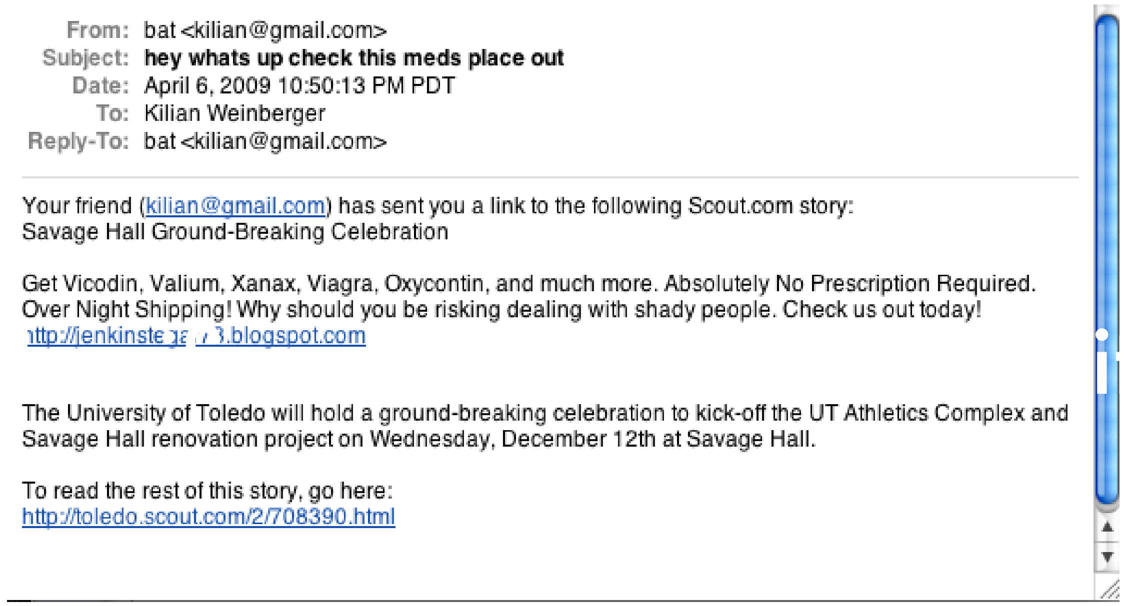
- Problems
  in machine learning
- Systems
  to run the algorithms
- Response
  batch/online/interactive
- Compression


OHIO ART
MAGIC Etch A Sketch SCREEN
some
Problems
in machine learning
1

Classification



Spam Filtering



From: bat <kilian@gmail.com>
Subject: **hey whats up check this meds place out**
Date: April 6, 2009 10:50:13 PM PDT
To: Kilian Weinberger
Reply-To: bat <kilian@gmail.com>

Your friend (kilian@gmail.com) has sent you a link to the following Scout.com story:
Savage Hall Ground-Breaking Celebration

Get Vicodin, Valium, Xanax, Viagra, Oxycontin, and much more. Absolutely No Prescription Required.
Over Night Shipping! Why should you be risking dealing with shady people. Check us out today!
http://jenkinste[illegible]3.blogspot.com

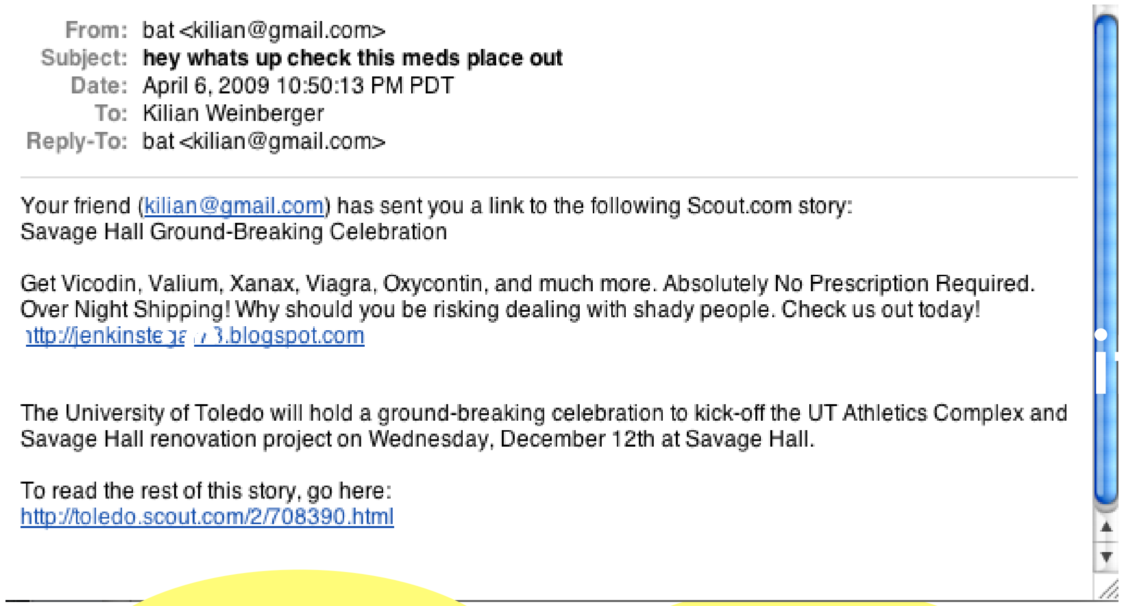
The University of Toledo will hold a ground-breaking celebration to kick-off the UT Athletics Complex and Savage Hall renovation project on Wednesday, December 12th at Savage Hall.

To read the rest of this story, go here:
http://toledo.scout.com/2/708390.html





Spam Filtering



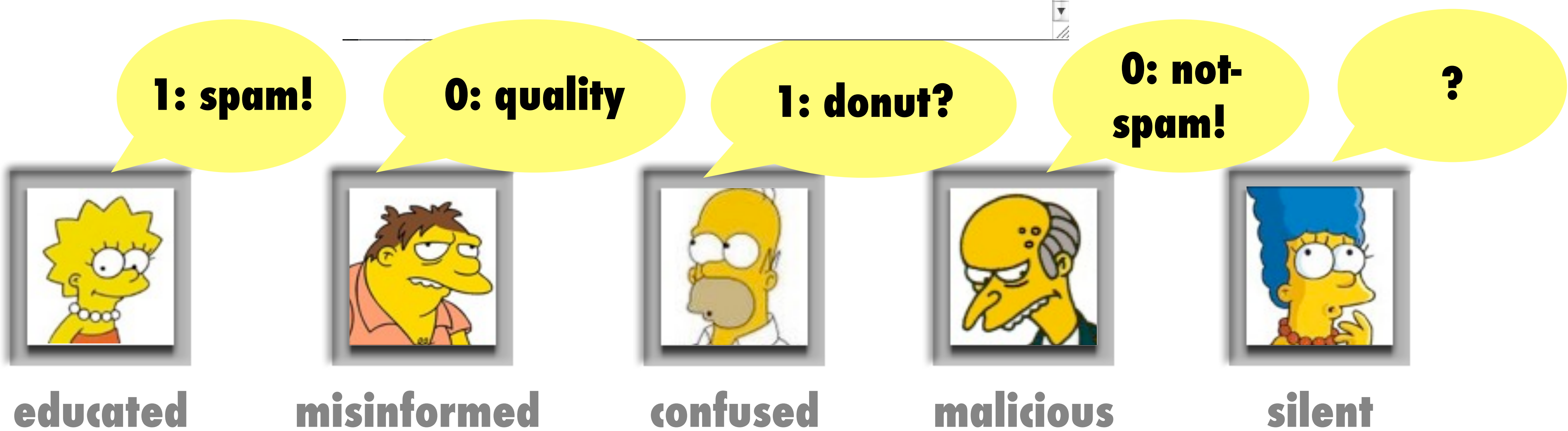
From: bat <kilian@gmail.com>
Subject: **hey whats up check this meds place out**
Date: April 6, 2009 10:50:13 PM PDT
To: Kilian Weinberger
Reply-To: bat <kilian@gmail.com>

Your friend (kilian@gmail.com) has sent you a link to the following Scout.com story:
Savage Hall Ground-Breaking Celebration

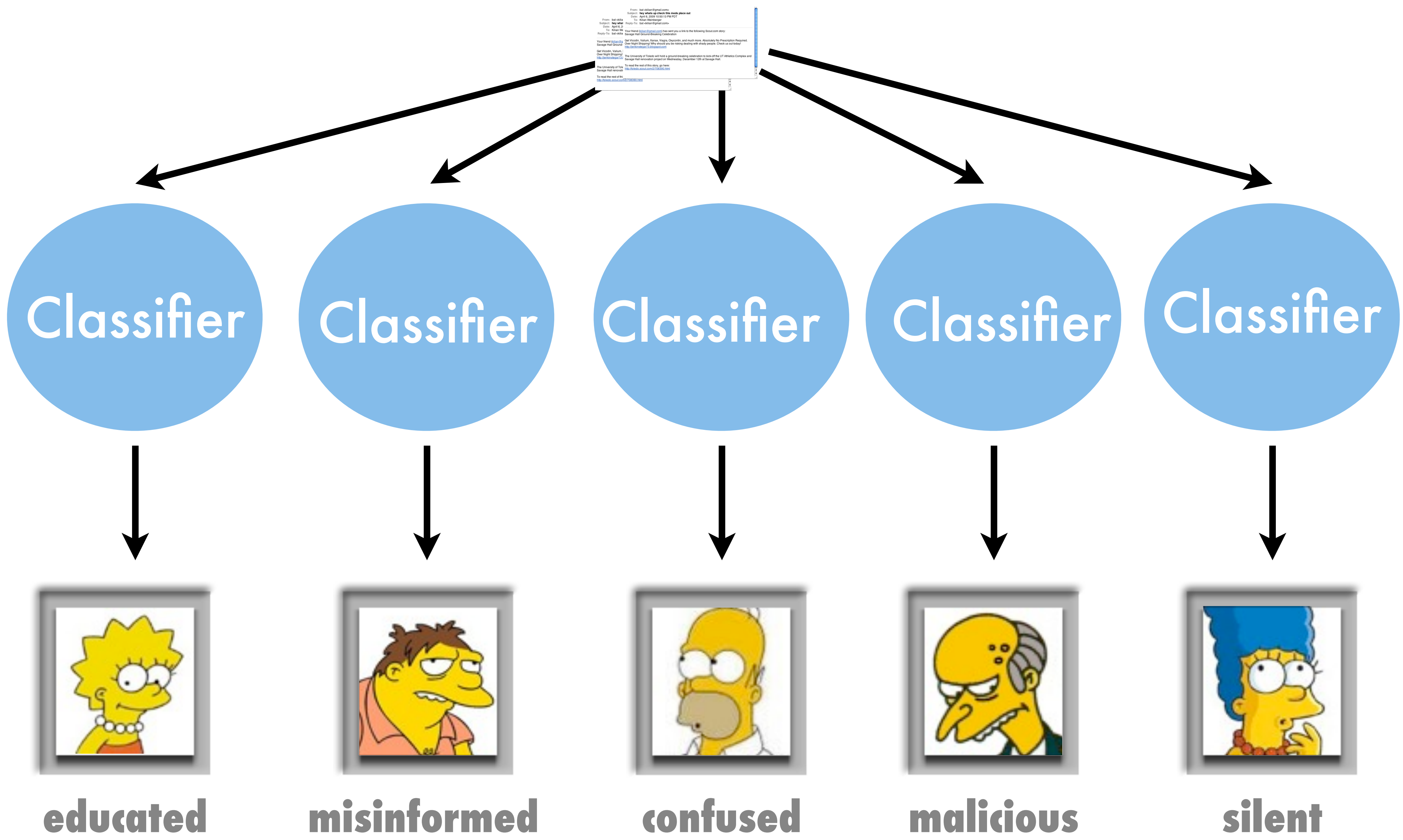
Get Vicodin, Valium, Xanax, Viagra, Oxycontin, and much more. Absolutely No Prescription Required. Over Night Shipping! Why should you be risking dealing with shady people. Check us out today!
ttp://jenkinste [illegible] 3.blogspot.com

The University of Toledo will hold a ground-breaking celebration to kick-off the UT Athletics Complex and Savage Hall renovation project on Wednesday, December 12th at Savage Hall.

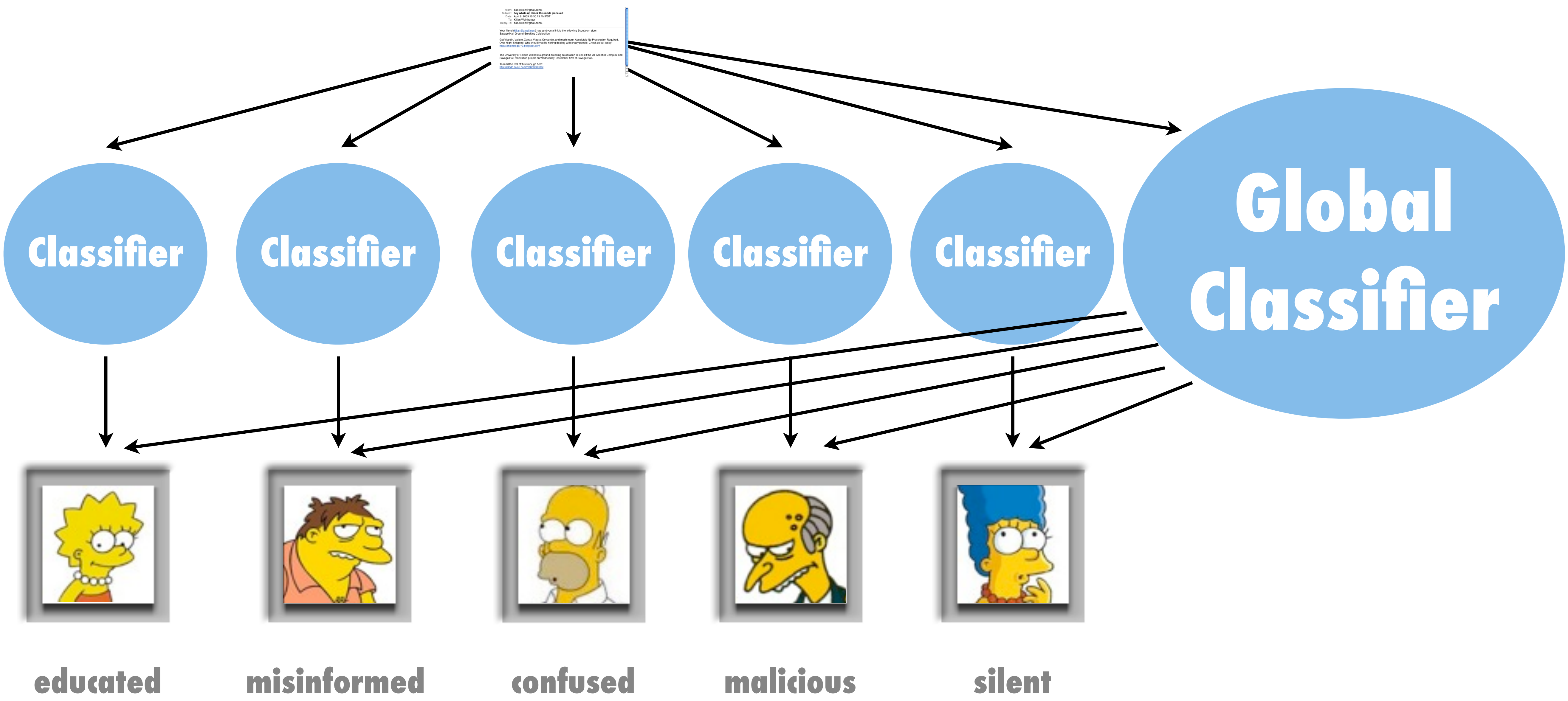
To read the rest of this story, go here:
http://toledo.scout.com/2/708390.html



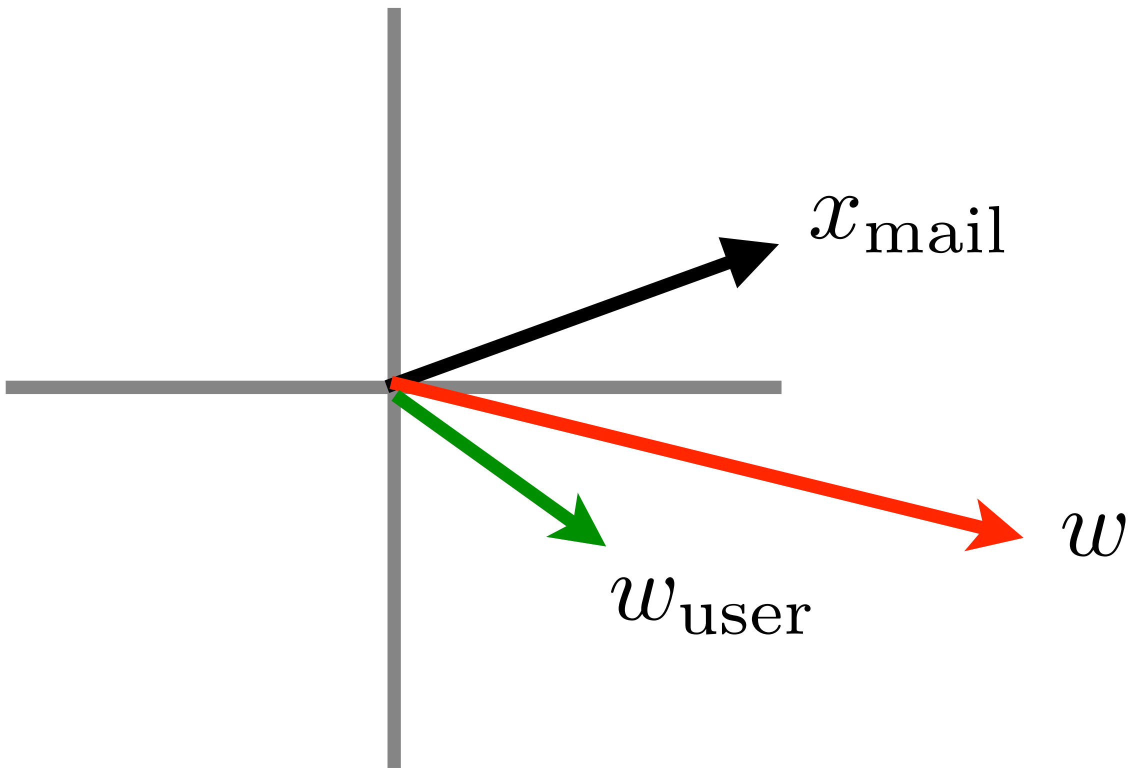
Spam Filtering



Personalized Spam Filtering



Personalized Spam Filtering



- Function representation

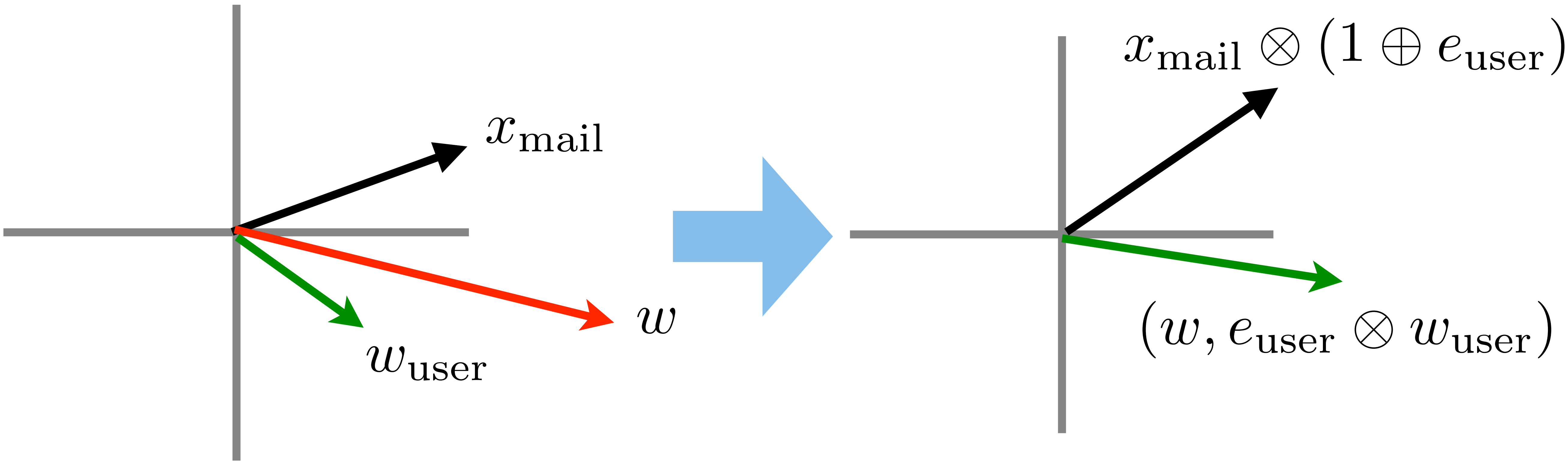
$$f(x, u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle$$

(corresponds to multitask kernel of Pontil & Michelli, Daume)

- Reduce to binary classification problem and classify with

$$\operatorname{sgn} f(x, u)$$

Personalized Spam Filtering



- Function representation

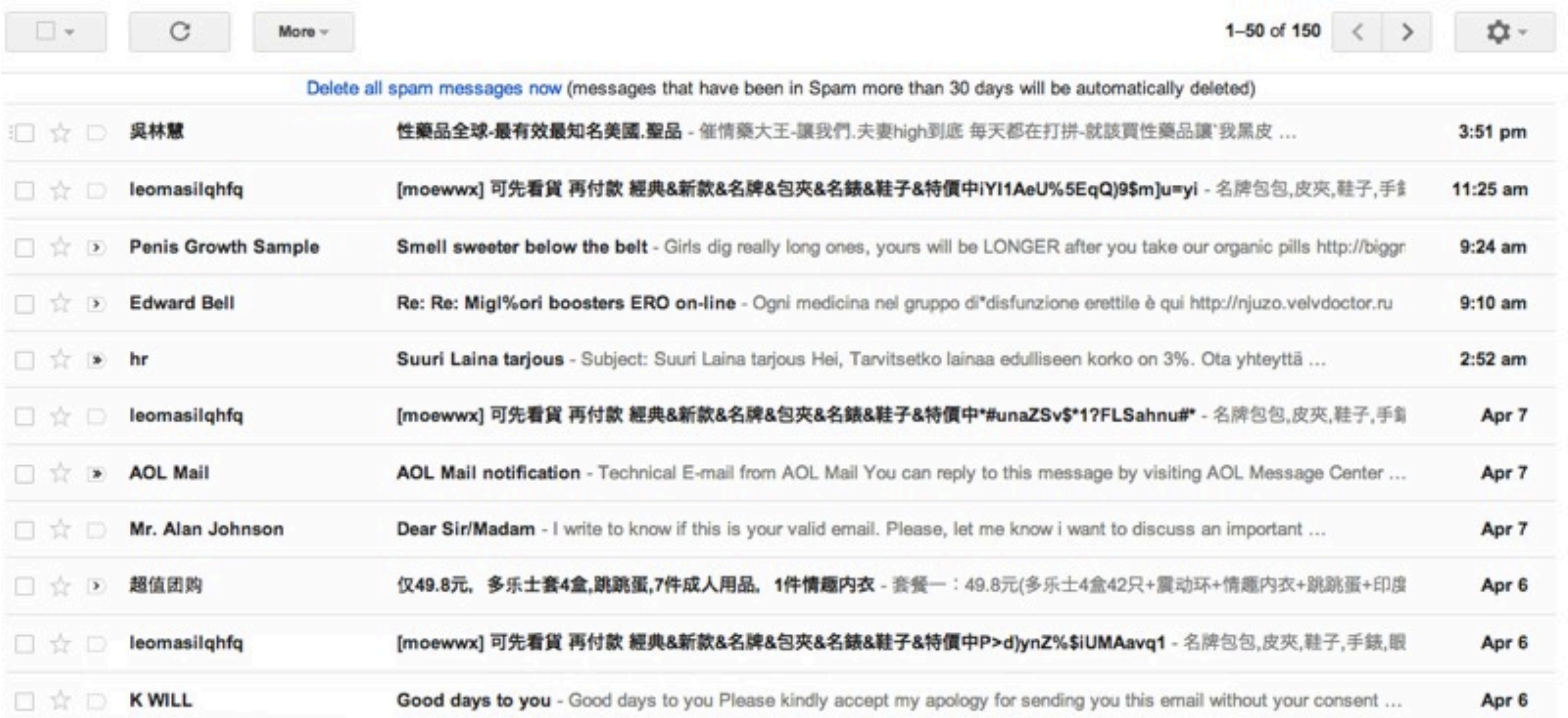
$$f(x,u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle$$

(corresponds to multitask kernel of Pontil & Michelli, Daume)

- Reduce to binary classification problem and classify with

$$\operatorname{sgn} f(x,u)$$

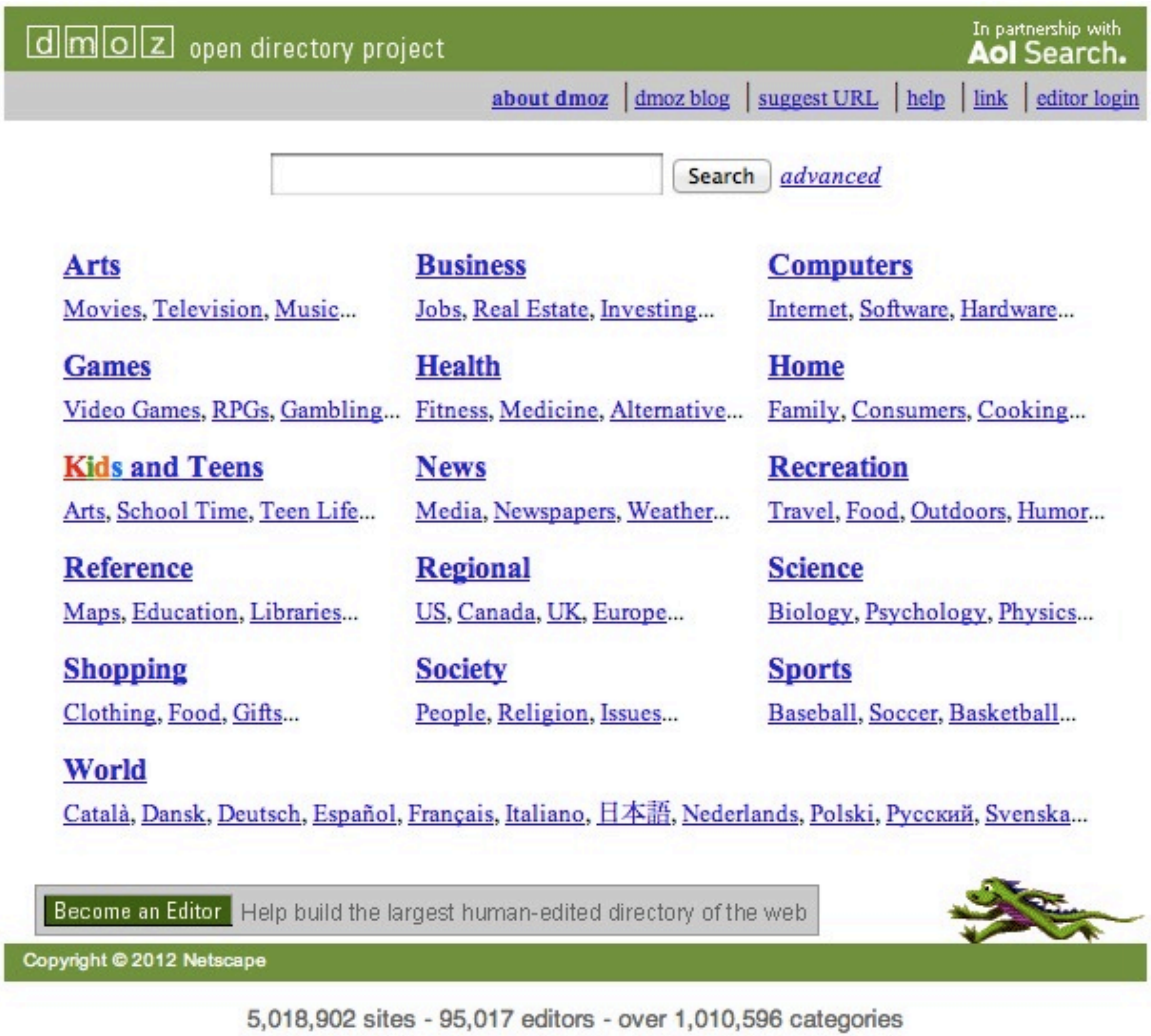
Personalized Spam Filtering



- 100-1000 million users
- 10-1000 messages per user
- Distributed storage and processing
- Real-time response required
- Implicit response

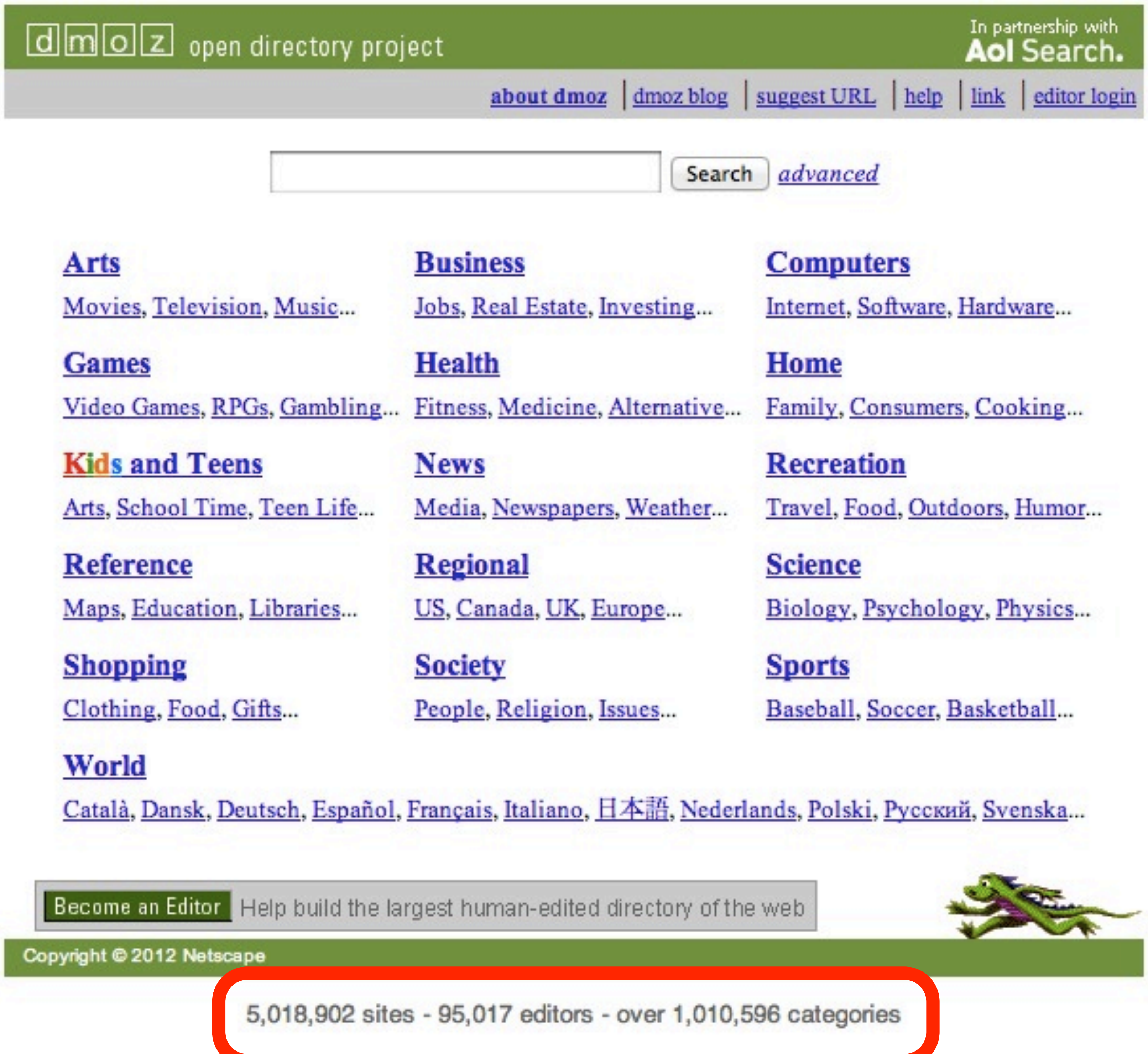
$$\underset{w}{\text{minimize}} \sum_{i=1}^{m} \max(0, 1 - y \langle w, x \rangle) + \frac{\lambda}{2} \|w\|^2$$

Ontologies



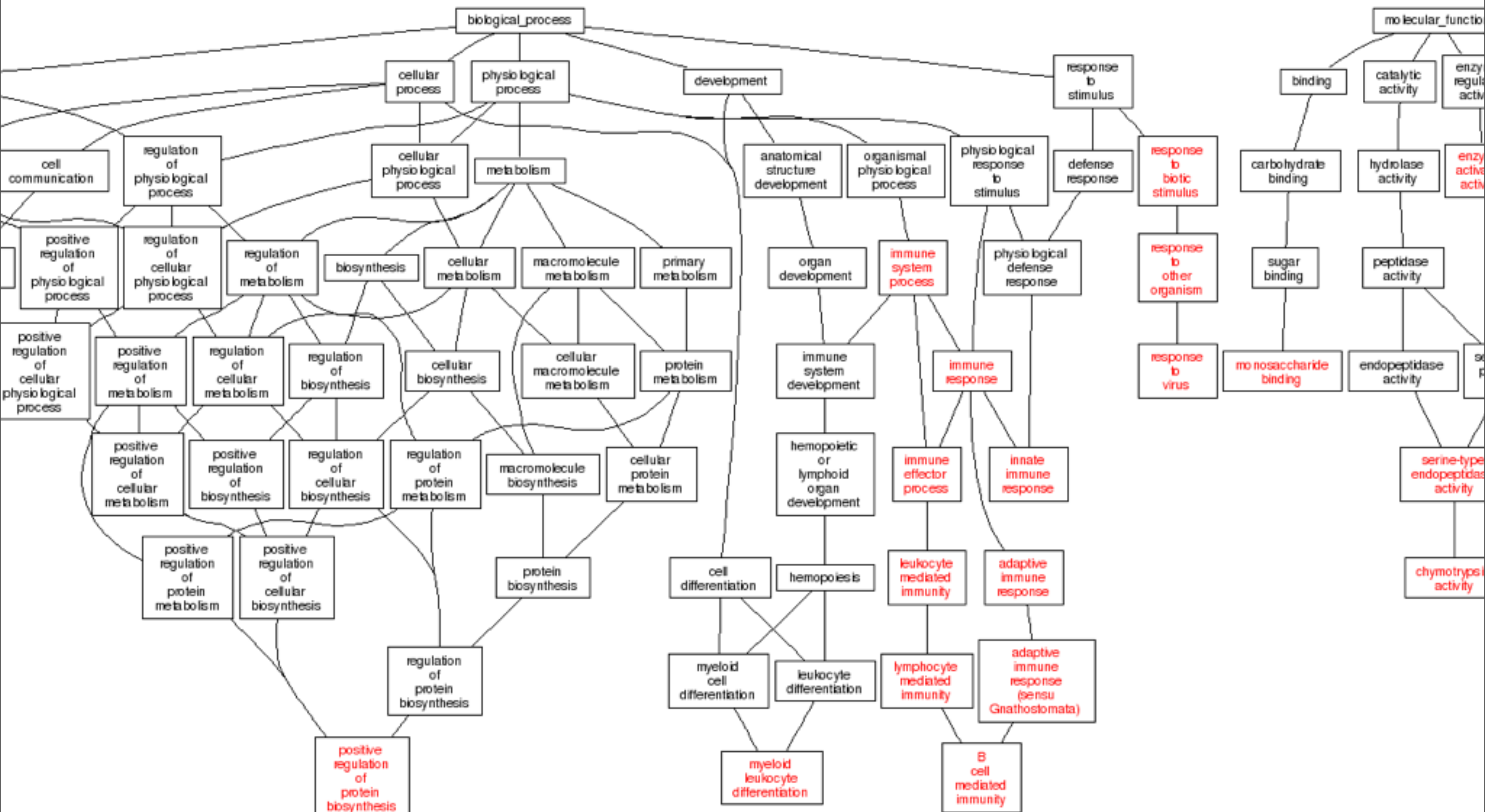
- 10k to 1M categories
- Few instances per category
- Hierarchical structure (top level more important than leaf)
- Category selection arbitrary
- Low entropy on leaves
- Often several ontologies in use

Ontologies



- 10k to 1M categories
- Few instances per category
- Hierarchical structure (top level more important than leaf)
- Category selection arbitrary
- Low entropy on leaves
- Often several ontologies in use

Gene Ontology DAG



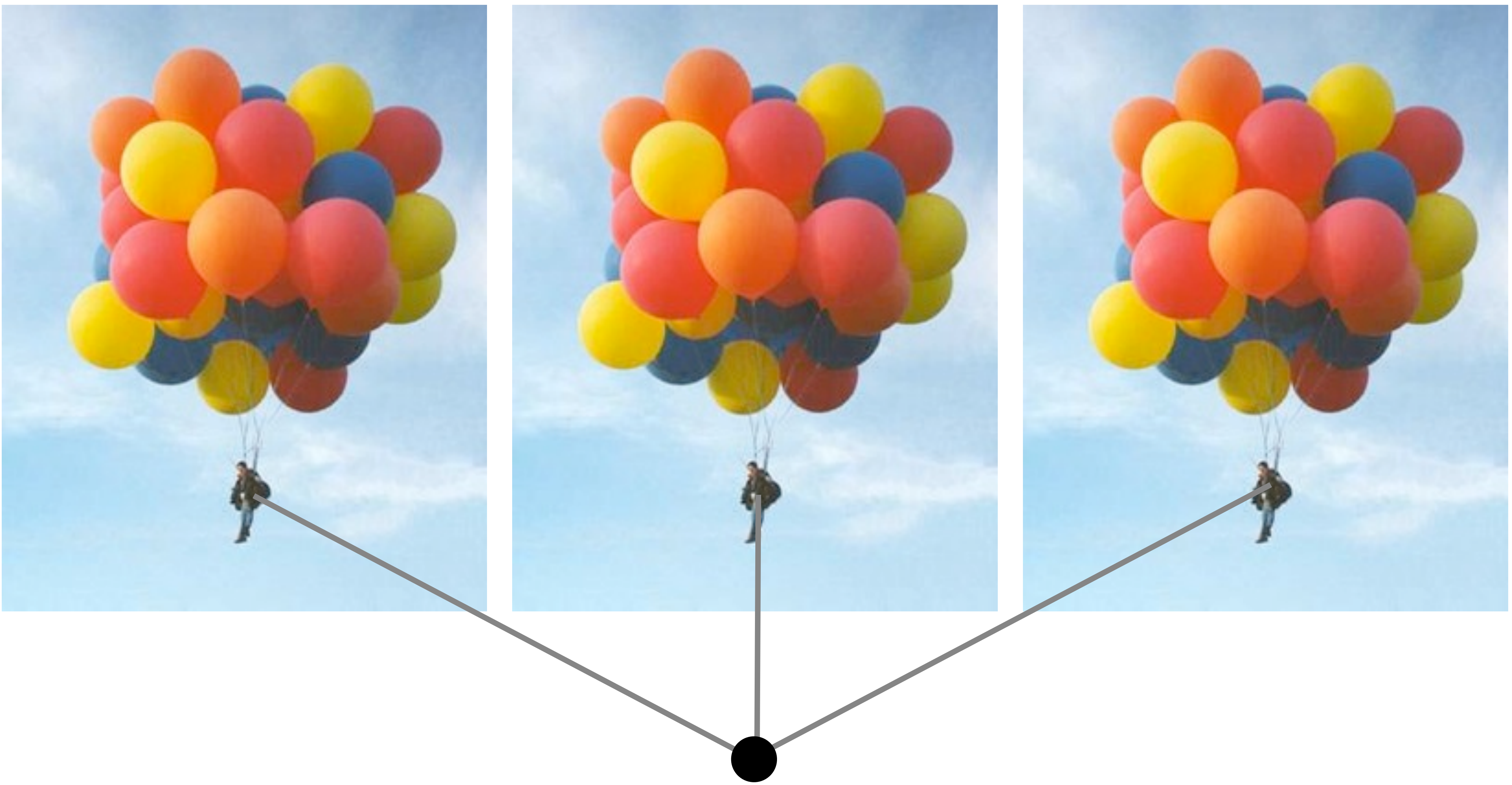
Ontologies

- 1000s of categories
- High error rate (impossible to learn them all)
- Structured loss
  (count common top level categories)
- Good strategy is additive function class

$$f(x, y) = \sum_{y' \in \mathrm{path}(y)} \langle w_{y'}, x \rangle$$

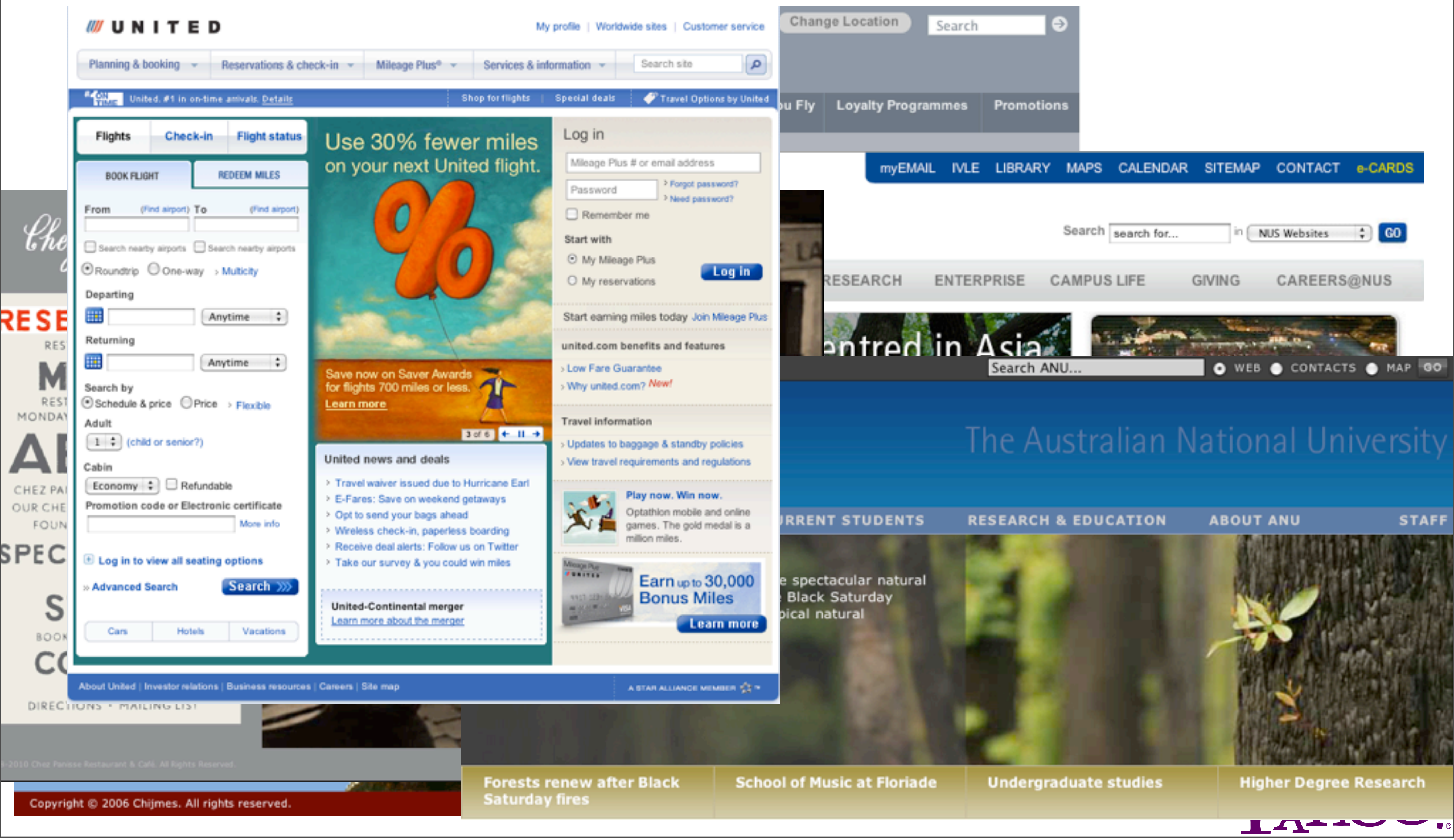
  Need efficient decoding on tree
- Alternative - obtain ontology automatically

Clustering



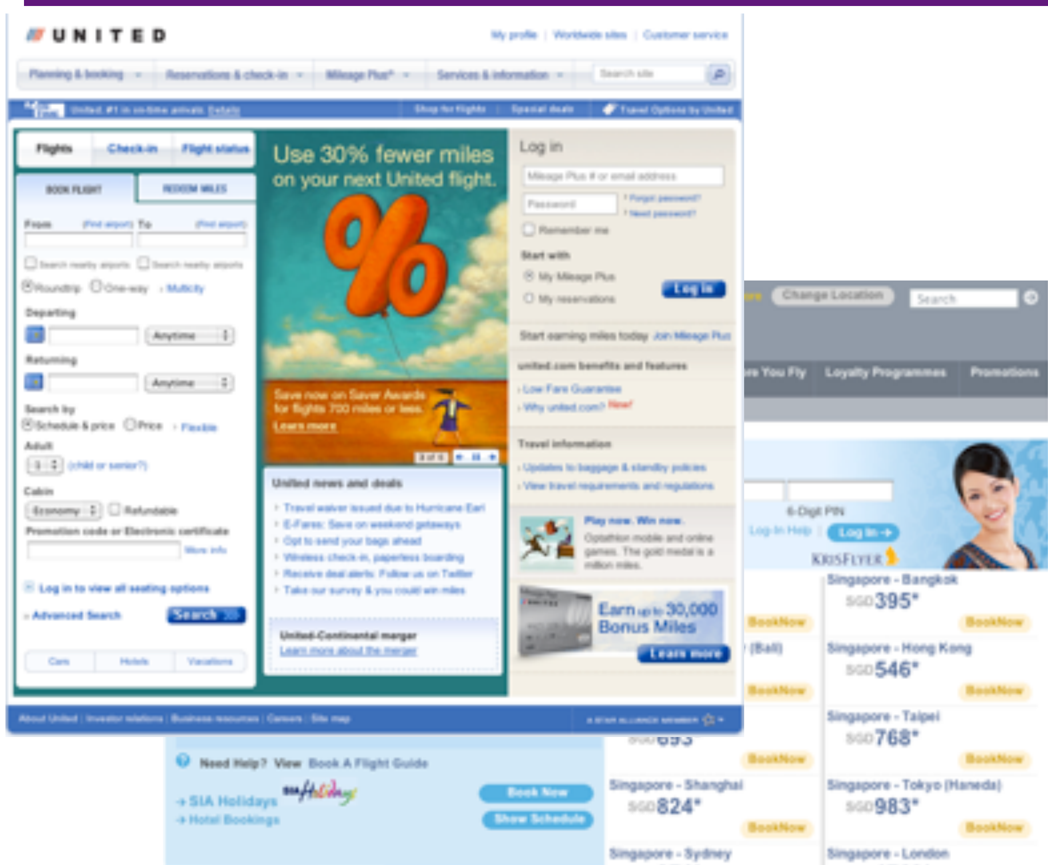
Clustering

Clustering



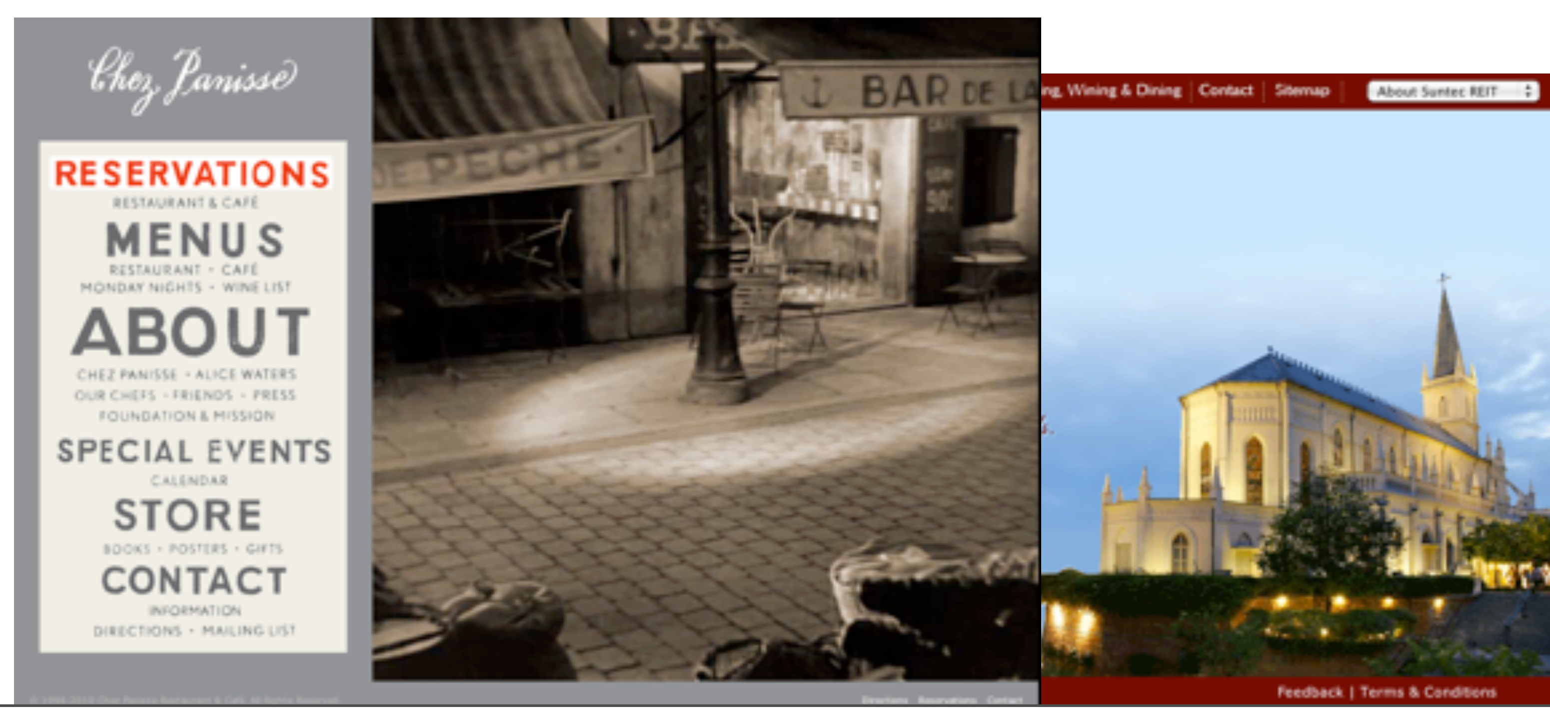
UNITED
My profile | Worldwide sites | Customer service
Planning & booking
Reservations & check-in
Mileage Plus®
Services & information
Search site
United. #1 in on-time arrivals. Details
Shop for flights
Special deals
Travel Options by United
Flights
Check-in
Flight status
BOOK FLIGHT
REDEEM MILES
From (Find airport) To (Find airport)
Search nearby airports
Roundtrip
One-way
Multicity
Departing
Anytime
Returning
Anytime
Search by
Schedule & price
Price
Flexible
Adult
(child or senior?)
Cabin
Economy
Refundable
Promotion code or Electronic certificate
More info
Log in to view all seating options
Advanced Search
Search
Cars
Hotels
Vacations
Use 30% fewer miles on your next United flight.
Save now on Saver Awards for flights 700 miles or less.
Learn more
United news and deals
Travel waiver issued due to Hurricane Earl
E-Fares: Save on weekend getaways
Opt to send your bags ahead
Wireless check-in, paperless boarding
Receive deal alerts: Follow us on Twitter
Take our survey & you could win miles
United-Continental merger
Learn more about the merger
Log in
Mileage Plus # or email address
Password
Forgot password?
Need password?
Remember me
Start with
My Mileage Plus
My reservations
Log in
Start earning miles today Join Mileage Plus
united.com benefits and features
Low Fare Guarantee
Why united.com? New!
Travel information
Updates to baggage & standby policies
View travel requirements and regulations
Play now. Win now.
Optathlon mobile and online games. The gold medal is a million miles.
Earn up to 30,000 Bonus Miles
Learn more
About United | Investor relations | Business resources | Careers | Site map
A STAR ALLIANCE MEMBER
Change Location
Search
Loyalty Programmes
Promotions
myEMAIL
IVLE
LIBRARY
MAPS
CALENDAR
SITEMAP
CONTACT
e-CARDS
Search
search for...
in
NUS Websites
GO
RESEARCH
ENTERPRISE
CAMPUS LIFE
GIVING
CAREERS@NUS
Search ANU...
WEB
CONTACTS
MAP
GO
The Australian National University
RESEARCH & EDUCATION
ABOUT ANU
STAFF
Forests renew after Black Saturday fires
School of Music at Floriade
Undergraduate studies
Higher Degree Research
Copyright © 2006 Chijmes. All rights reserved.

Clustering



UNITED
Use 30% fewer miles on your next United flight.
Log in

EXPLORE ANU
A-Z INDEX
ANU
The Australian National University
HOME
FUTURE STUDENTS
CURRENT STUDENTS
RESEARCH & EDUCATION
ABOUT ANU
STAFF
Ash forests rise and rise again
Forests renew after Black Saturday fires
School of Music at Floriade
Undergraduate studies
Higher Degree Research
Joint Evacuation Exercises
PROSPECTIVE STUDENTS
CURRENT STUDENTS
STAFF
ALUMNI
VISITORS

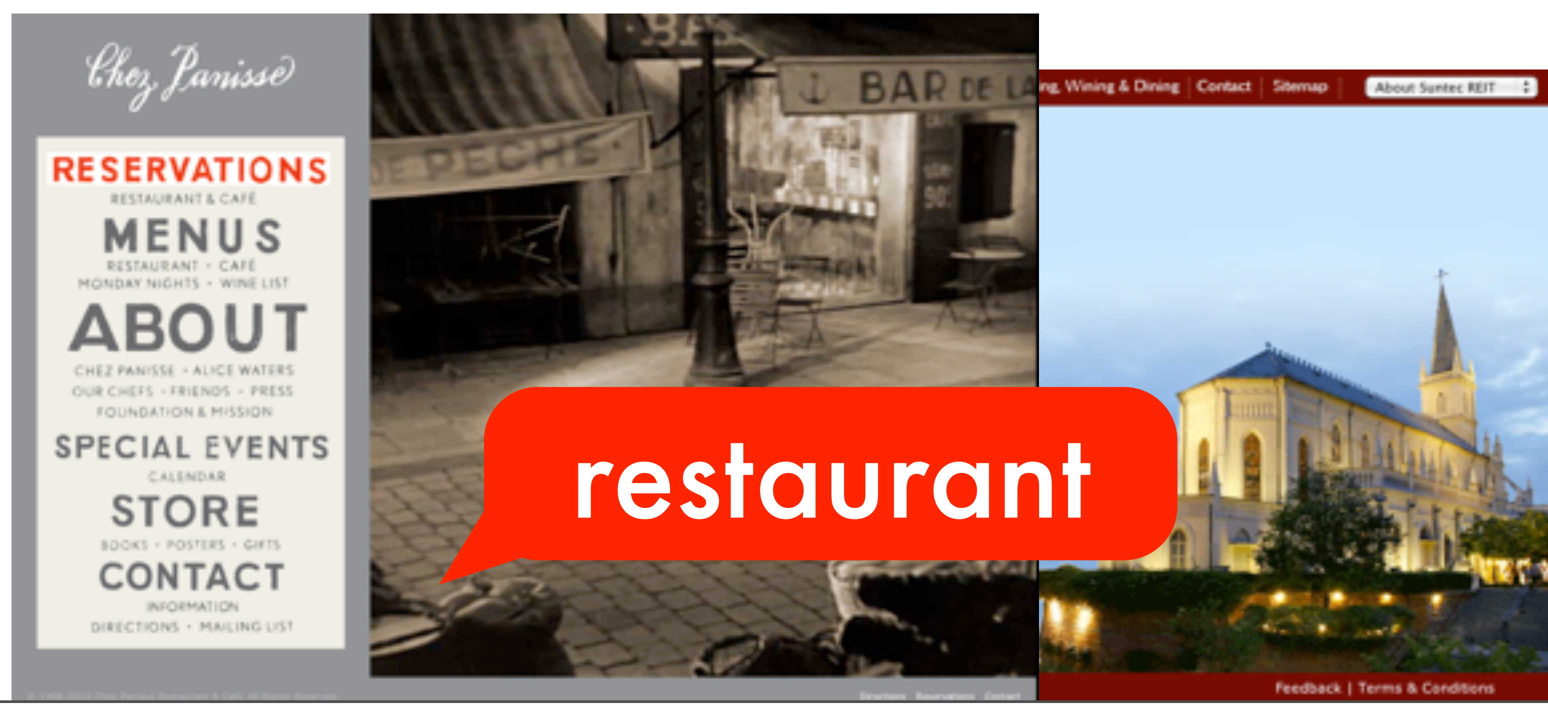


Chez Panisse
RESERVATIONS
MENUS
ABOUT
SPECIAL EVENTS
STORE
CONTACT
BAR DE LA

Clustering

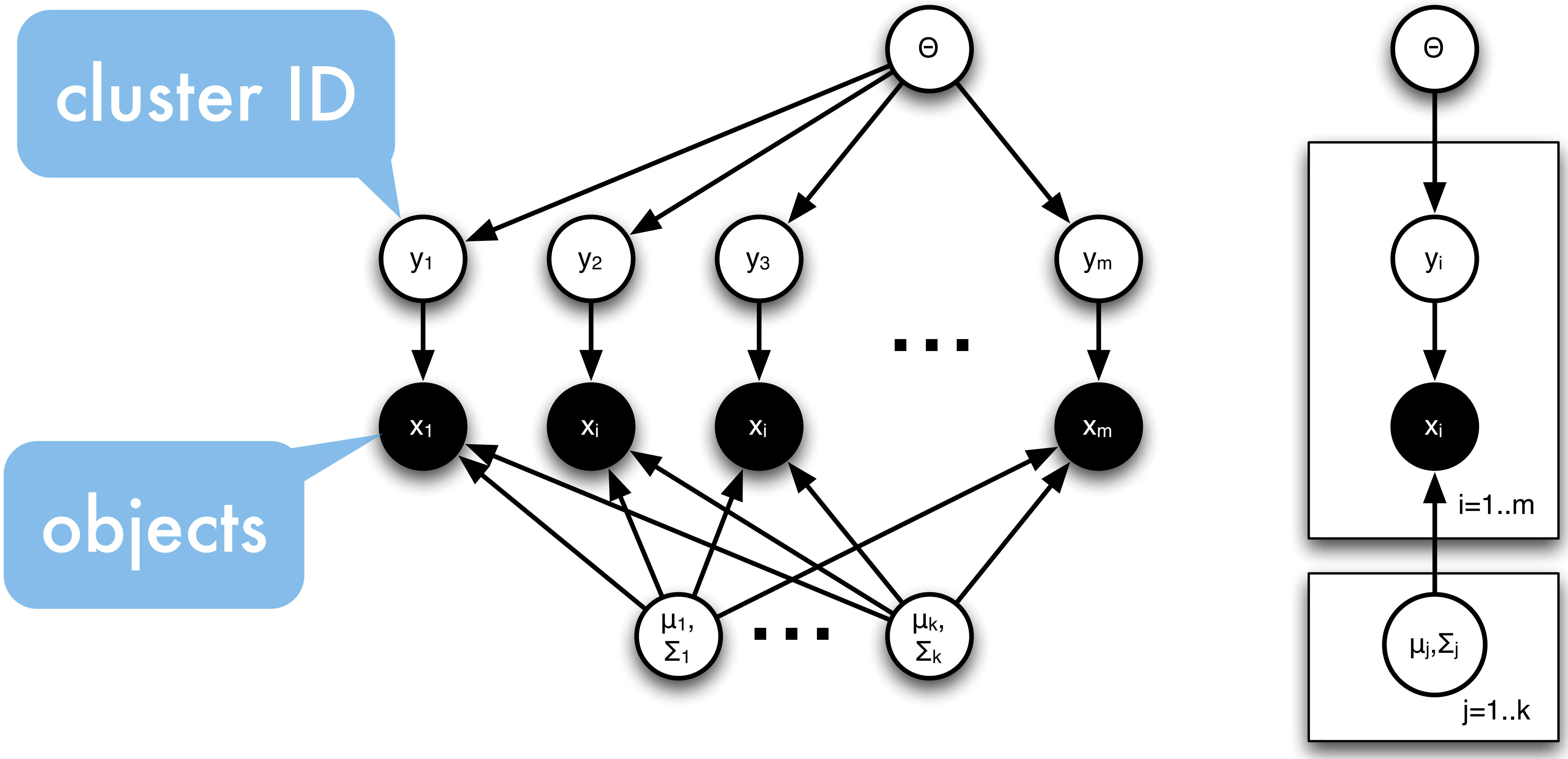
UNITED
Use 30% fewer miles on your next United flight.
airline

ANU
The Australian National University
university

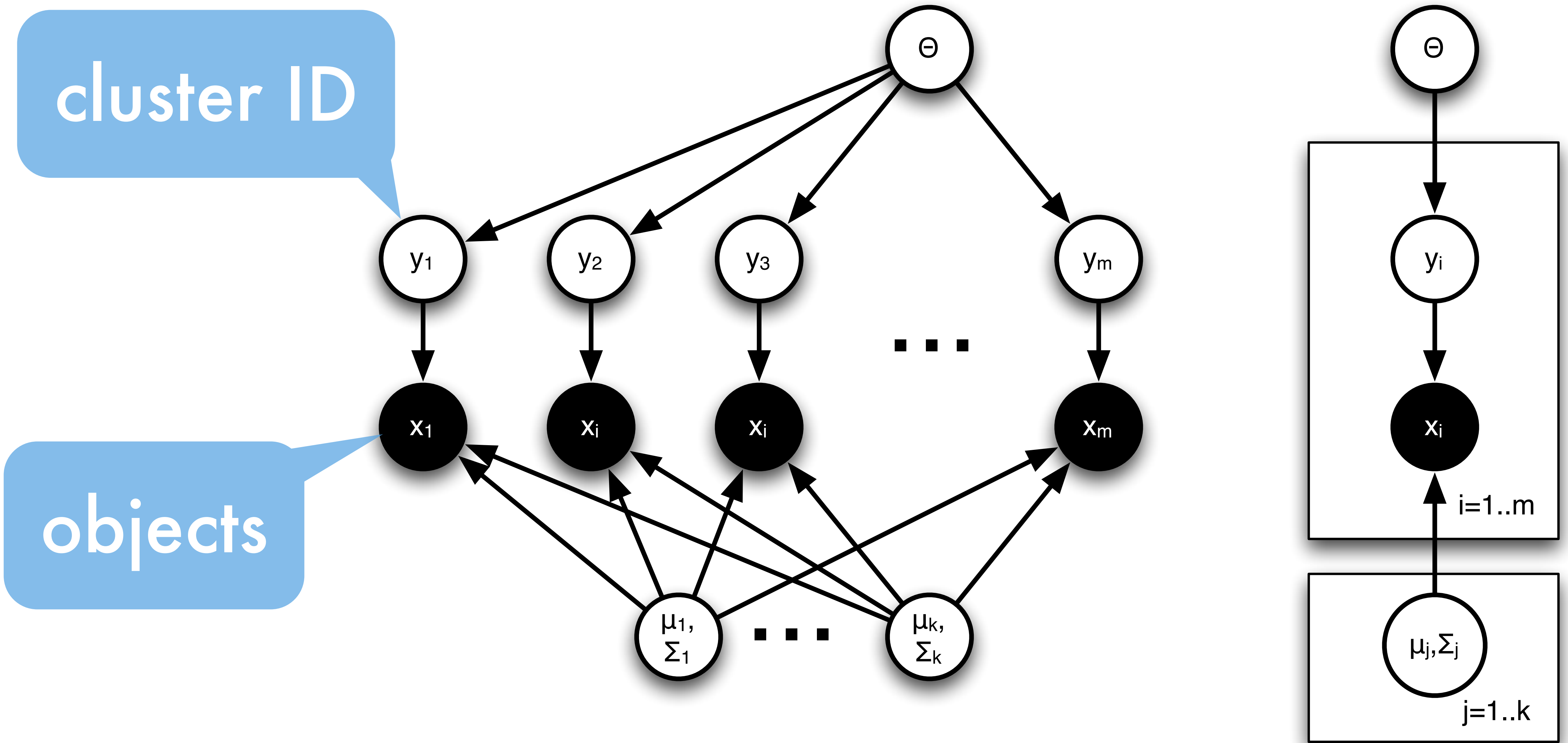


Chez Panisse
RESERVATIONS
MENUS
ABOUT
SPECIAL EVENTS
STORE
CONTACT
restaurant

Generative Model



Generative Model



$$p(X, Y|\theta, \sigma, \mu) = \prod_{i=1}^{n} p(x_i|y_i, \sigma, \mu)p(y_i|\theta)$$

What can we cluster?

What can we cluster?

products

urls

mails

text

news

users

queries

locations

spammers

ads

events

abuse

Topic Models

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

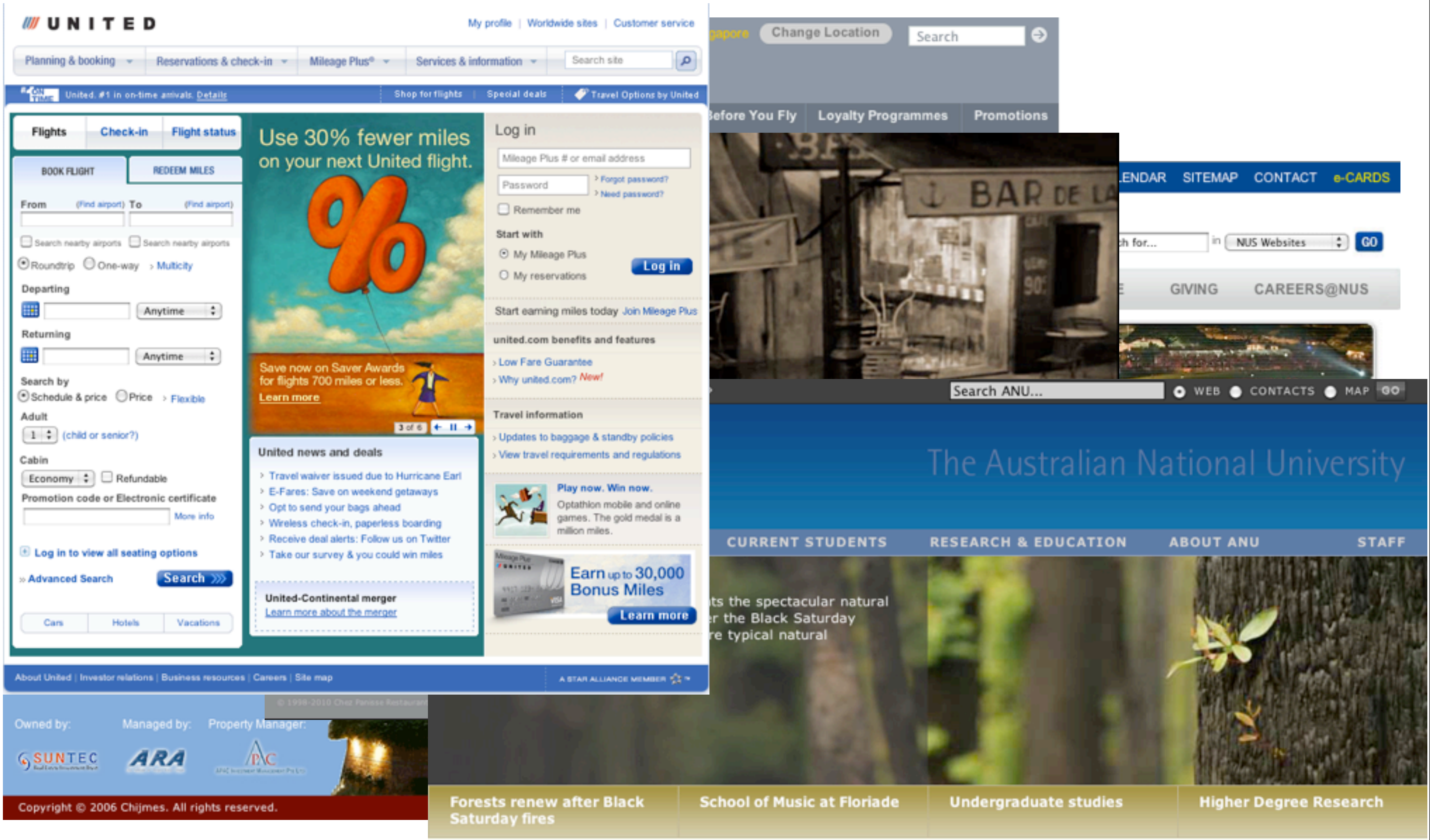
Latent Dirichlet Allocation; Blei, Ng, Jordan, JMLR 2003

Grouping objects

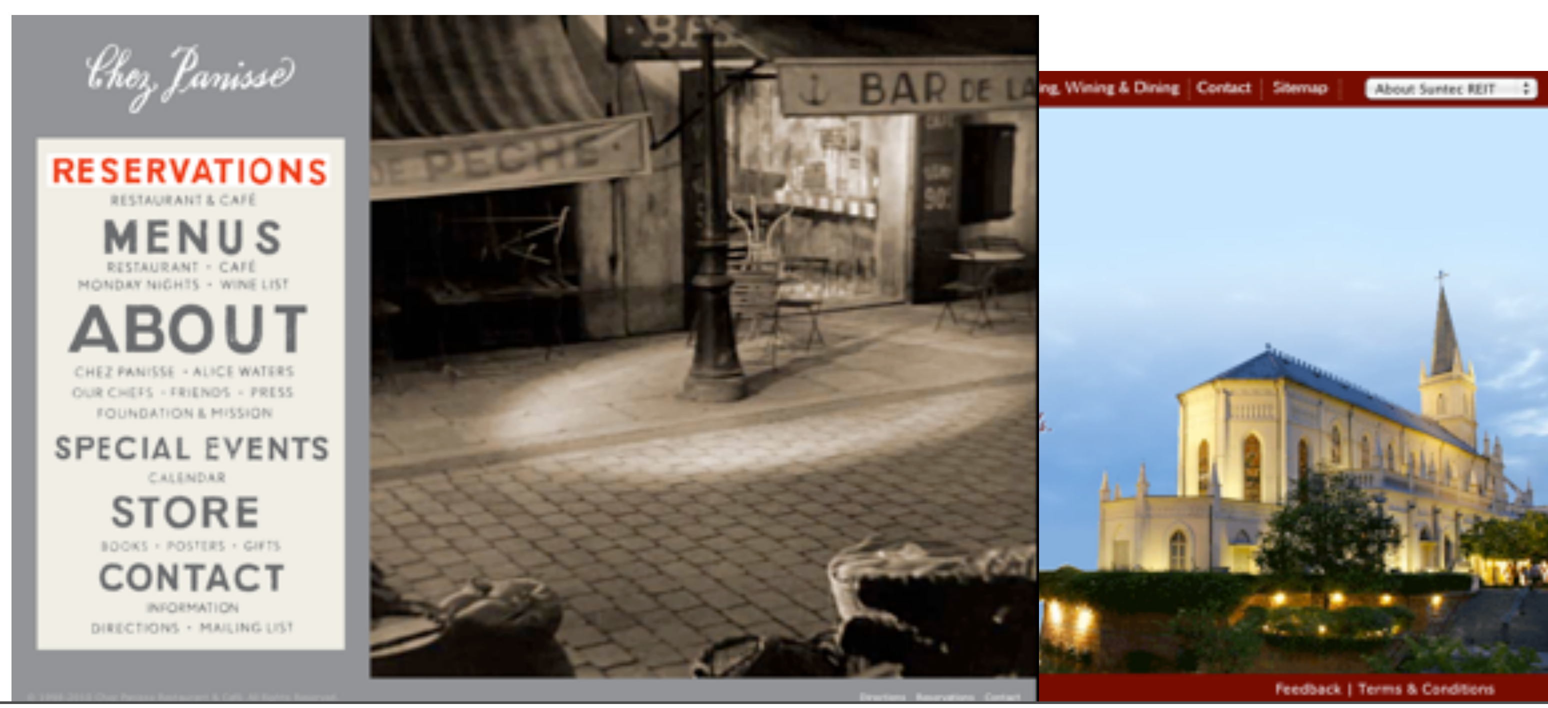
Grouping objects

YAHOO!

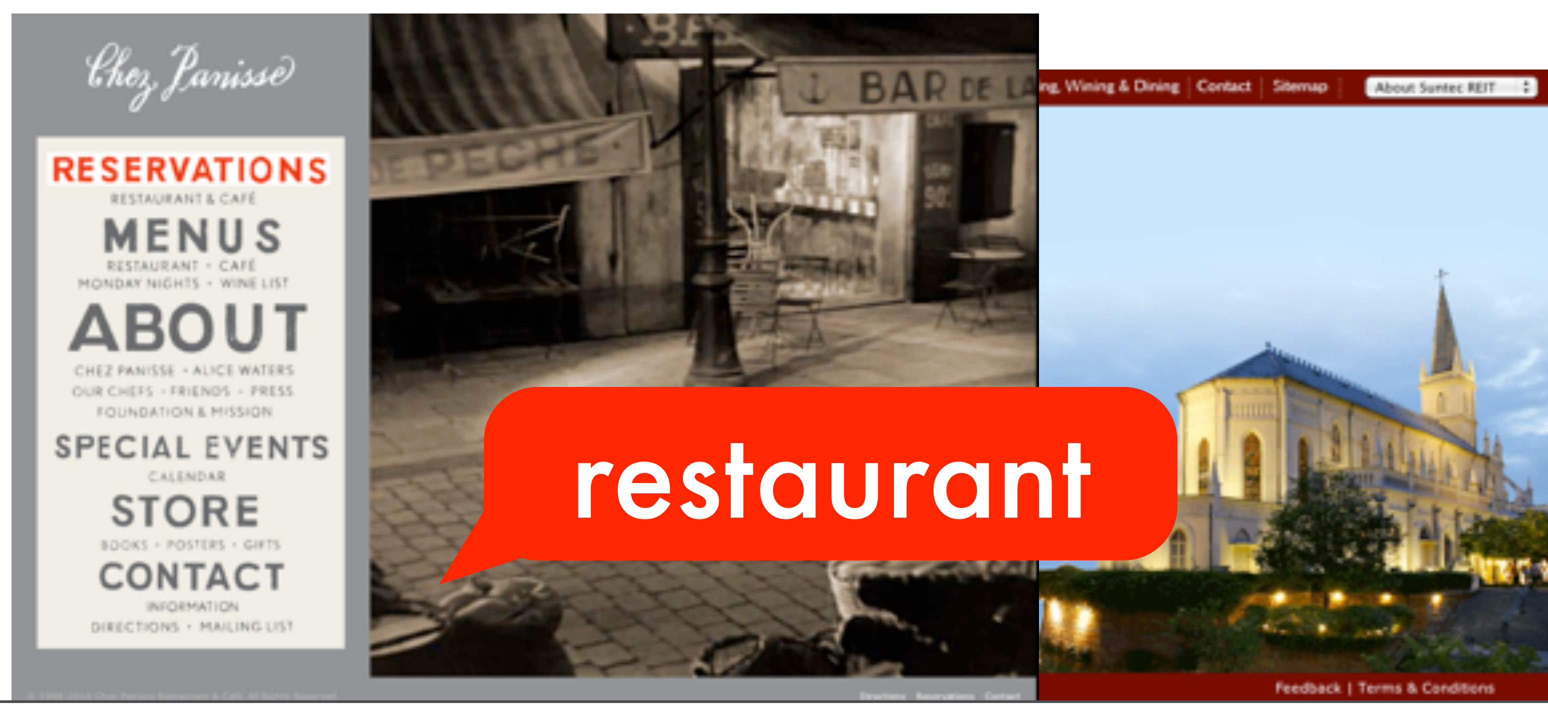
Grouping objects



Grouping objects



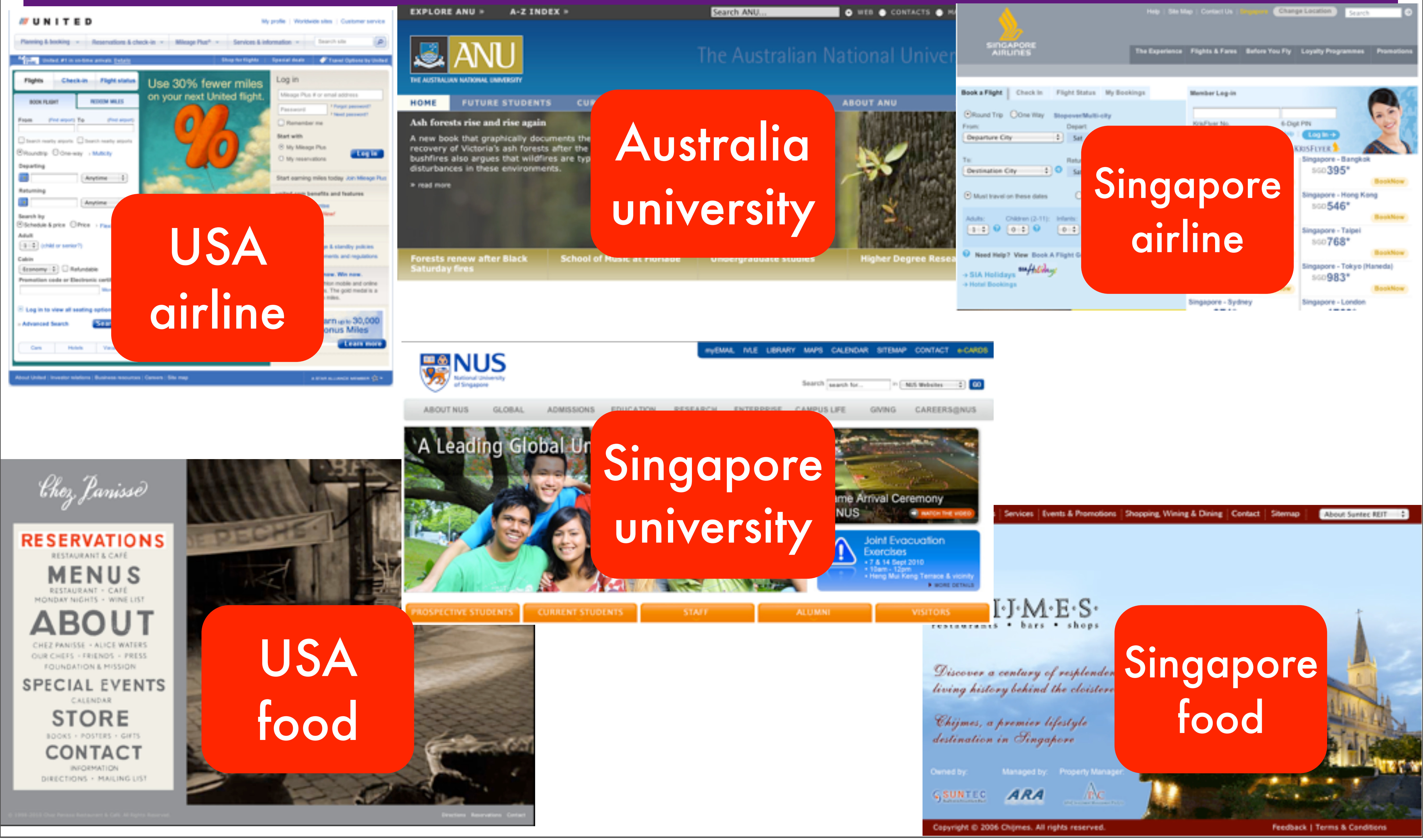
Grouping objects



YAHOO!

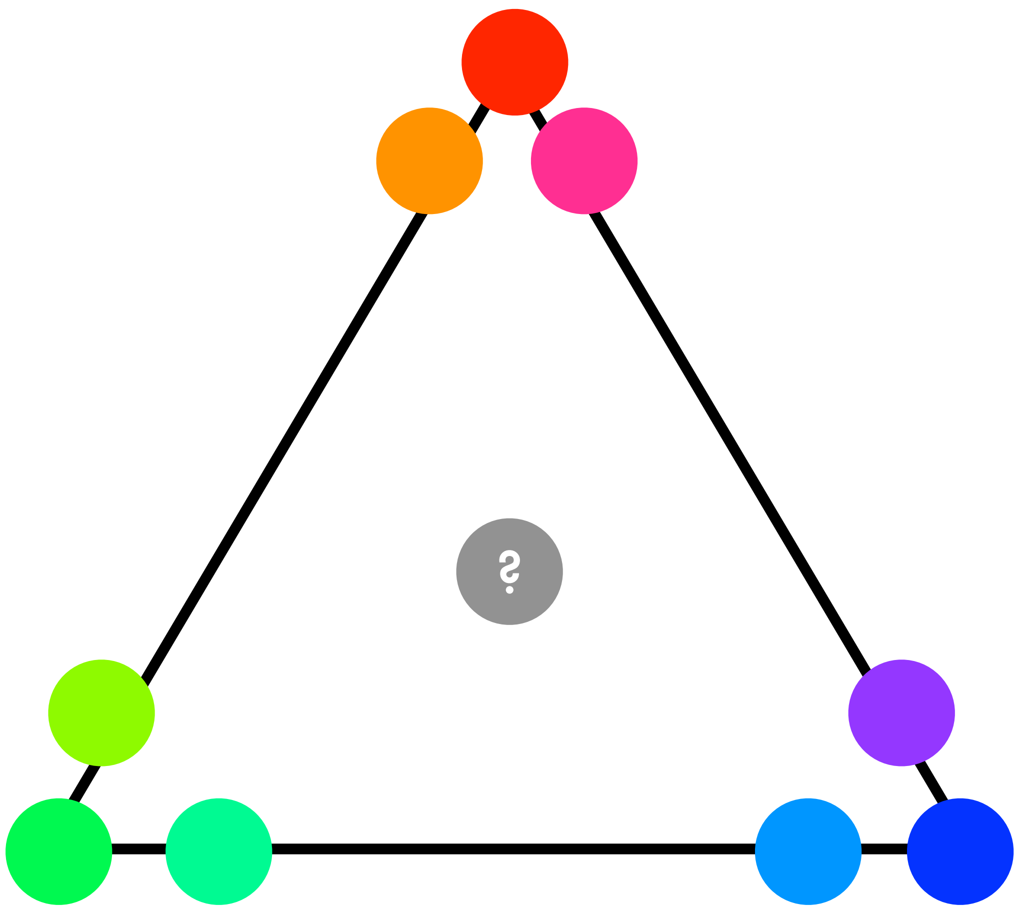
Grouping objects

Topic Models



Clustering & Topic Models

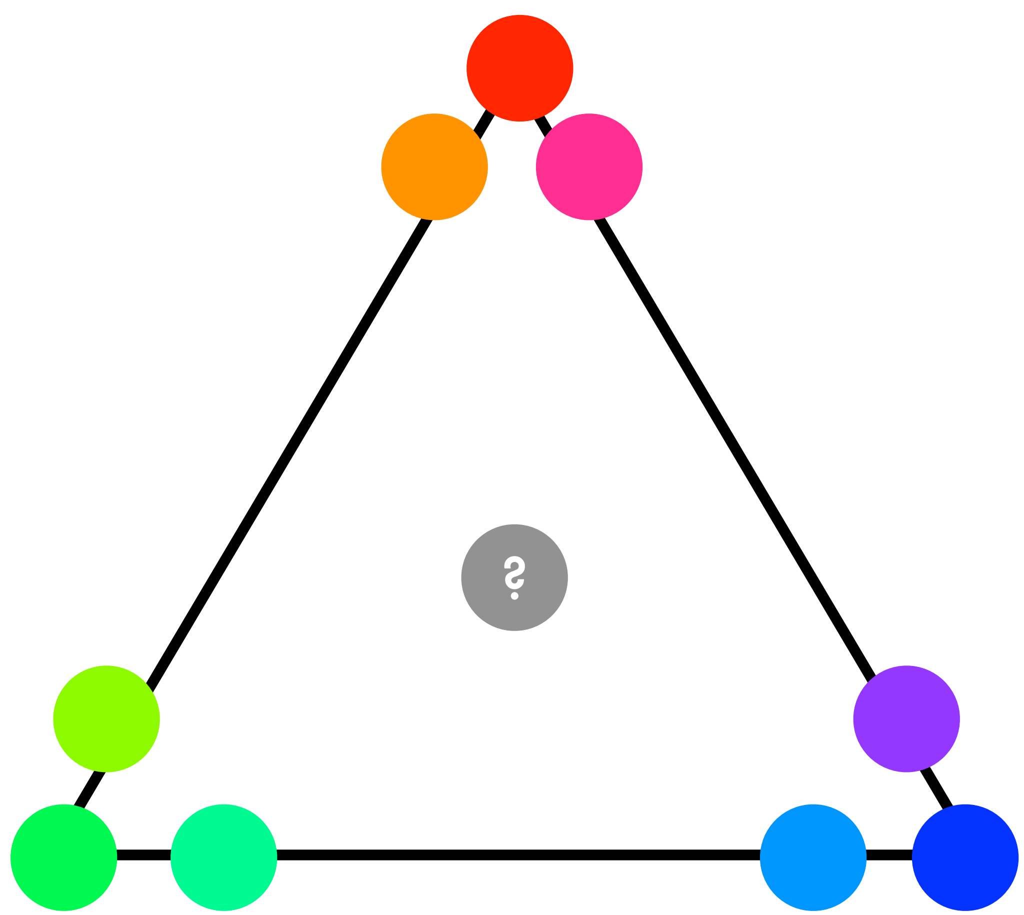
Clustering



group objects
by prototypes

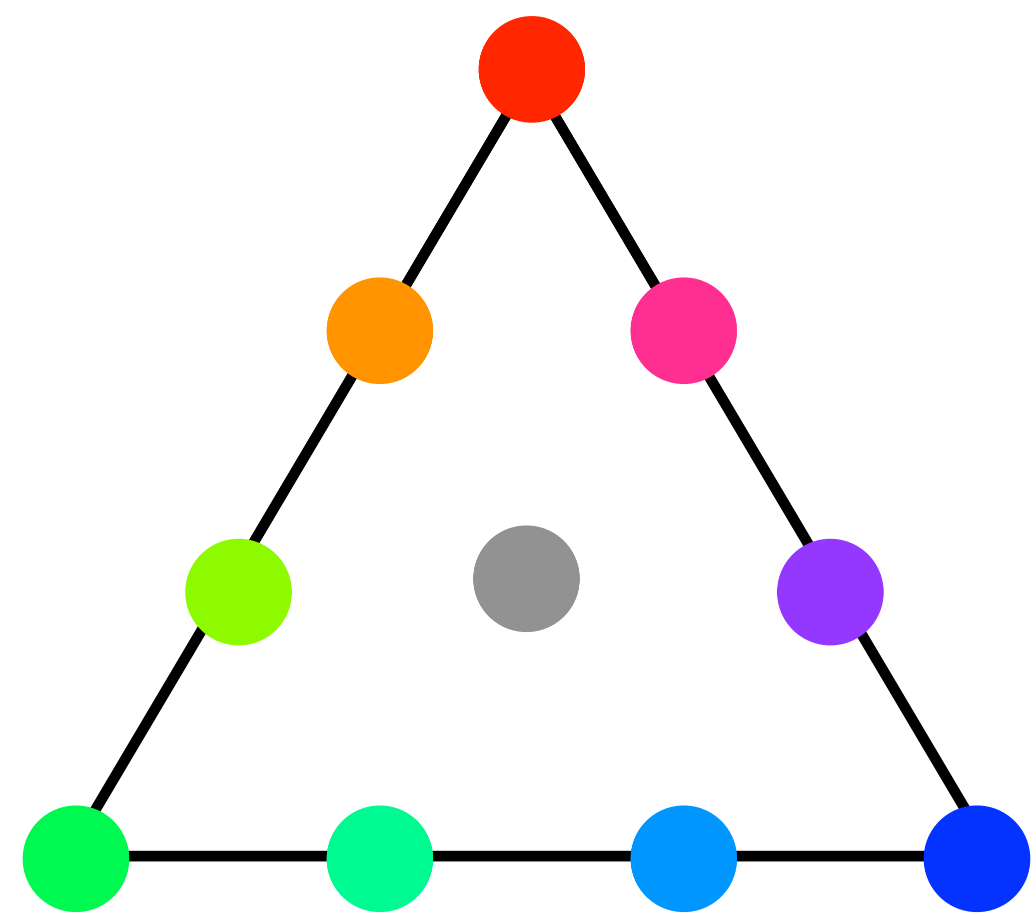
Clustering & Topic Models

Clustering



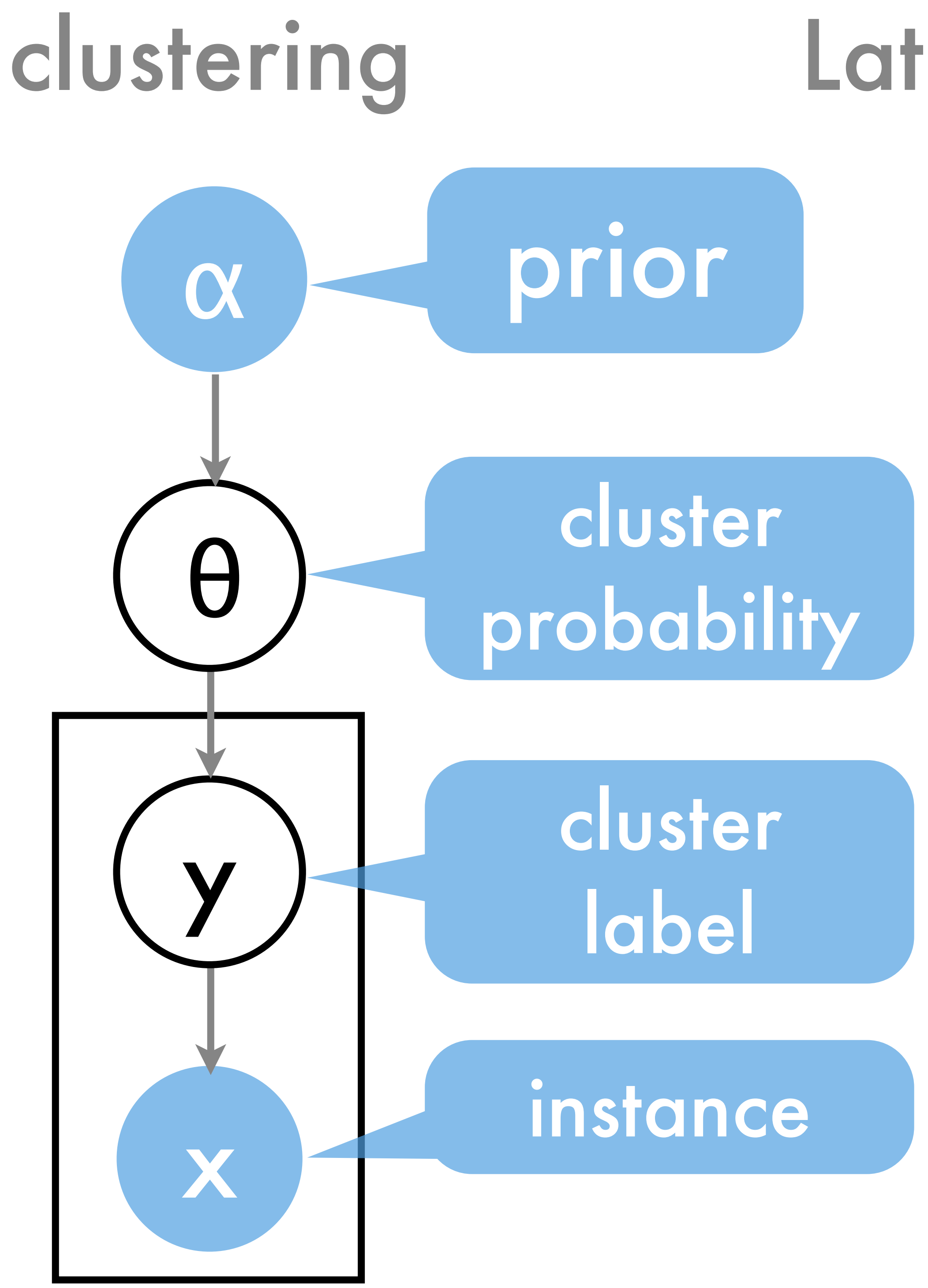
group objects
by prototypes

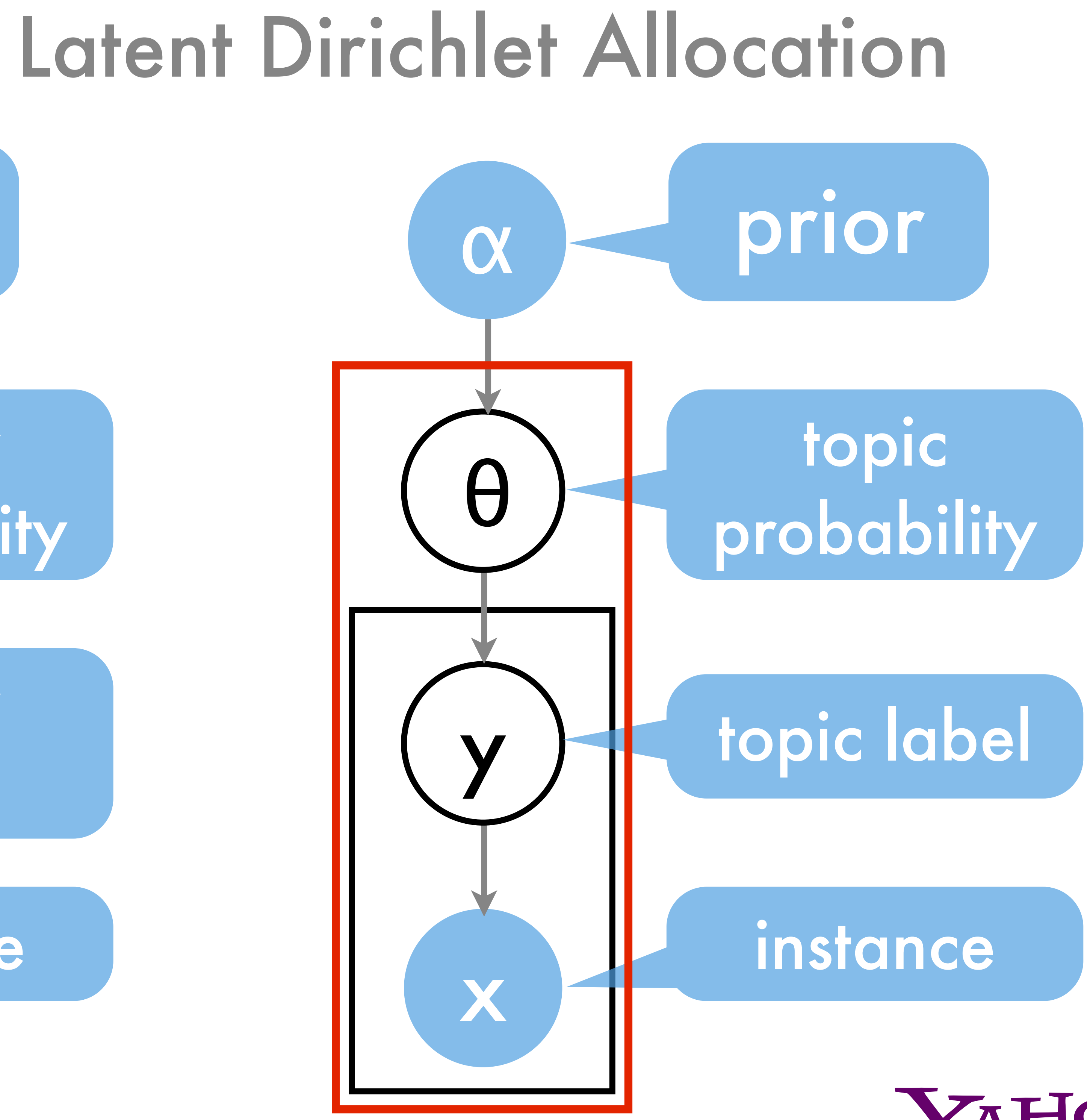
Topics



decompose objects
into prototypes

Clustering & Topic Models





YAHOO!

Clustering & Topic Models

Cluster/ topic distributions x membership = Documents

clustering: (0, 1) matrix
topic model: stochastic matrix
LSI: arbitrary matrices

Many more

- Regression
  inventory, traffic, reserve price, elasticity
- Novelty detection
  abuse, change in traffic, server farm
- Entity tagging
  keywords, named entities, segmentation
- Collaborative filtering
  recommend related movies, books, songs
- Inferring structure from data
  trees, DAGs, segmentation boundaries, user models

Optimization & inference problems (horrible oversimplification)

- Supervised problems

$$\underset{w}{\text{minimize}} \sum_{i=1}^{m} l(x_i, y_i, w) + \lambda \|w\|^{\alpha}$$

goodness of fit

complexity penalty

  - convex problem
  - solve subproblem and merge works well
- Unsupervised problems
  - nonconvex problem (looks similar)
  - fast synchronization required


OHIO ART
MAGIC Etch A Sketch® SCREEN
Systems
to run our algorithms on
2

Hardware

- NOT High Performance Computing

- Consumer hardware
  Cheap, efficient, not very reliable

The Joys of Real Hardware

Typical first year for a new cluster:

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)

~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)

~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)

~1 network rewiring (rolling ~5% of machines down over 2-day span)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~5 racks go wonky (40-80 machines see 50% packetloss)

~8 network maintenances (4 might cause ~30-minute random connectivity losses)

~12 router reloads (takes out DNS and external vips for a couple minutes)

~3 router failures (have to immediately pull traffic for an hour)

~dozens of minor 30-second blips for dns

~1000 individual machine failures

~thousands of hard drive failures

slow disks, bad memory, misconfigured machines, flaky machines, etc.

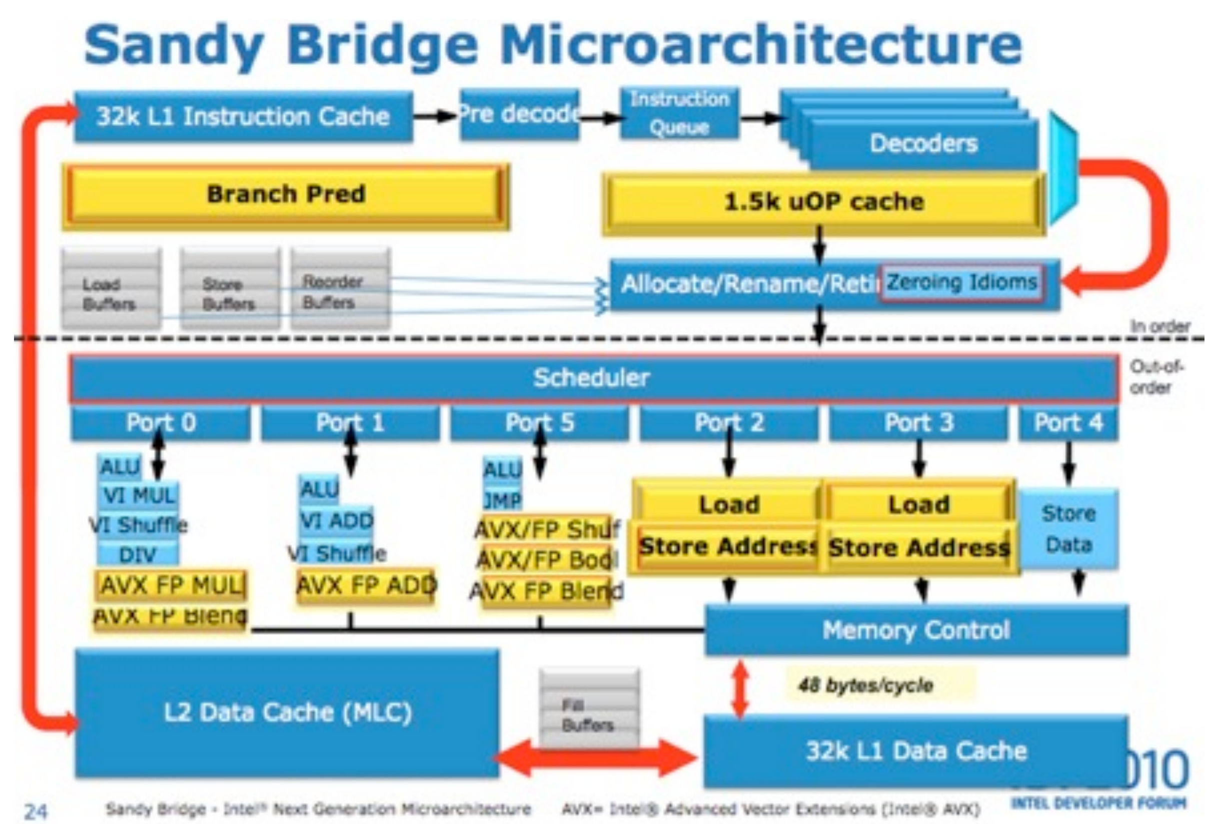
Slide from talk of Jeff Dean

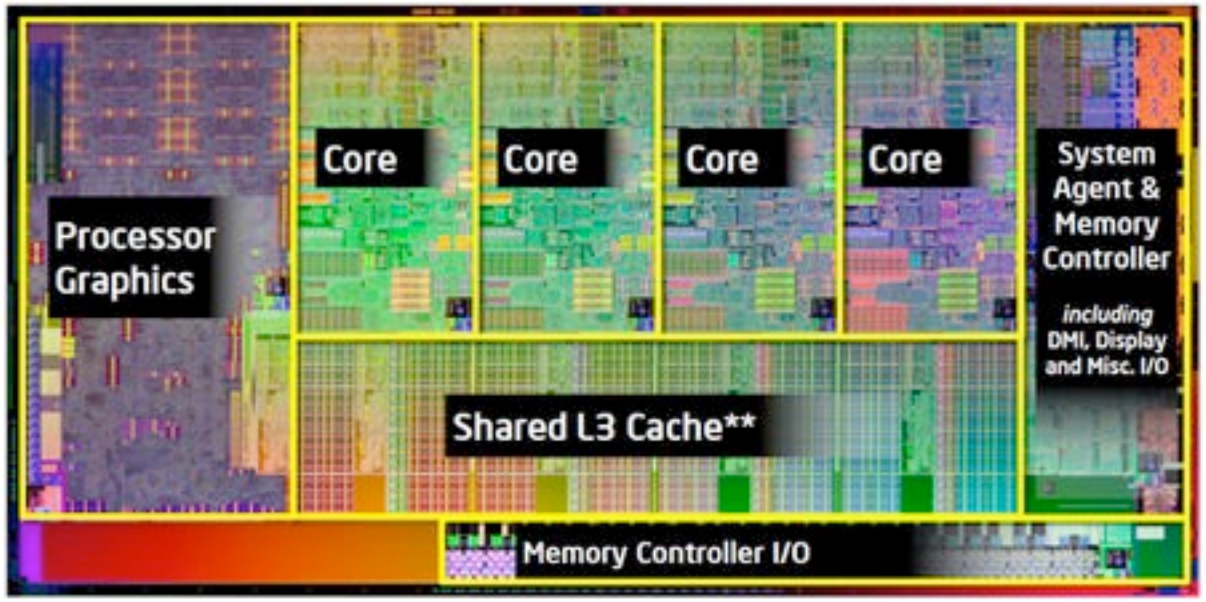
Google

http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en//people/jeff/stanford-295-talk.pdf

CPU

- 8-32 cores
- Memory interface 20-60GB/s
- Internal bandwidth >100GB/s
- >100 GFlops for matrix matrix multiply
- Integrated low end GPU





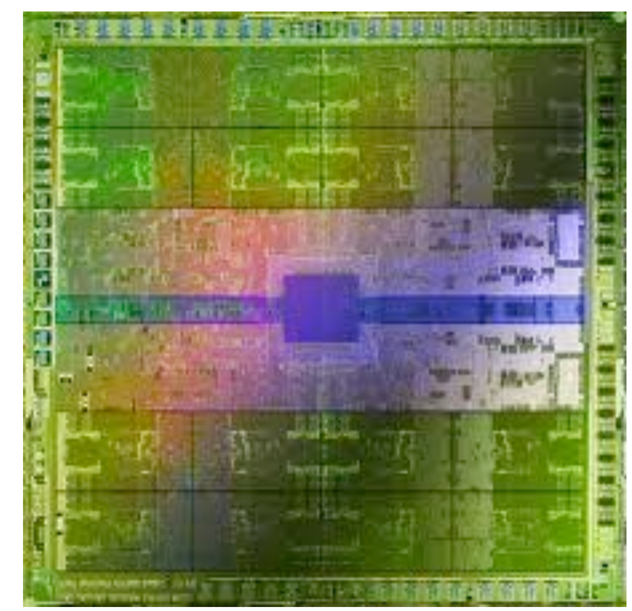
RAM

- High latency (100ns for DDR3)
- High burst data rate (>10 GB/s)

- Avoid random access in code if possible.
- Memory align variables
- Know your platform (FBDIMM vs. DDR)

http://www.anandtech.com/show/3851/everything-you-always-wanted-to-know-about-sdram-memory-but-were-afraid-to-ask

GPU



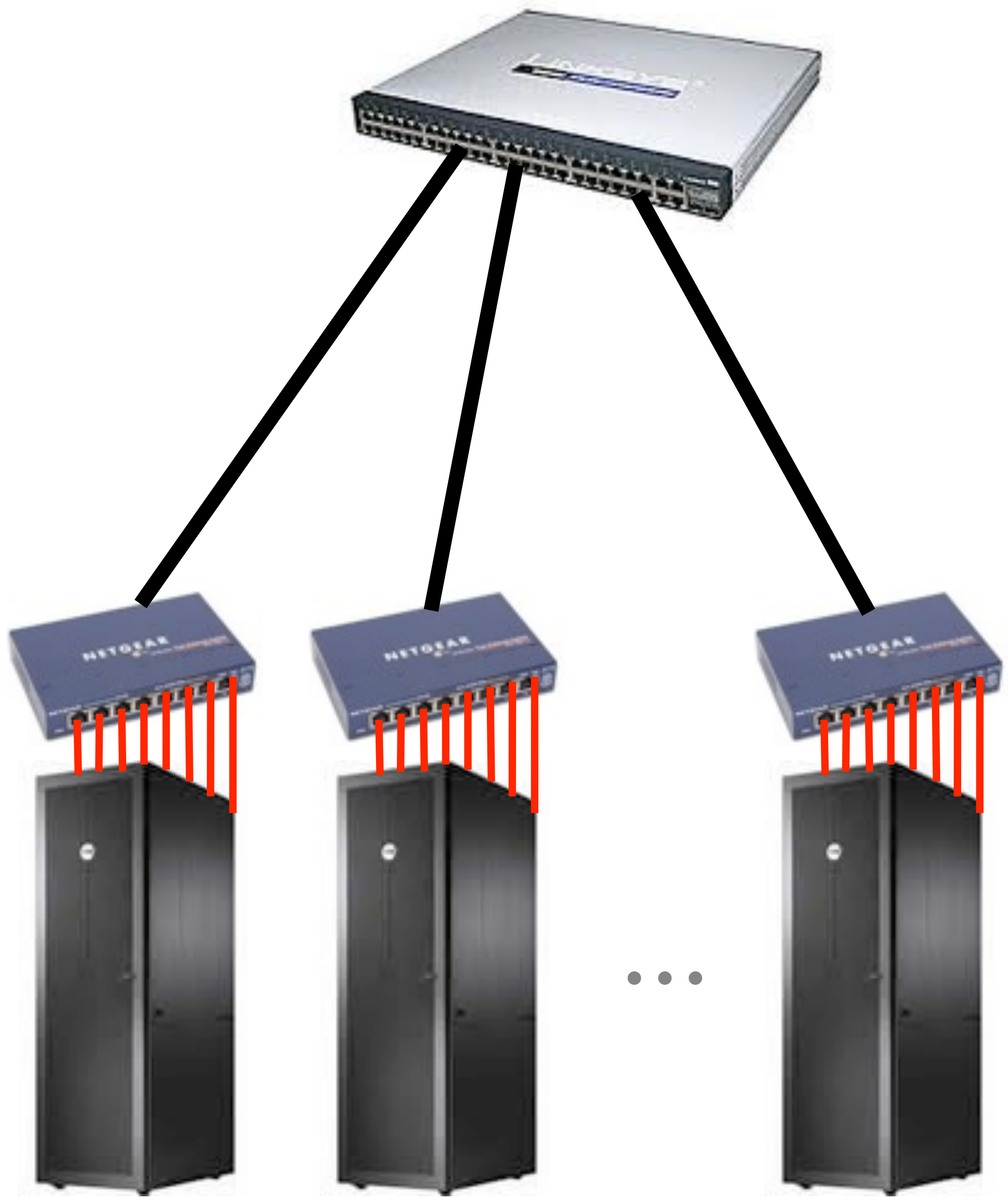
- Up to 512 cores / 200W
- Tricky to synchronize threads
- 1-3GB memory (Tesla 6GB)
- 1 TFlop
- Memory bandwidth > 100GB/s
- 4GB/s PCI bus bottleneck

Storage

- Harddisks
  - 3TB of storage (30MB/$)
  - 100 MB/s bandwidth (sequential)
  - 5 ms seek (200 IOPS)
- SSD
  - 100-500 MB storage (1MB/$)
  - 300 MB/s bandwidth (sequential)
  - 50,000 IOPS / 1 ms seek (queueing)

Switches & Colos

- Big switches are expensive
- Switches have finite buffers
  - many connections to single machine
  - dropped packets / collisions
- Hierarchical structure
  - more bandwidth within rack
  - lower latency within rack
  - lots of latency between colos



recent development on ‘flat’ networks

Numbers Everyone Should Know

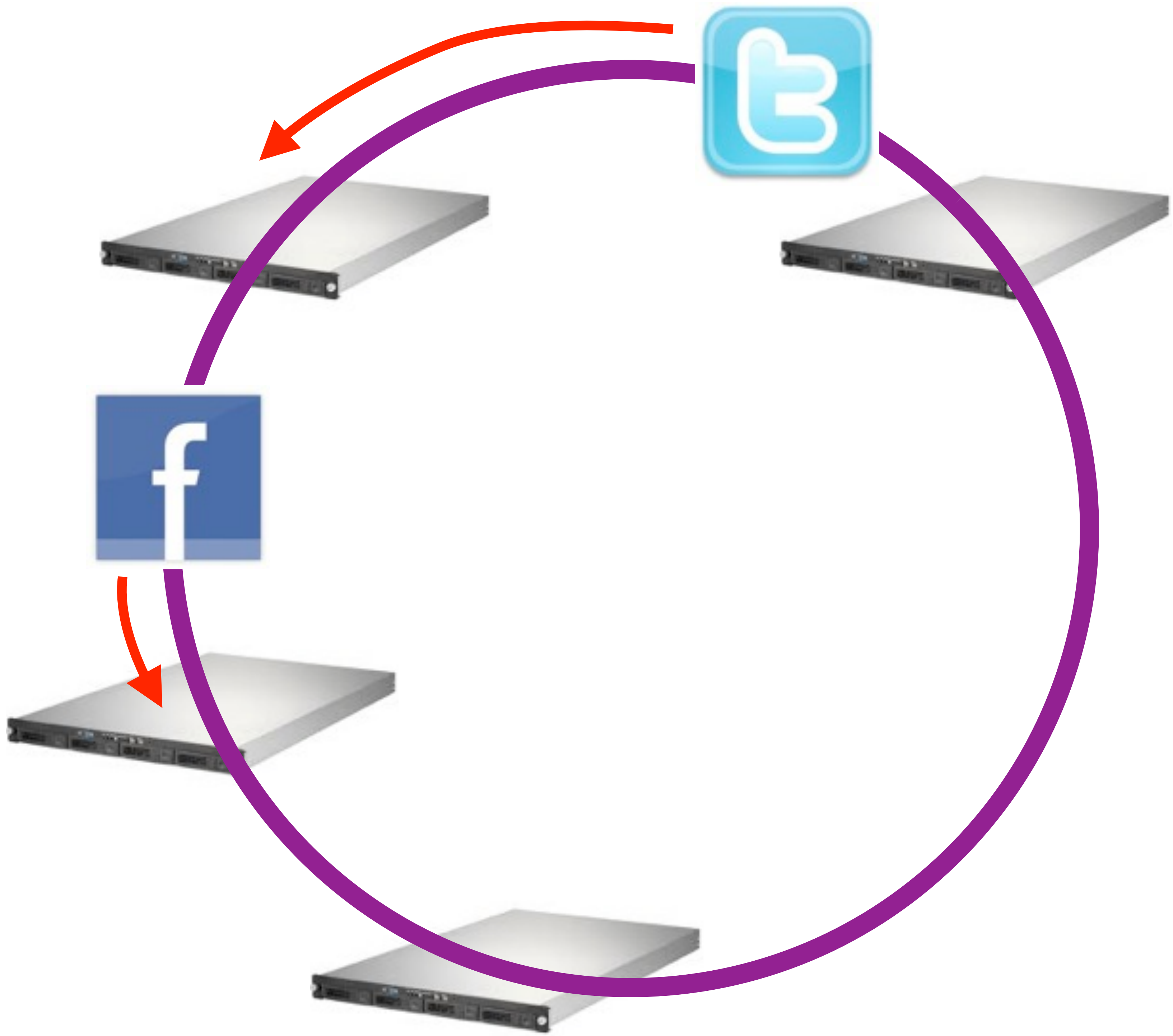
L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Slide from talk of Jeff Dean

Google

http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en//people/jeff/stanford-295-talk.pdf

Distribution and Balancing

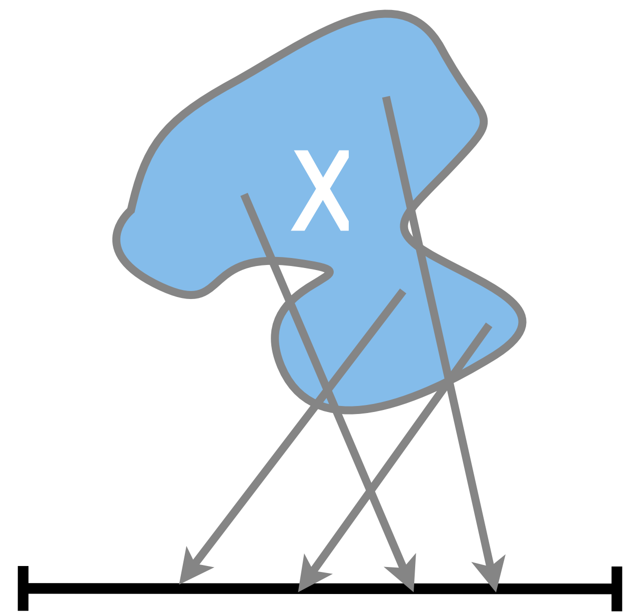


Concepts

- Large number of objects (a priori unknown)
- Large pool of machines (often faulty)
- Assign objects to machines such that
  - Object goes to the same machine (if possible)
  - Machines can be added/fail dynamically
- Consistent hashing (elements, sets, proportional)

symmetric (no master), dynamically scalable, fault tolerant

Hash function



- Mapping from domain X to integer range [1..N]
- Indistinguishable from uniform distribution
- n-ways independent hash function
  - Draw h from set hash functions H at random
  - For n instances in X their hash $[h(x_1), \ldots h(x_n)]$ is essentially indistinguishable from n random draws from [1 ... N]
- For many cases we only need 2-ways independence

$$\text{for all } x, y \quad \Pr_{y \in H} \{h(x) = h(y)\} = \frac{1}{N}$$

- In practice use MD5 or Murmur Hash for high quality
  https://code.google.com/p/smhasher/

Argmin Hash

- Consistent hashing

$$m(\text{key}) = \operatorname*{argmin}_{m \in \mathcal{M}} h(\text{key}, m)$$

  - Uniform distribution over machine pool M
  - Fully determined by hash function h. No need to ask master
  - If we add/remove machine m' all but O(1/m) keys remain

$$\Pr\left\{m(\text{key}) = m'\right\} = \frac{1}{m}$$

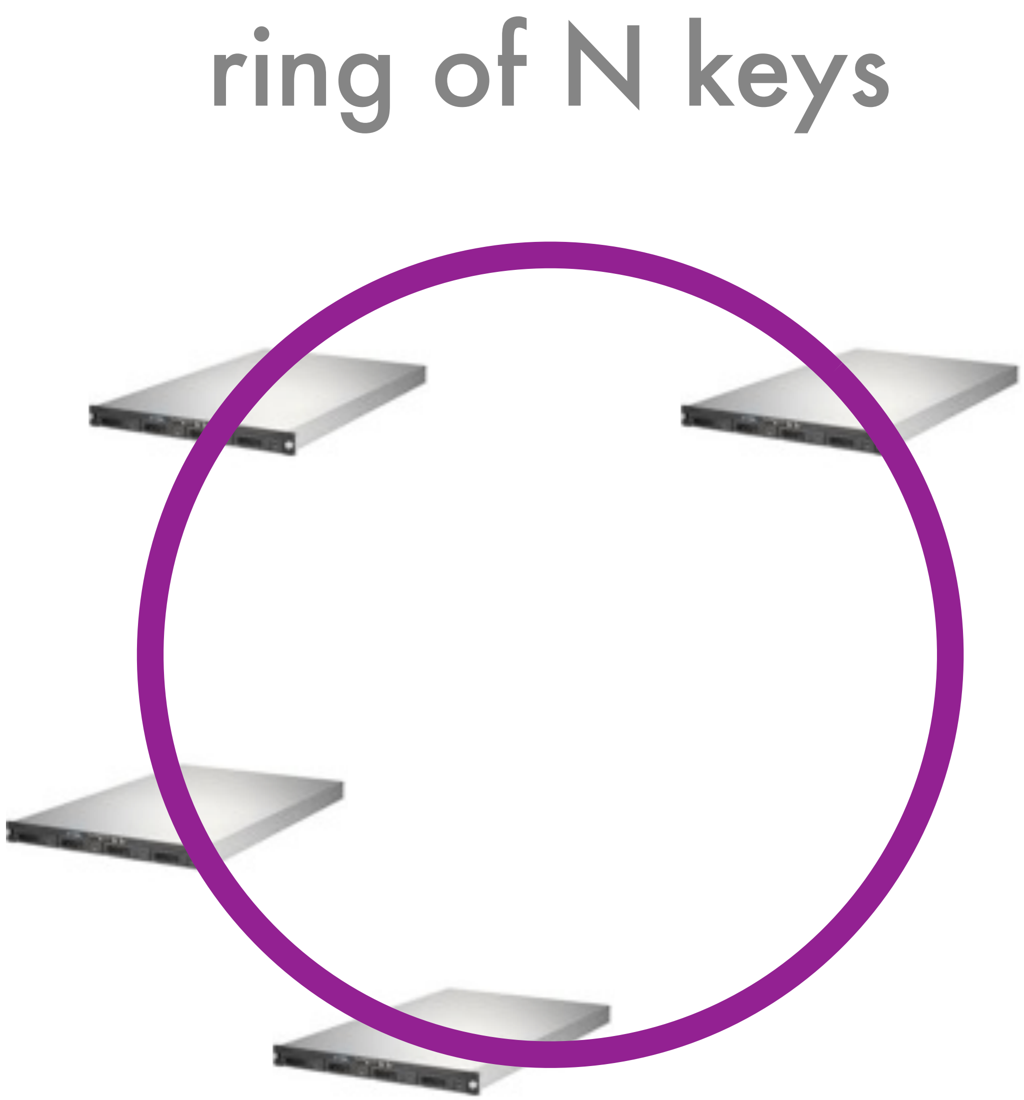
- Consistent hashing with k replications

$$m(\text{key}, k) = k \operatorname*{smallest}_{m \in \mathcal{M}} h(\text{key}, m)$$

  - If we add/remove a machine only O(k/m) need reassigning
- Cost to assign is O(m). This can be expensive for 1000 servers

Distributed Hash Table

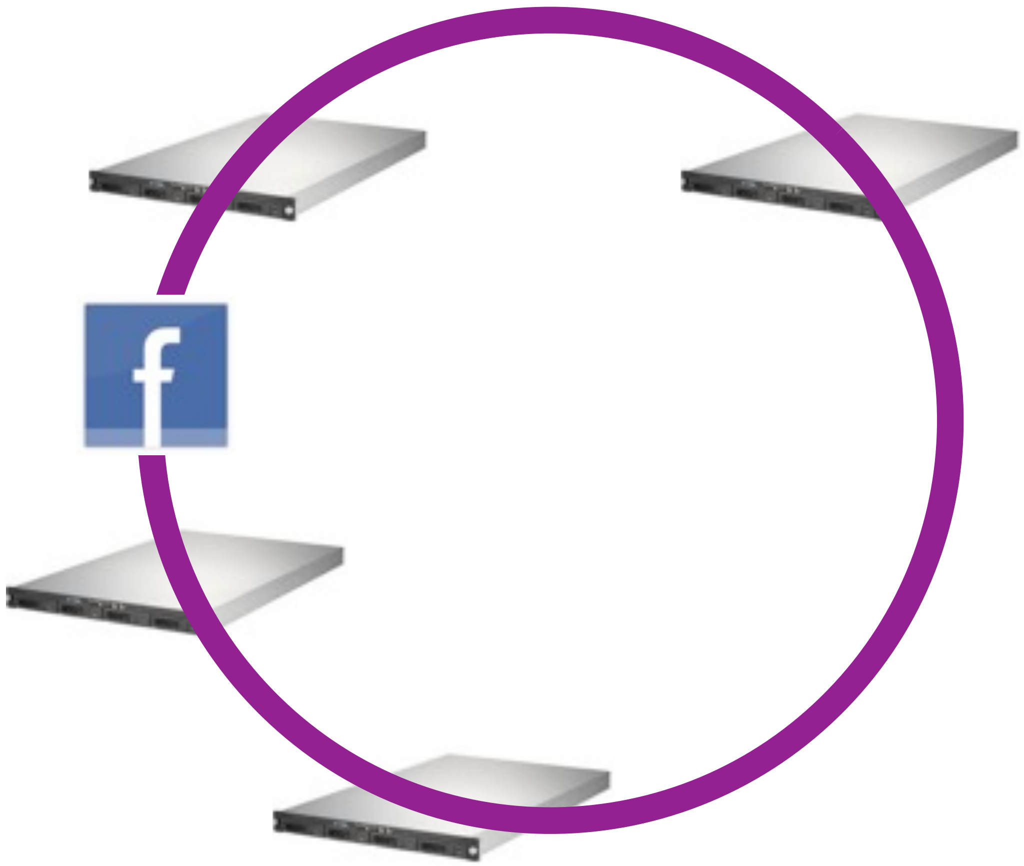
- Fixing the O(m) lookup
  - Assign machines to ring via hash h(m)
  - Assign keys to ring
  - Pick machine nearest to key to the left
- O(log m) lookup
- Insert/removal only affects neighbor
  (however, big problem for neighbor)
- Uneven load distribution
  (load depends on segment size)
- Insert machine more than once to fix this
- For k term replication, simply pick the k leftmost machines (skip duplicates)



Distributed Hash Table

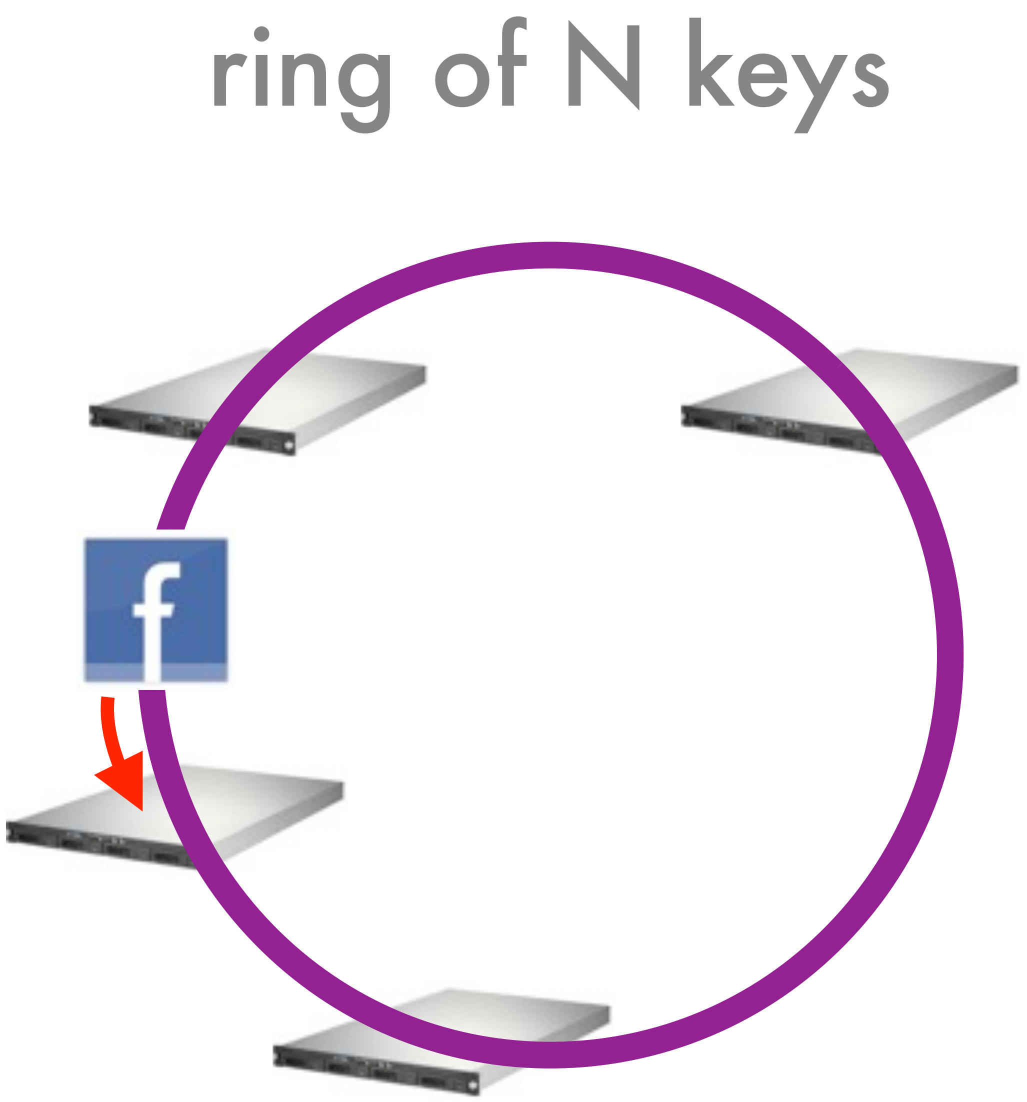
- Fixing the O(m) lookup
  - Assign machines to ring via hash h(m)
  - Assign keys to ring
  - Pick machine nearest to key to the left
- O(log m) lookup
- Insert/removal only affects neighbor
  (however, big problem for neighbor)
- Uneven load distribution
  (load depends on segment size)
- Insert machine more than once to fix this
- For k term replication, simply pick the k leftmost machines (skip duplicates)

ring of N keys



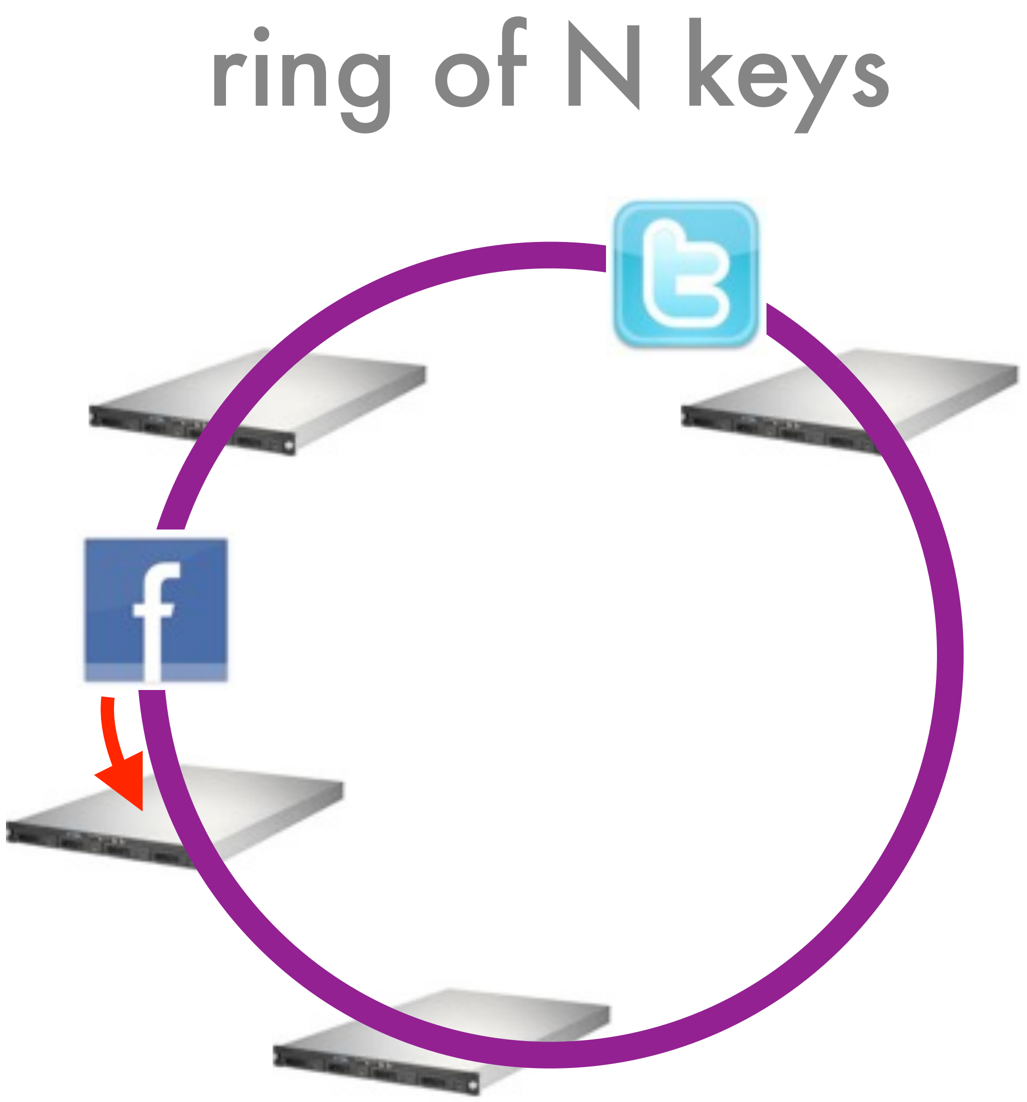
Distributed Hash Table

- Fixing the O(m) lookup
  - Assign machines to ring via hash h(m)
  - Assign keys to ring
  - Pick machine nearest to key to the left
- O(log m) lookup
- Insert/removal only affects neighbor (however, big problem for neighbor)
- Uneven load distribution (load depends on segment size)
- Insert machine more than once to fix this
- For k term replication, simply pick the k leftmost machines (skip duplicates)



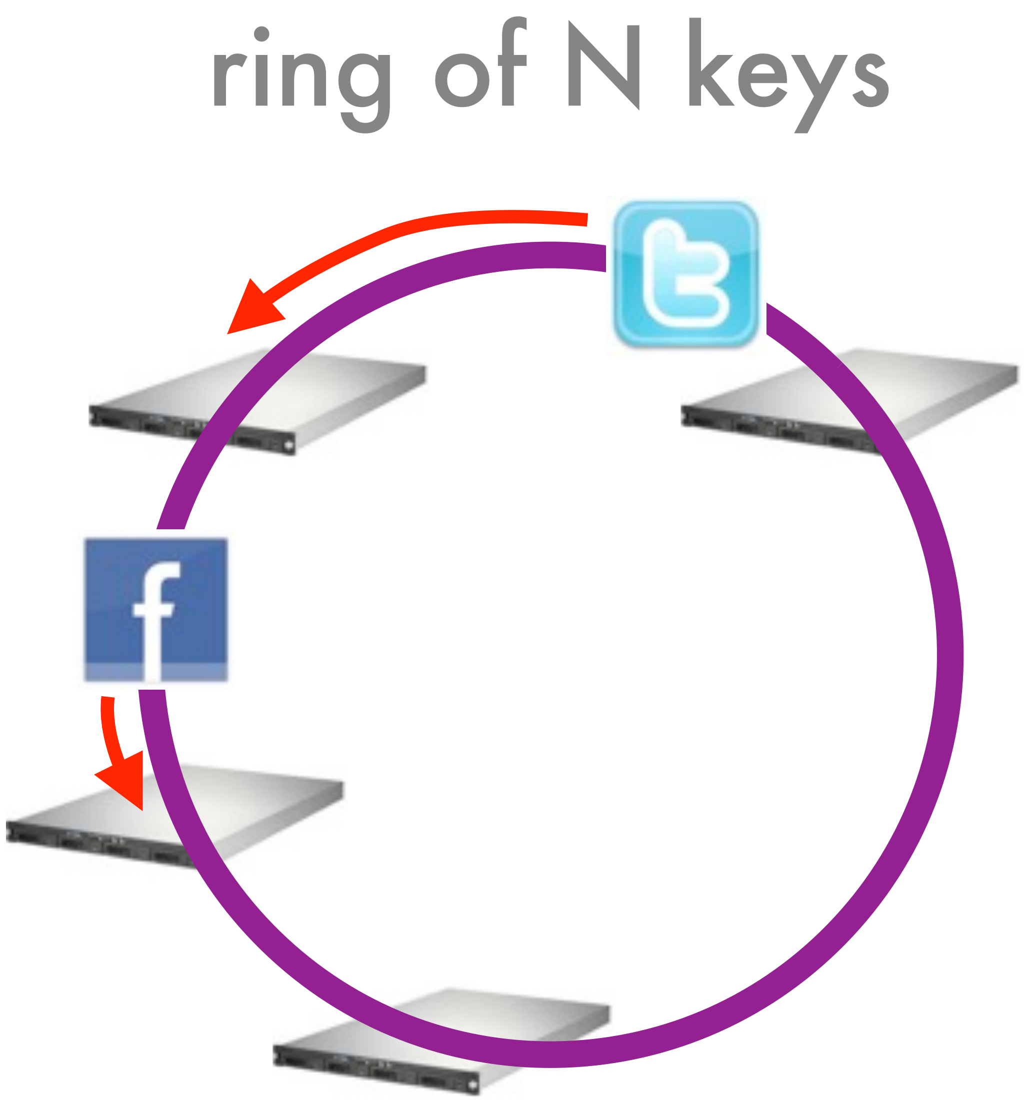
Distributed Hash Table

- Fixing the O(m) lookup
  - Assign machines to ring via hash h(m)
  - Assign keys to ring
  - Pick machine nearest to key to the left
- O(log m) lookup
- Insert/removal only affects neighbor
  (however, big problem for neighbor)
- Uneven load distribution
  (load depends on segment size)
- Insert machine more than once to fix this
- For k term replication, simply pick the k leftmost machines (skip duplicates)



Distributed Hash Table

- Fixing the O(m) lookup
  - Assign machines to ring via hash h(m)
  - Assign keys to ring
  - Pick machine nearest to key to the left
- O(log m) lookup
- Insert/removal only affects neighbor
  (however, big problem for neighbor)
- Uneven load distribution
  (load depends on segment size)
- Insert machine more than once to fix this
- For k term replication, simply pick the k leftmost machines (skip duplicates)



D2 - Distributed Hash Table

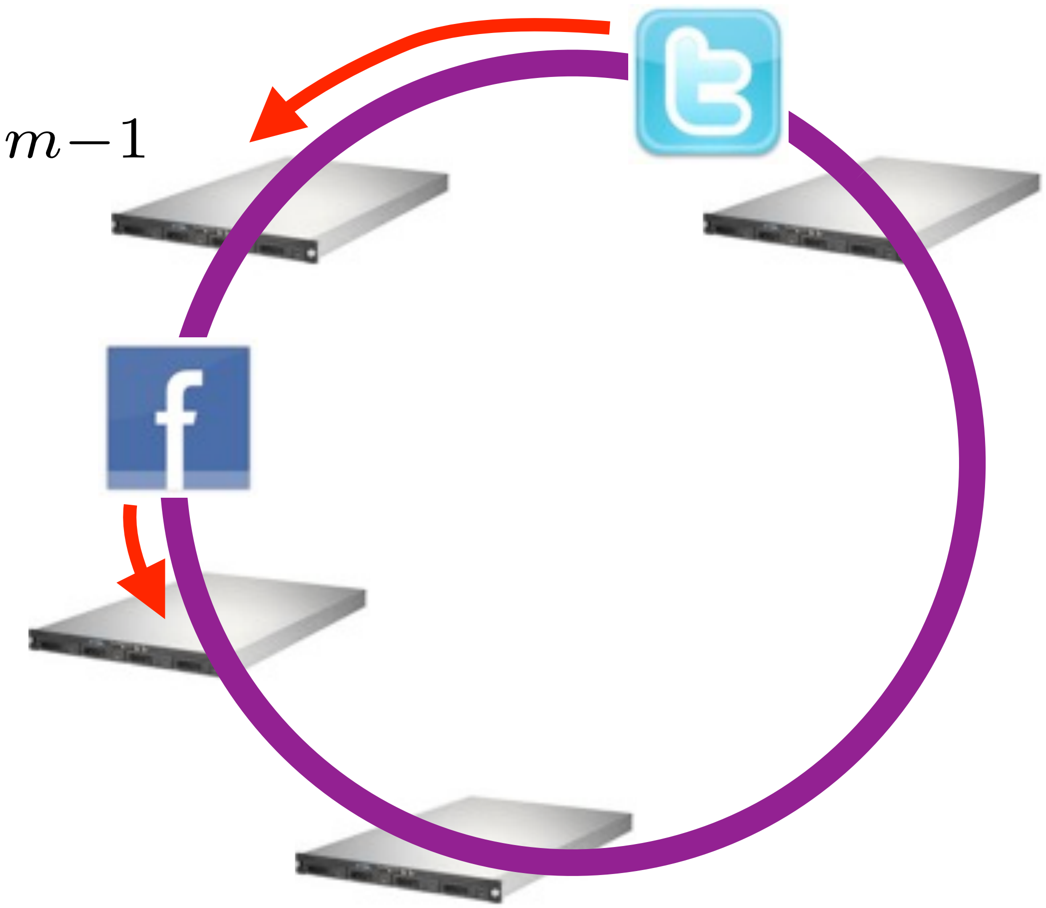
- For arbitrary node segment size is minimum over (m-1) independent uniformly distributed random variables

$$\Pr\{x \geq c\} = \prod_{i=2}^{m} \Pr\{s_i \geq c\} = (1-c)^{m-1}$$

- Density is given by derivative
$$p(c) = (m-1)(1-c)^{m-2}$$
- Expected segment length is $c = \frac{1}{m}$ (follows from symmetry)
- Probability of exceeding expected segment length (for large m)

$$\Pr\left\{x \geq \frac{k}{m}\right\} = \left(1 - \frac{k}{m}\right)^{m-1} \longrightarrow e^{-k}$$

ring of N keys

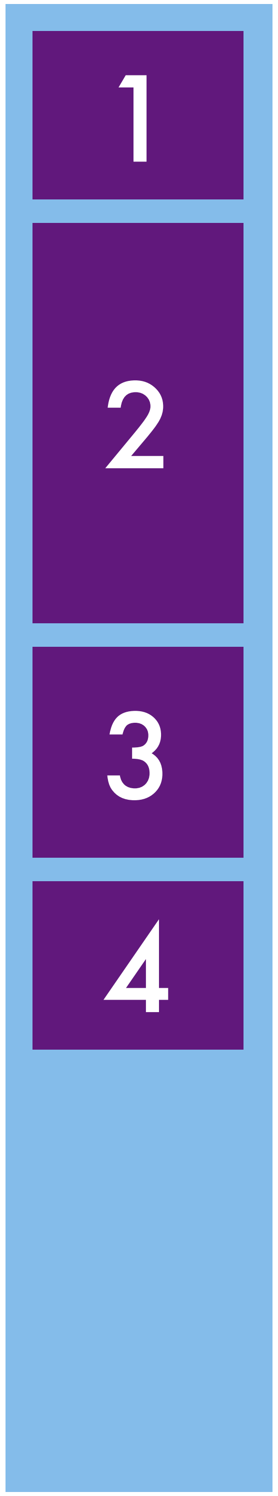


Proportional Allocation Table

- Assign items according to machine capacity

- Create allocation table with segments proportional to capacity
- Leave space for additional machines
- Hash key h(x) and pick machine covering it
- If failure, re-hash the hash until it hits a bin
- For replication hit k bins in a row

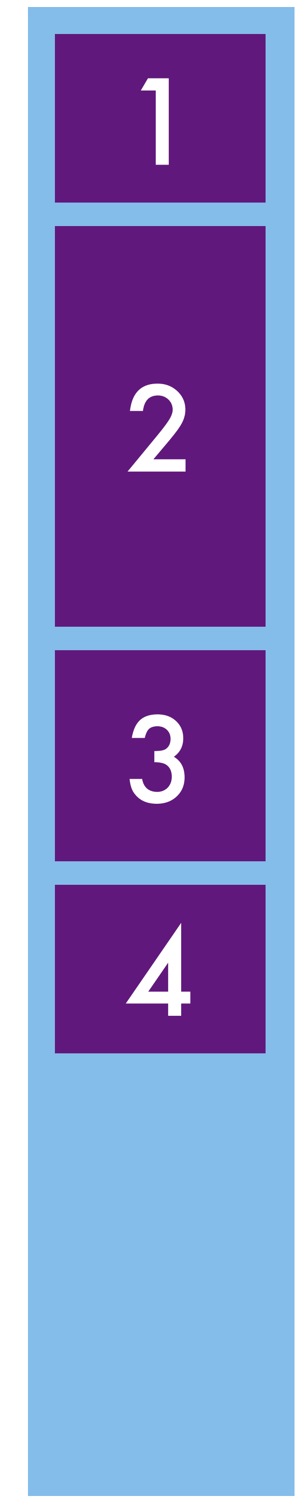
- Proportional load distribution
- Limited scalability
- Need to distribute and update table
- Limit peak load by further delegation (SPOCA - Chawla et al., USENIX 2011)



Proportional Allocation Table

- Assign items according to machine capacity

- Create allocation table with segments proportional to capacity
- Leave space for additional machines
- Hash key h(x) and pick machine covering it
- If failure, re-hash the hash until it hits a bin
- For replication hit k bins in a row



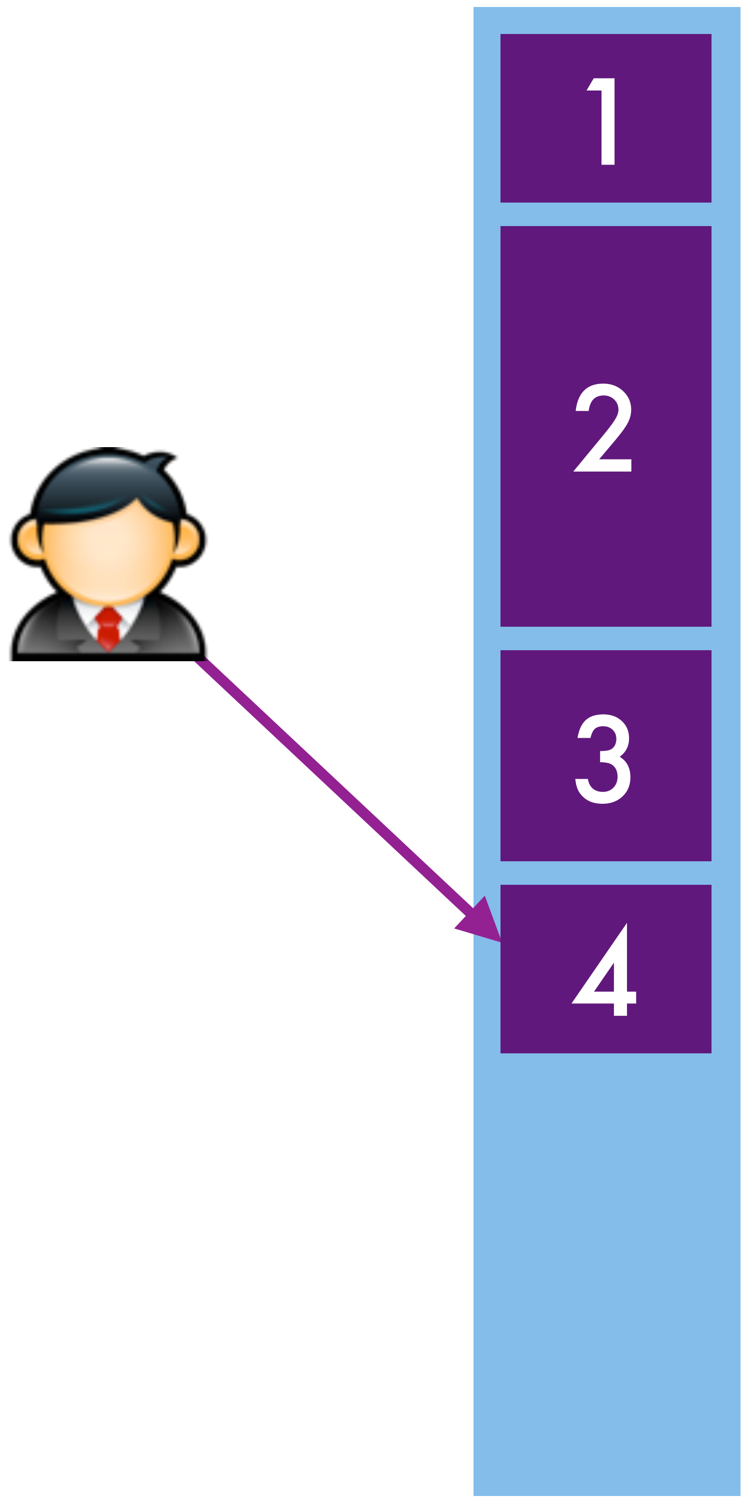
- Proportional load distribution
- Limited scalability
- Need to distribute and update table
- Limit peak load by further delegation (SPOCA - Chawla et al., USENIX 2011)

Proportional Allocation Table

- Assign items according to machine capacity

- Create allocation table with segments proportional to capacity
- Leave space for additional machines
- Hash key h(x) and pick machine covering it
- If failure, re-hash the hash until it hits a bin
- For replication hit k bins in a row

- Proportional load distribution
- Limited scalability
- Need to distribute and update table
- Limit peak load by further delegation (SPOCA - Chawla et al., USENIX 2011)

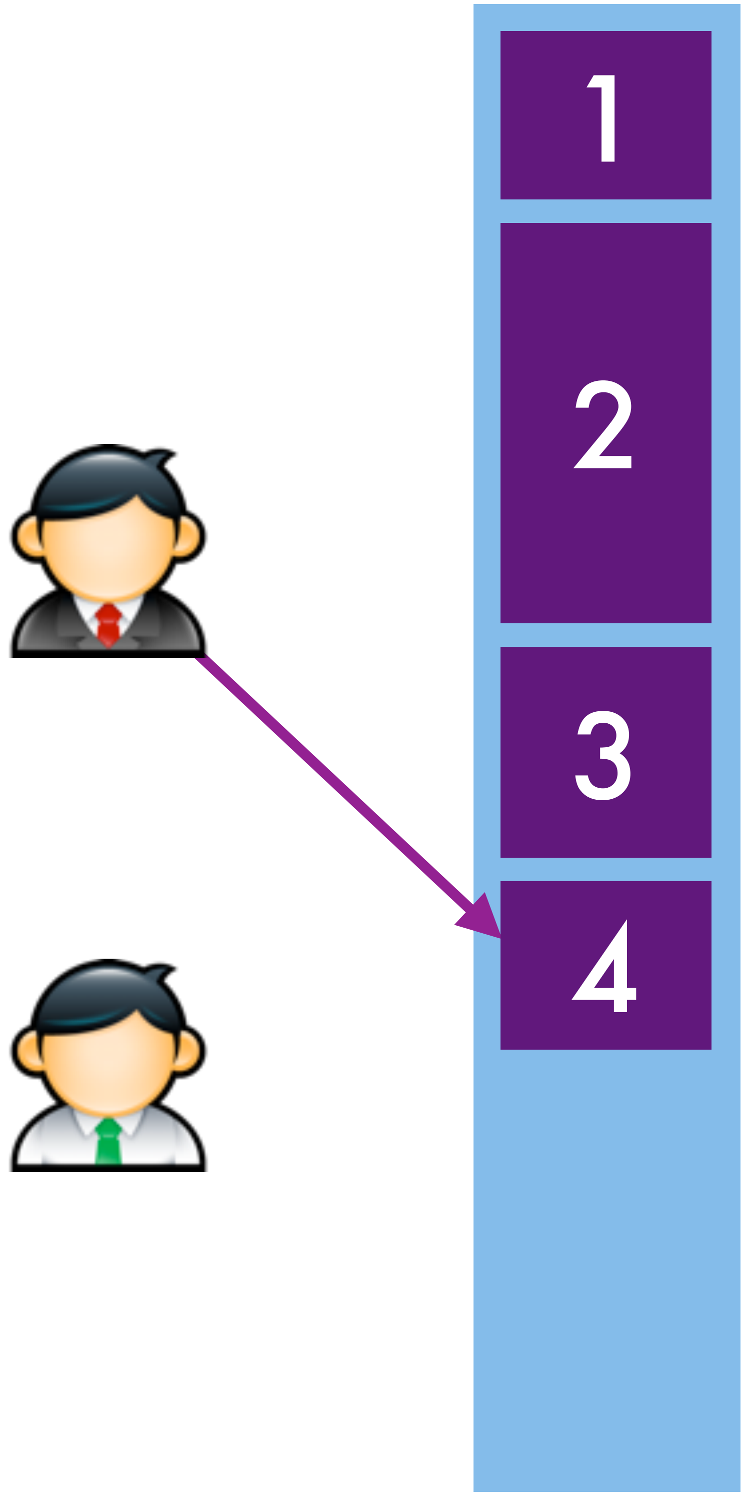


Proportional Allocation Table

- Assign items according to machine capacity

- Create allocation table with segments proportional to capacity
- Leave space for additional machines
- Hash key h(x) and pick machine covering it
- If failure, re-hash the hash until it hits a bin
- For replication hit k bins in a row

- Proportional load distribution
- Limited scalability
- Need to distribute and update table
- Limit peak load by further delegation (SPOCA - Chawla et al., USENIX 2011)

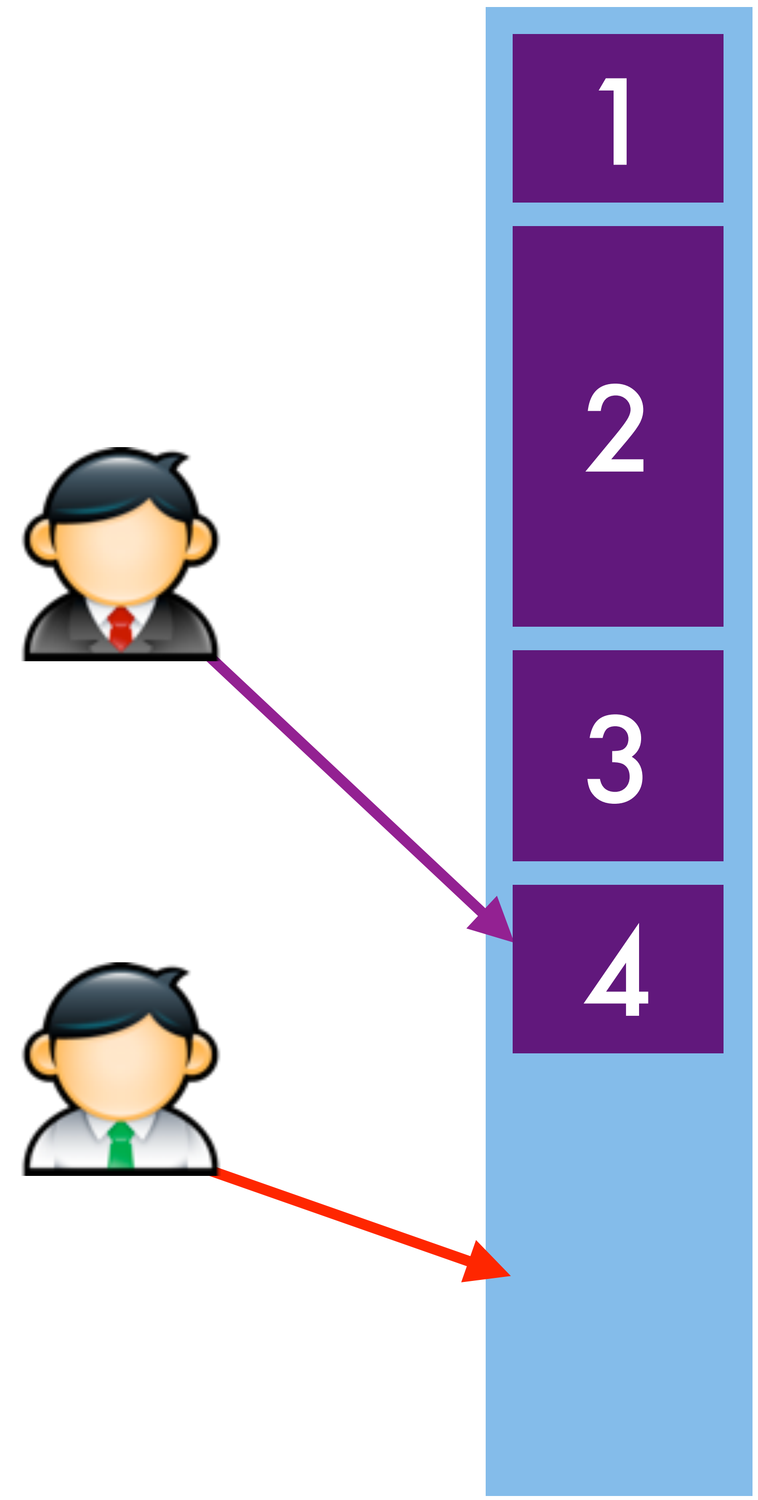


Proportional Allocation Table

- Assign items according to machine capacity

- Create allocation table with segments proportional to capacity
- Leave space for additional machines
- Hash key h(x) and pick machine covering it
- If failure, re-hash the hash until it hits a bin
- For replication hit k bins in a row

- Proportional load distribution
- Limited scalability
- Need to distribute and update table
- Limit peak load by further delegation (SPOCA - Chawla et al., USENIX 2011)

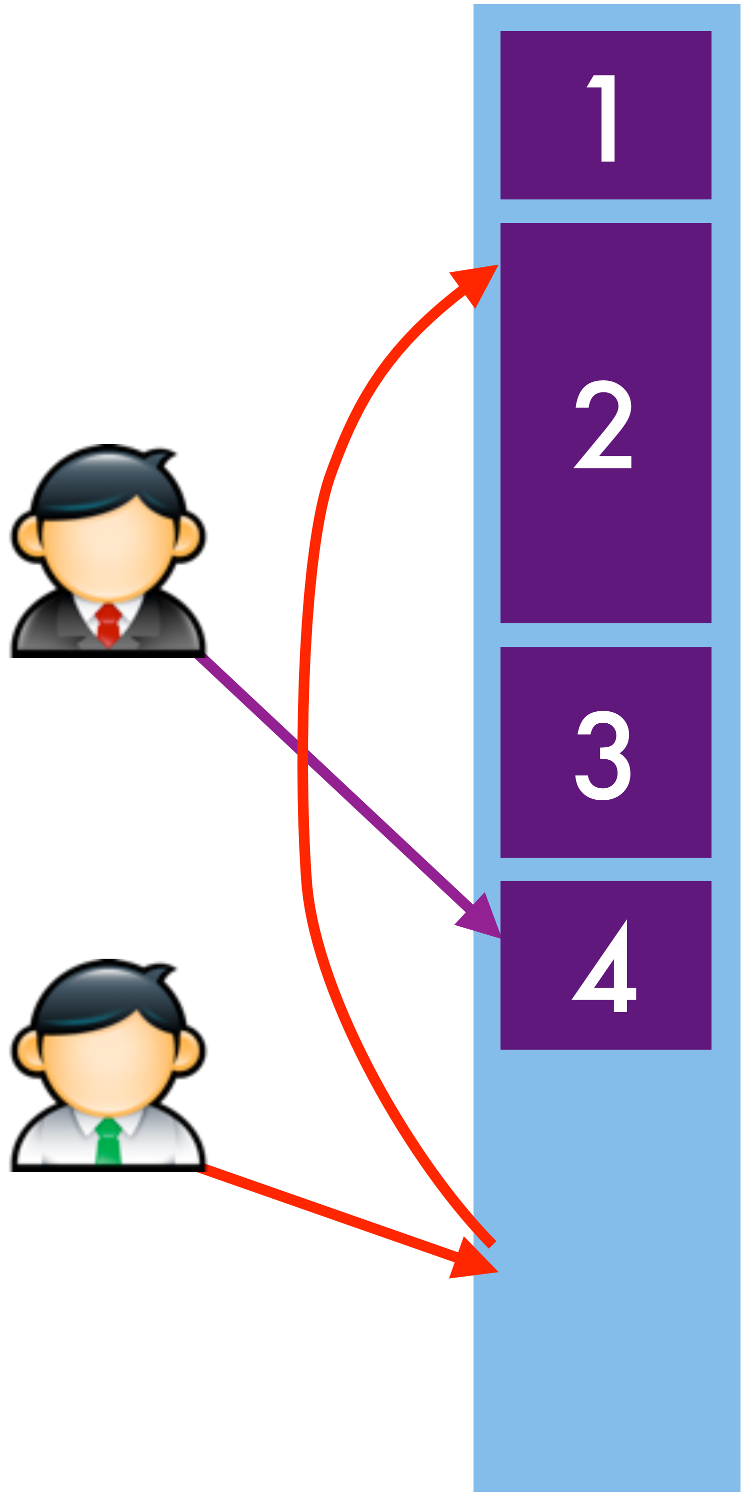


Proportional Allocation Table

- Assign items according to machine capacity

- Create allocation table with segments proportional to capacity
- Leave space for additional machines
- Hash key h(x) and pick machine covering it
- If failure, re-hash the hash until it hits a bin
- For replication hit k bins in a row

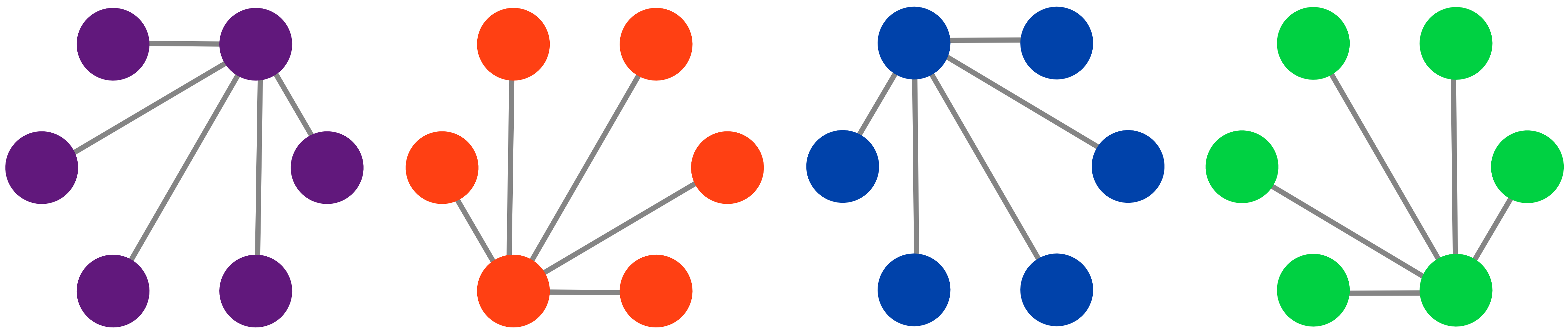
- Proportional load distribution
- Limited scalability
- Need to distribute and update table
- Limit peak load by further delegation (SPOCA - Chawla et al., USENIX 2011)



Random Caching Trees (Karger et al. 1999, Akamai paper)

- Cache / synchronize an object
- Uneven load distribution
- Must not generate hotspot

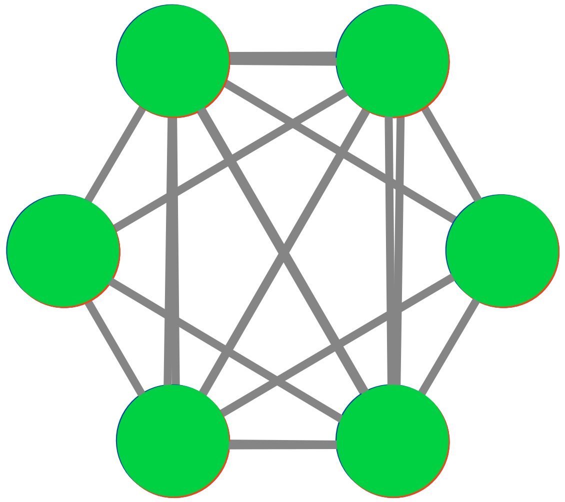
- For given key, pick random order of machines
- Map order onto tree / star via BFS ordering



Random Caching Trees

- Cache / synchronize an object
- Uneven load distribution
- Must not generate hotspot

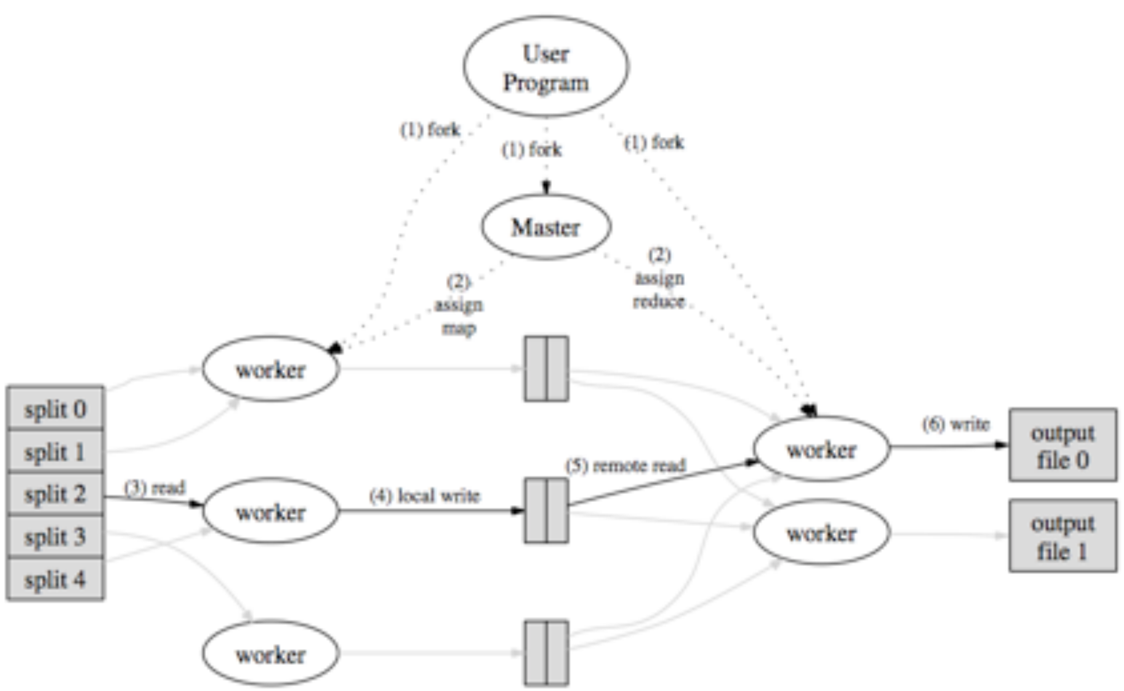
- For given key, pick random order of machines
- Map order onto tree / star via BFS ordering

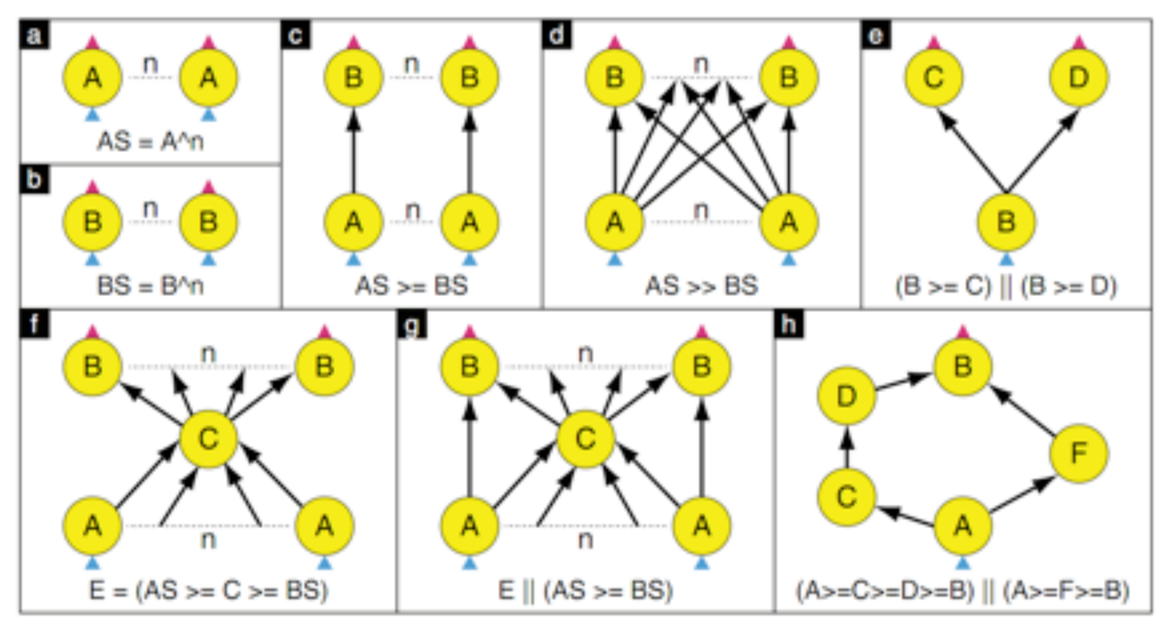


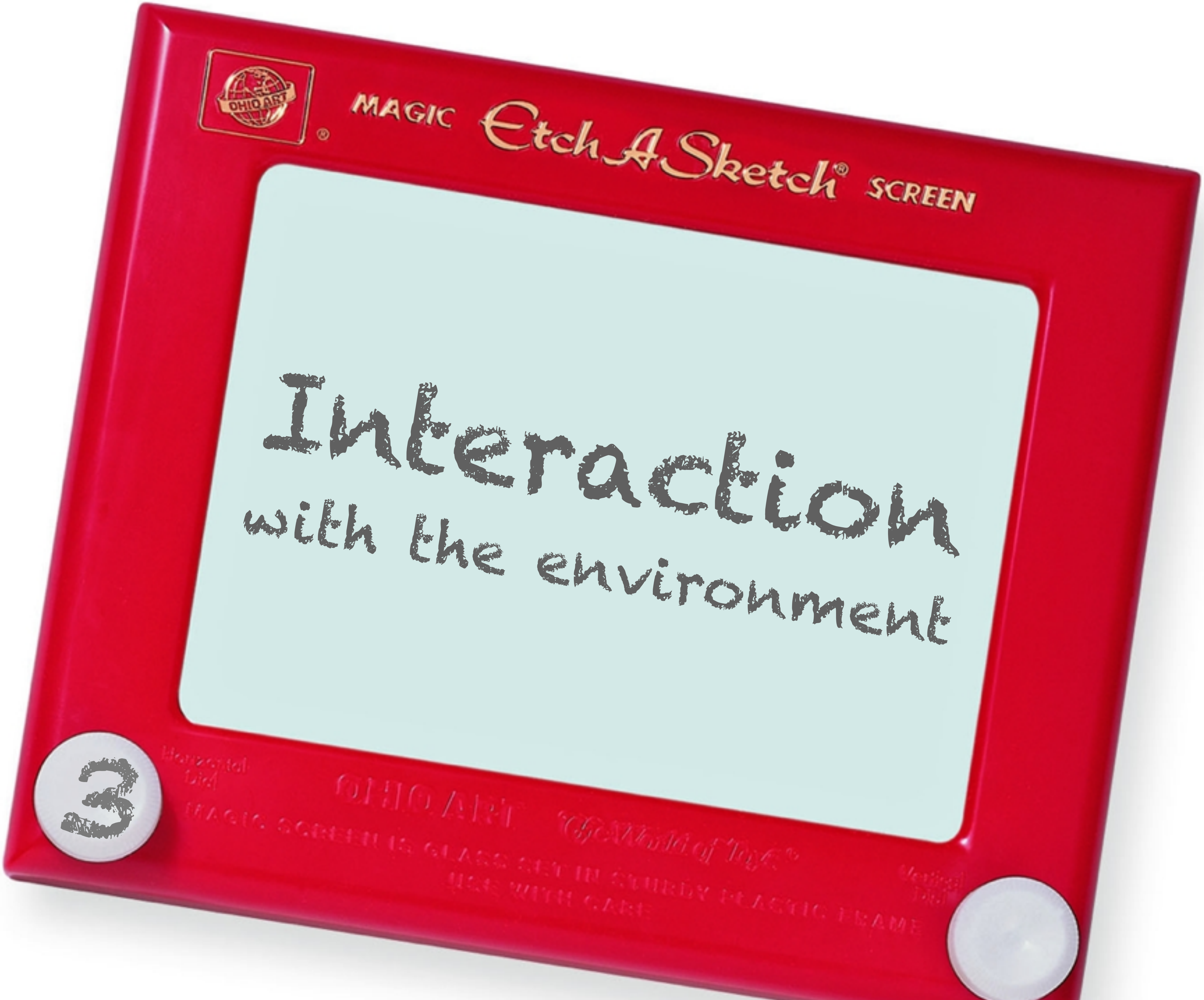
e.g. memcached

More stuff

- Map reduce (e.g. Hadoop)
- Online streaming (e.g. S4, Dryad, Storm)
- NoSQL Database (e.g. pnuts, bigtable)
- Fault tolerant (key,value) storage (e.g. dynamo)
- Smart file system layout (e.g. ceph, GFS2)






MAGIC Etch A Sketch SCREEN
OHIO ART
Interaction
with the environment
3

Batch

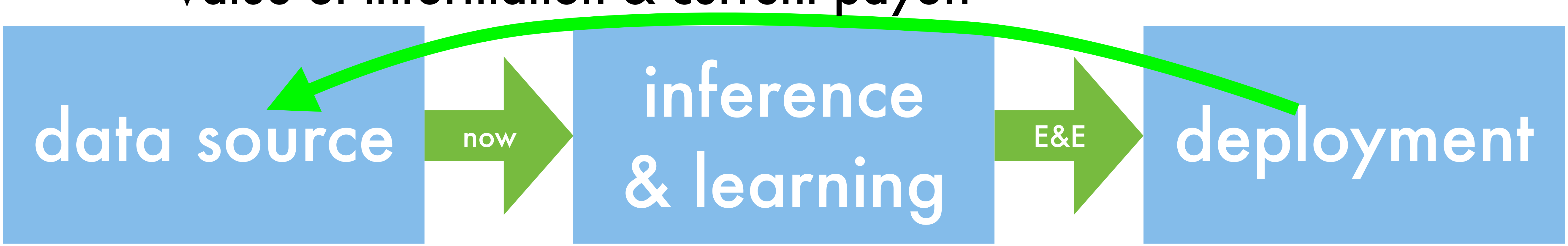
- Data generated independently
    - Editors label data
    - Recorded log files
- Learning algorithm
    - Often invoked from scratch
    - No influence on data source
- Deployment
    - No direct influence on learning
    - Ignores influence on source

Online

- Data generated independently
  - Editors label data
  - Incoming log files
- Learning algorithm
  - Update happens in (near) realtime
  - Adapts to changing data source (good for spam, attacks, news)
- Deployment
  - No direct influence on learning
  - Ignores influence on source

Interactive / Explore & Exploit

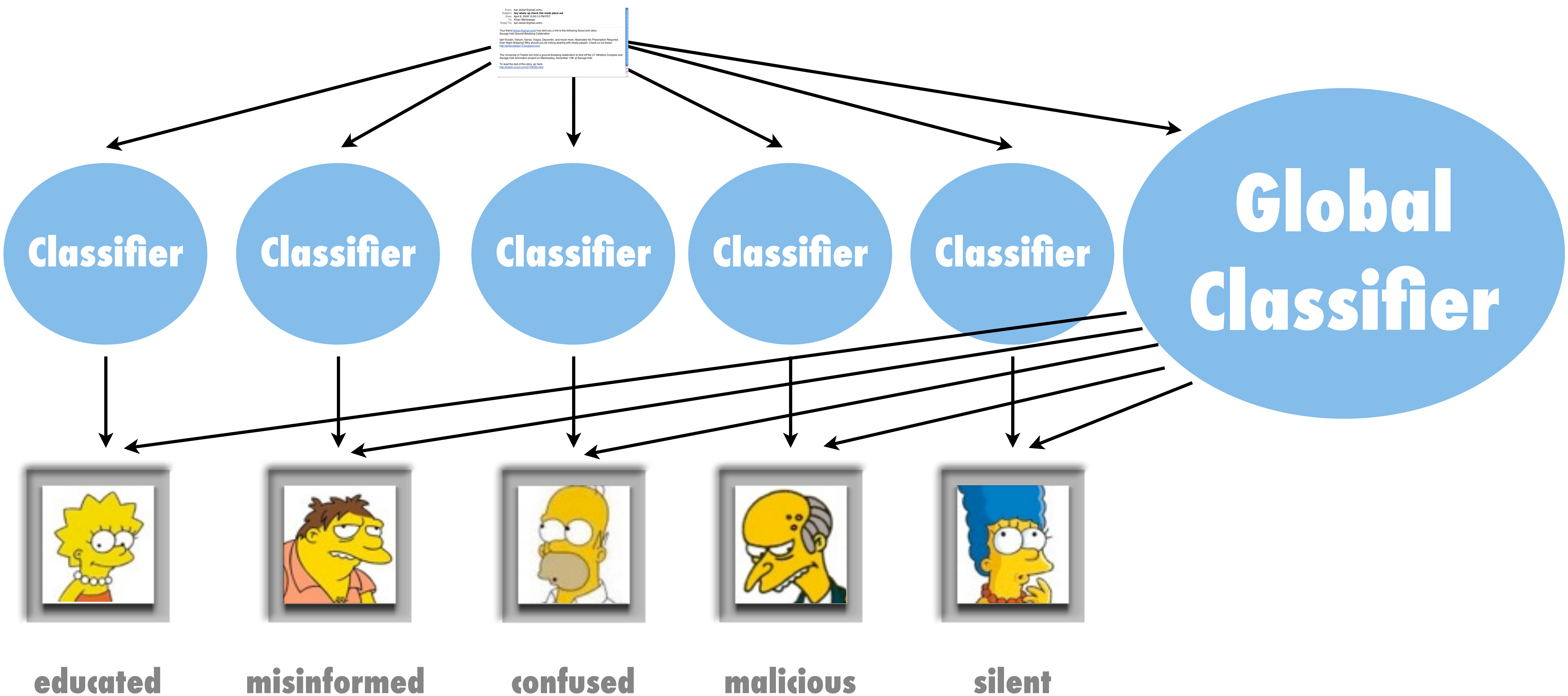
- Data is response to current model
  - Story recommendations
  - Personalized news ranking
- Learning algorithm
  - Update happens in (near) realtime
  - Adapts to changing data source
- Deployment
  - Predictive uncertainty influences exploration
  - Value of information & current payoff



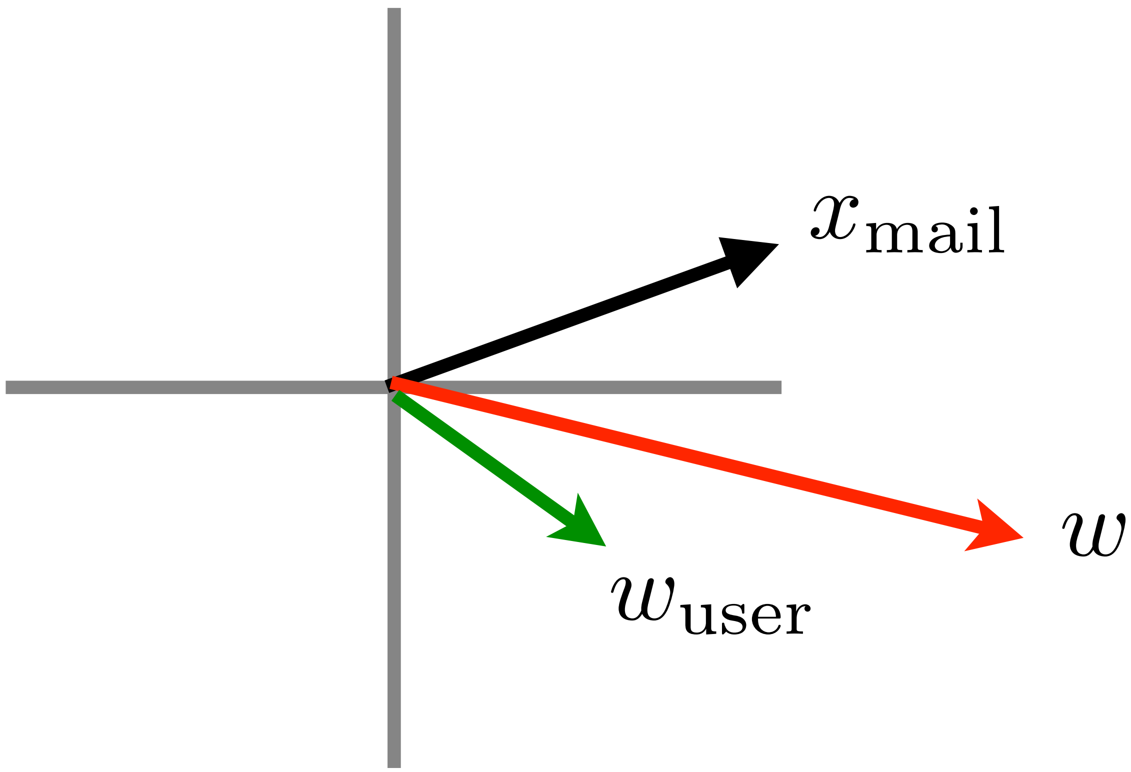

OHIO ART
MAGIC Etch A Sketch SCREEN
• Problems
in machine learning
• Systems
to run the algorithms
• Response
batch/online/interactive
• Compression


MAGIC Etch A Sketch® SCREEN
OHIO ART
Compression
hashing for limited memory
4

Personalized Spam Classification



Personalized Spam Classification



- **Primal representation**

$$f(x,u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle$$

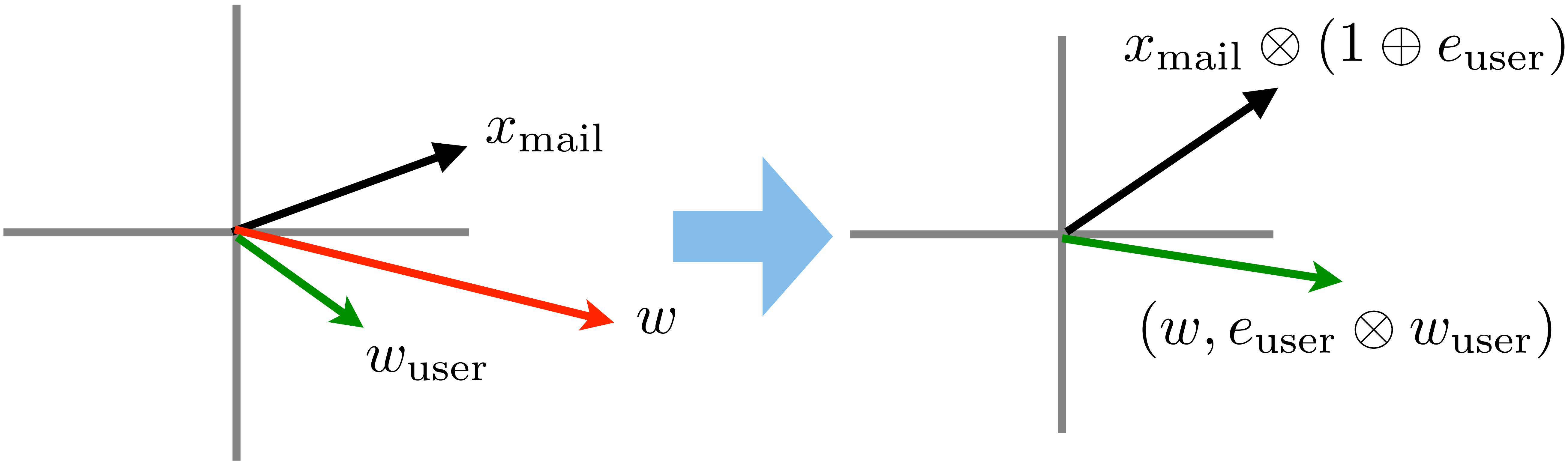
  Kernel representation

$$k((x,u),(x',u')) = k(x,x')[1 + \delta_{u,u'}]$$

  Multitask kernel (e.g. Pontil & Michelli, Daume). Usually does not scale well ...

- **Problem -** dimensionality is 10^6 x 10^8. That is 400TB of space

Personalized Spam Classification



- **Primal representation**

  $$f(x,u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle$$

  Kernel representation

  $$k((x,u),(x',u')) = k(x,x')[1 + \delta_{u,u'}]$$

  Multitask kernel (e.g. Pontil & Michelli, Daume). Usually does not scale well ...
- **Problem -** dimensionality is 10^6 x 10^8. That is 400TB of space

Hash Kernels

Hash Kernels

instance:

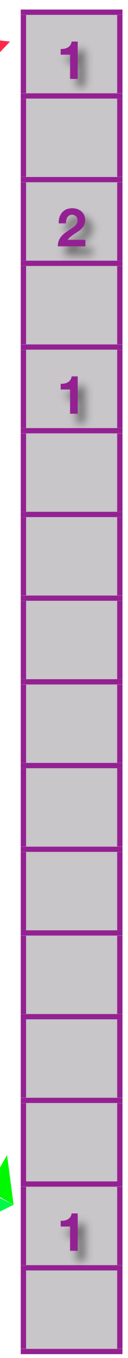
dictionary:

Hey,

please mention subtly during your talk that people should use Yahoo products more often.
Thanks,

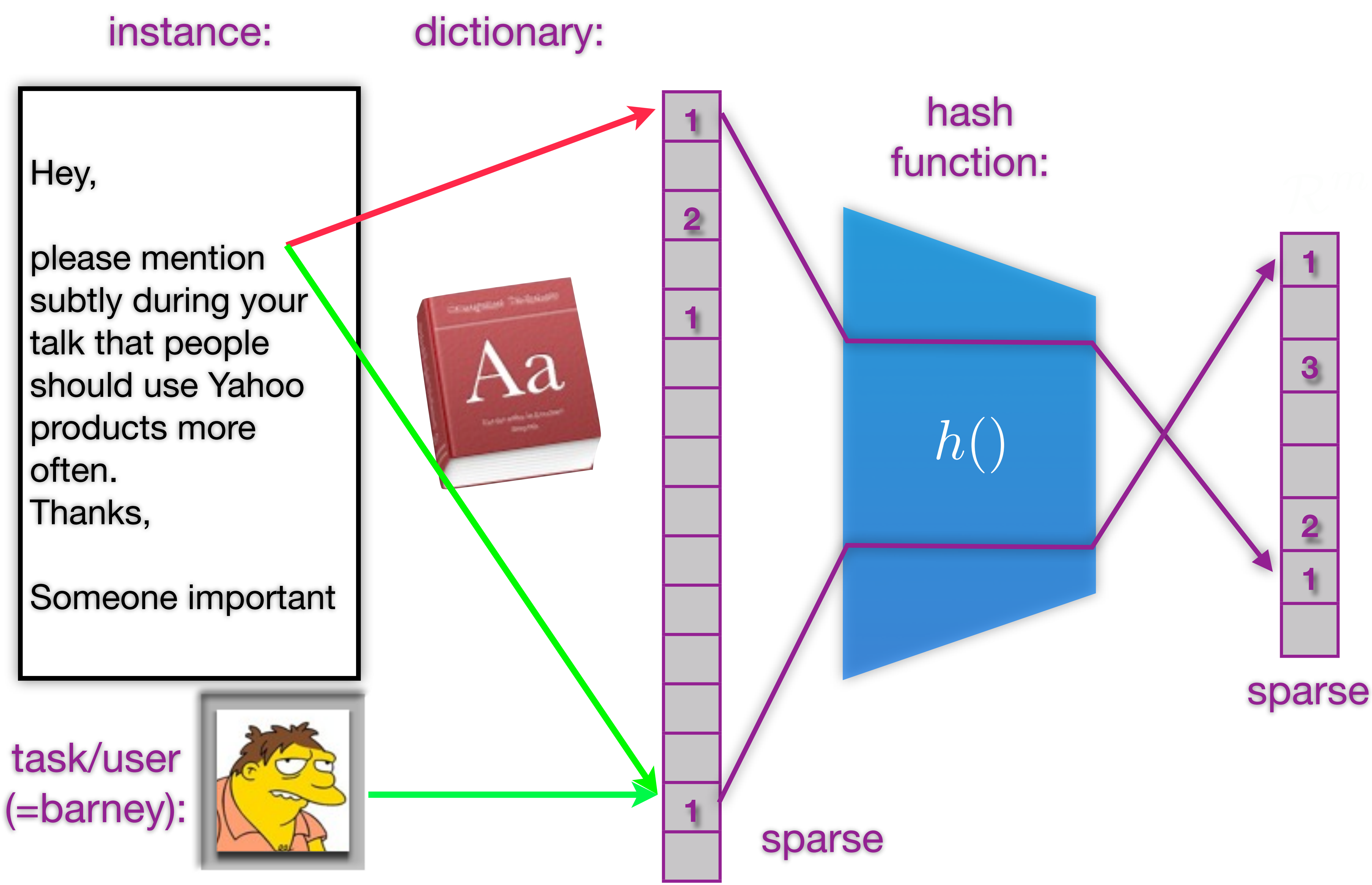
Someone important

task/user (=barney):



sparse

Hash Kernels



Hash Kernels

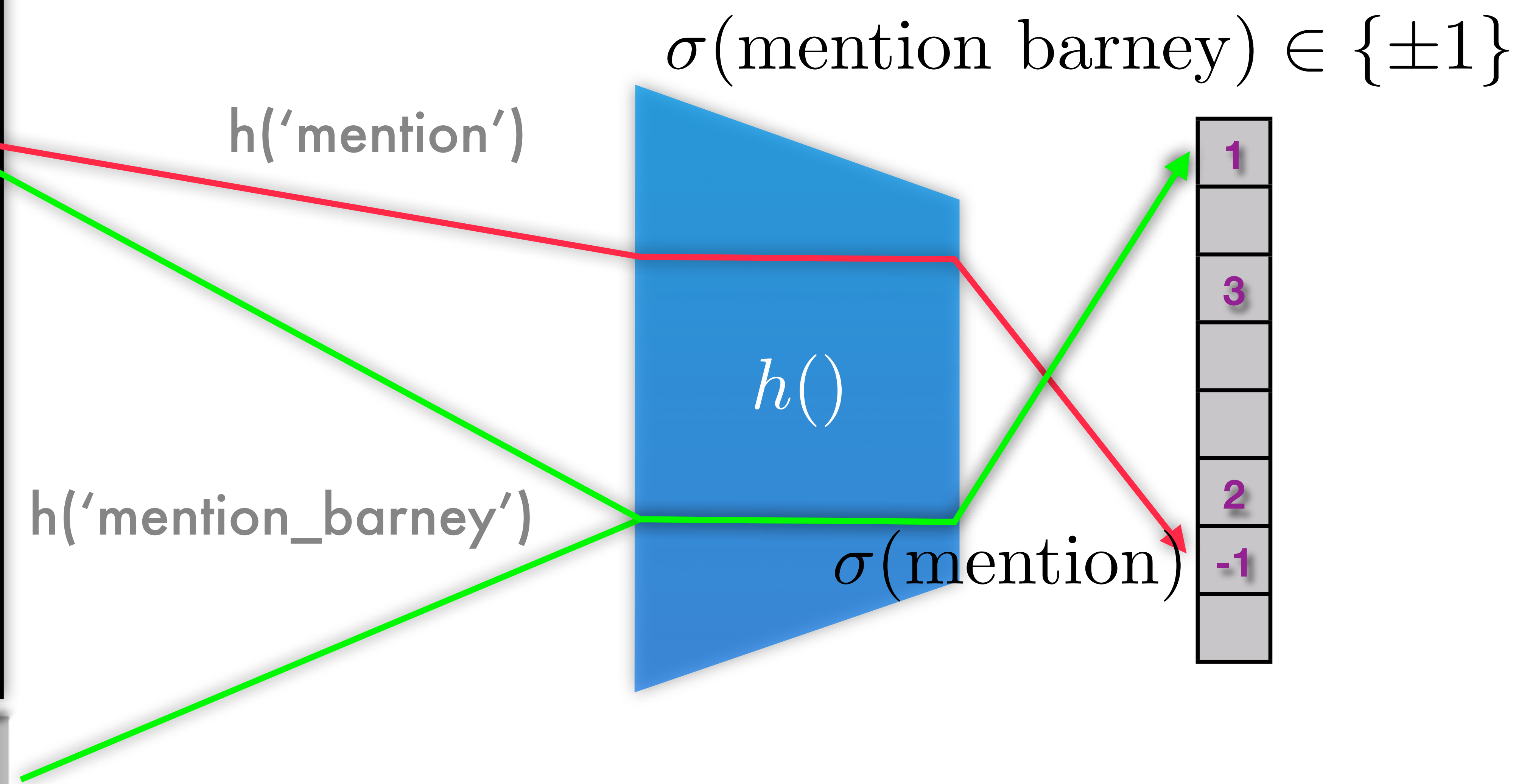
instance:

sparsity preserving, dictionary free

Hey,

please mention subtly during your talk that people should use Yahoo products more often.
Thanks,

Someone important



task/user
= barney

Similar to count sketch
(Charikar, Chen, Farrach-Colton, 2003)

Hash Kernels

- Function evaluation

$$f(x) = \sum_i w_i x_i + b$$

$$f_{\text{hash}}(x) = \sum_i \sigma(i) w[h(i)] x_i + b$$

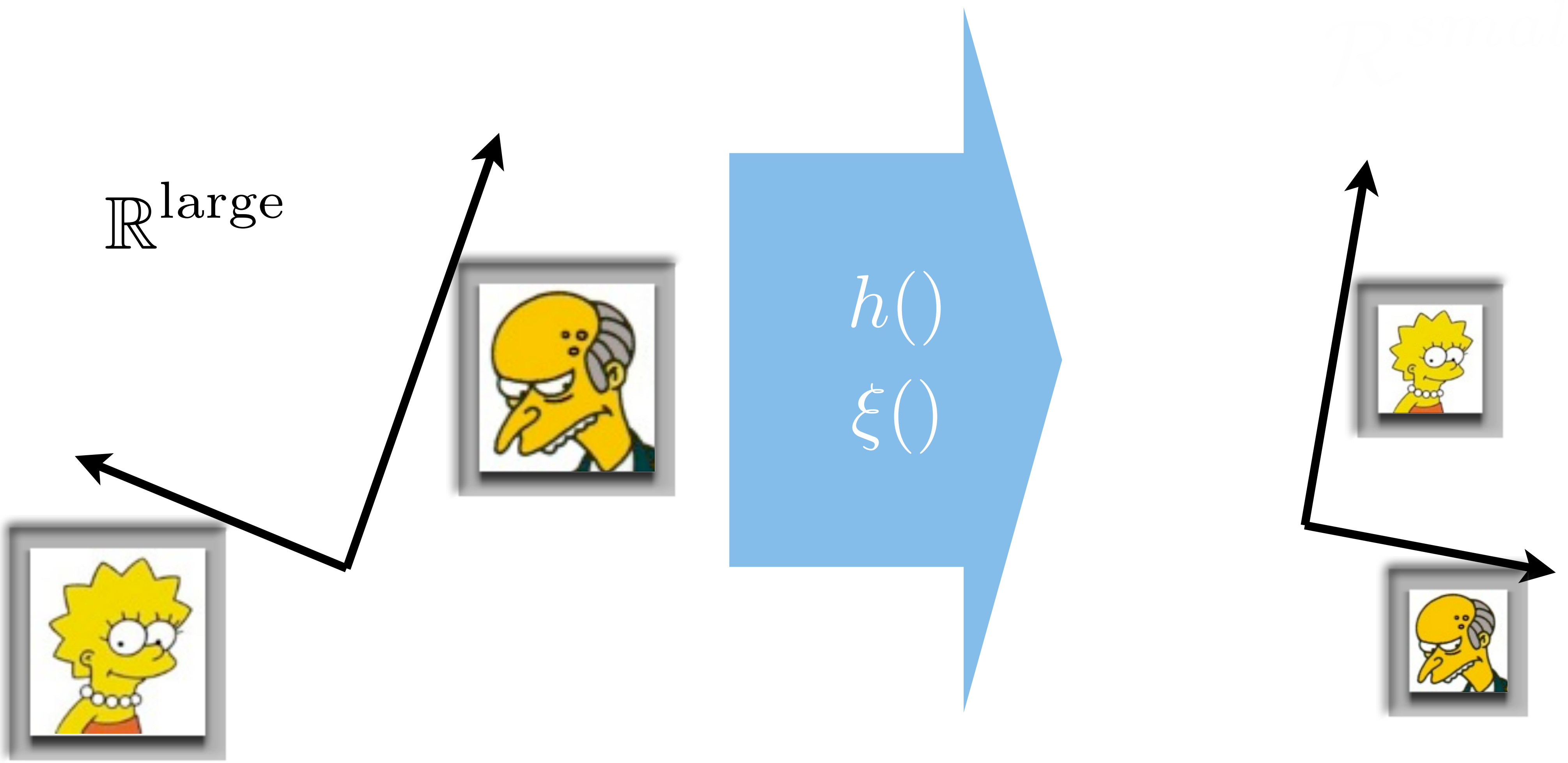
- Kernel

$$k(x, x') = \sum_i x_i x'_i$$

collisions

$$k_{\text{hash}}(x, x') = \sum_{j=1}^{n} \left[\sum_{i:h(i)=j} x_i \sigma(i) \right] \left[\sum_{i:h(i)=j} x'_i \sigma(i) \right]$$

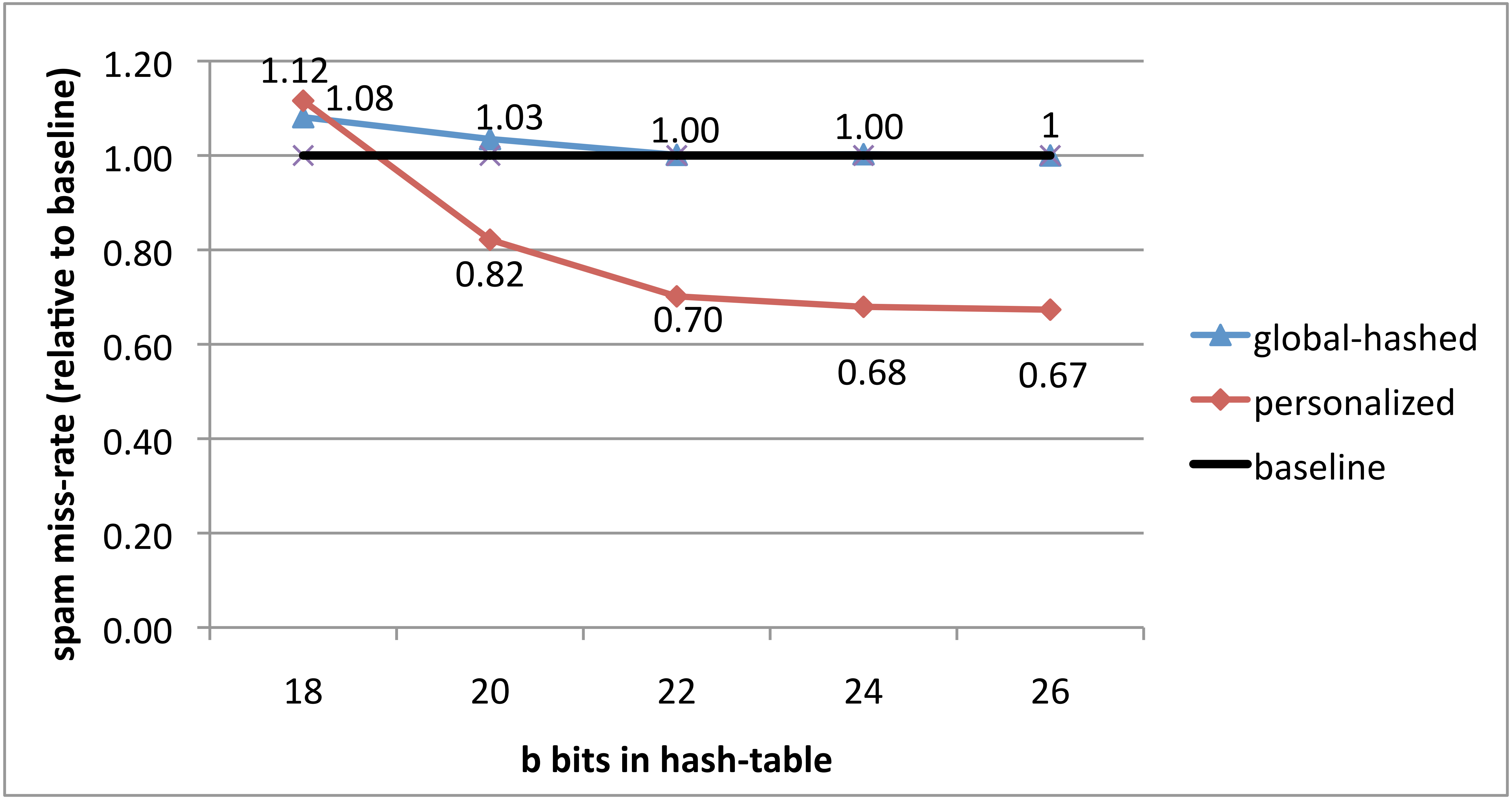
Approximate Orthogonality



We can do multi-task learning!

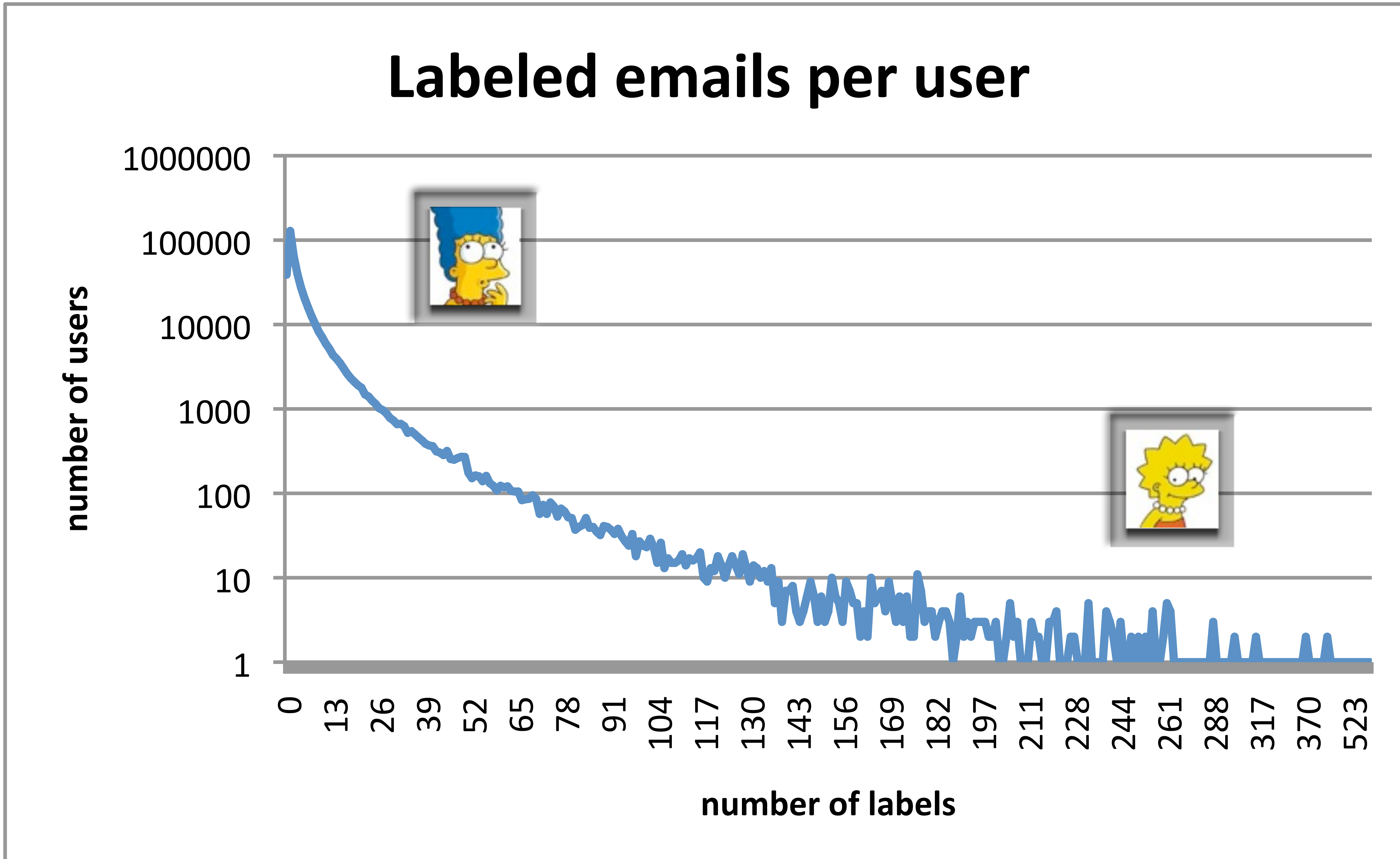
Direct sum in Hilbert Space → **Sum** in Hash Space

Spam classification results



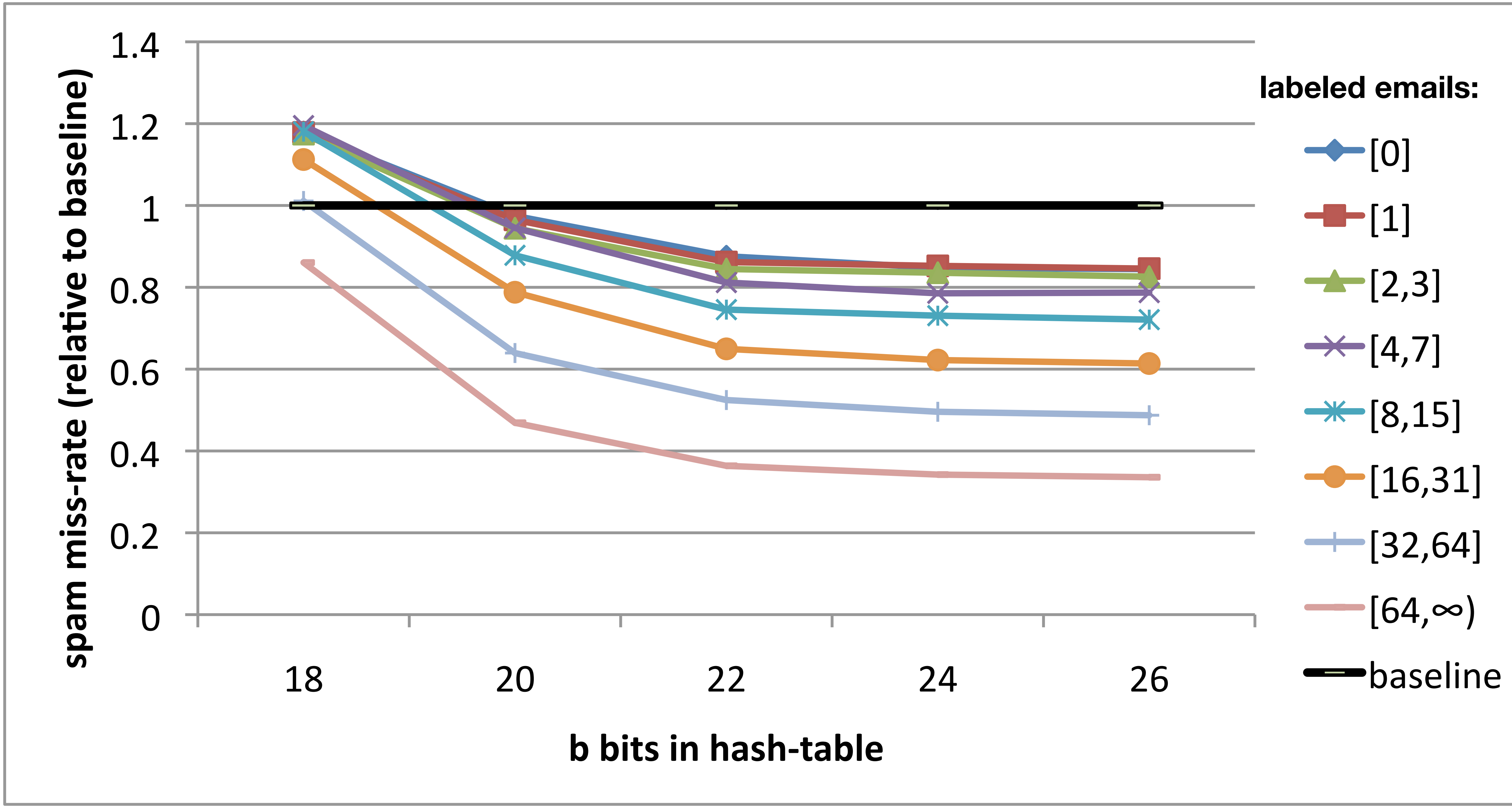
N=20M, U=400K

Lazy users ...

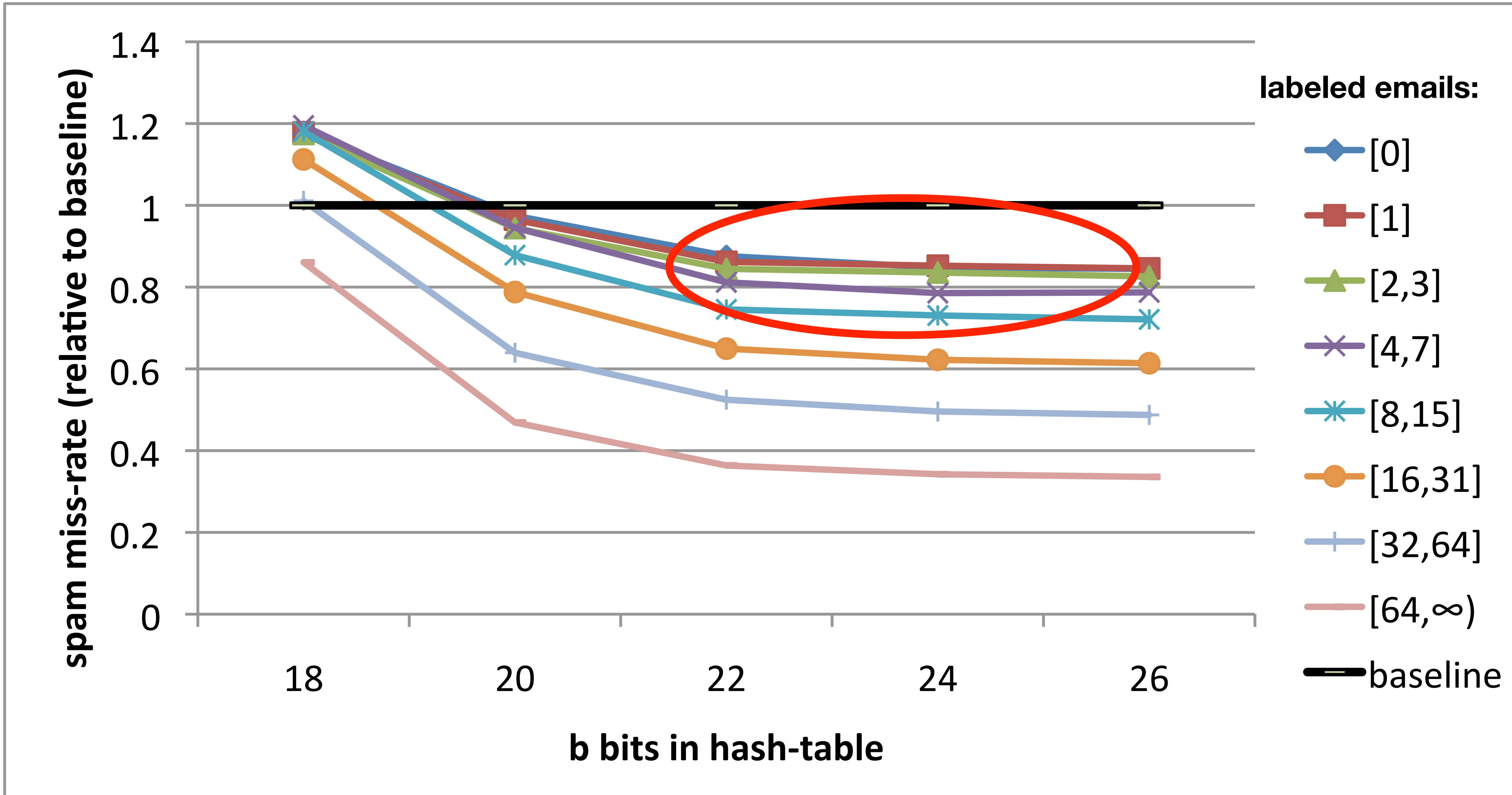

Labeled emails per user
1000000
100000
10000
1000
100
10
1
number of users
0
13
26
39
52
65
78
91
104
117
130
143
156
169
182
197
211
228
244
261
288
317
370
523
number of labels

Results by user group

Results by user group



Results by user group



Even more

- Fast graph comparison
  - Extract subgraph signatures
- Avoiding to implement dynamic data structures
  - Ontologies (hash ontology path labels)
  - Hierarchical factorization (hash context)
  - Content personalization (hash source, user, context)
- Collaborative filtering
  - Compress many users into common parameter vector
- String comparison (kernels)
  - Generate sequence with mismatches, hash and weight
    e.g. dog becomes {(dog,1), (*og, 0.5), (d*g, 0.5), (do*, 0.5)}
- Replace w[complicated key] by w[h(complicated key)]