

# 10-605 ML from Large Datasets

William Cohen

Section A or B

# ADMINISTRIVIA

# Who/Where/When

- Office Hours:
  - Still TBA for all of us, check the wiki
- Course assistant: Dorothy Holland-Minkley (dfh@cs)
- Wiki: [google://" data-bbox="110 440 450 480">cohen CMU" → teaching →](http://google://)
  - [http://curtis.ml.cmu.edu/w/courses/index.php/Machine Learning with Large Datasets 10-605 in Fall 2017](http://curtis.ml.cmu.edu/w/courses/index.php/Machine_Learning_with_Large_Datasets_10-605_in_Fall_2017)
  - this should point to everything else
- Instructor:
  - William W. Cohen
- TAs:
  - Intros coming up....

# Who/Where/When

- We have two sections!
  - One meets here in this room
  - One is on-line
    - The lectures will be recorded and will be up for everyone
  - I predict there will be space for everyone to attend in person by October
    - Maybe even take the exams in one room

# Who/Where/When

- One pending issue with on-line sections:
  - Some students can only sign up for one on-line class.
  - There's some confusion about if section B is on-line or not.
  - Who's having a problem with this?

# Who/Where/When

- Most days there will be an on-line quiz you should do after lecture
- Typically it closes noon the next day
- We have one today
- They don't count for a lot but there are no makeups
  - Quizzes reinforce the lecture
  - And make sure you keep up

# Who/Where/When

- William W. Cohen
  - 1990-mid 1990's: AT&T Bell Labs (working on ILP and scalable propositional rule-learning algorithms)
  - Mid 1990's—2000: AT&T Research (text classification, data integration, knowledge-as-text, information extraction)
  - 2000-2002: Whizbang! Labs (info extraction from Web)
  - 2002-2008, 2010-now: CMU (IE, biotext, social networks, learning in graphs, info extraction from Web, scalable first-order learning)
  - 2008-2009, Jan-Jul 2017: Visiting Scientist at Google

# TAs

# Anant Subramanian – MS in CS

I am a second year Master's student in the Computer Science Department.

I'm interested in developing distributed and parallel systems that cater to machine learning algorithms and workloads.

I hope you enjoy the course as much as I did, and learn a lot in the process :)



# Bo Chen

- I am a second year master student in MIIS, LTI.
- I am interested in machine learning and reinforcement learning. 10605 helps me understand scaling machine learning algorithms, really like it.



# Chen Hu – MIIS in LTI

I am a second year master student from MIIS program in LTI, School of Computer Science.

I am interested in scaling and parallelizing machine learning algorithms in large system. I took 10605 last Fall and got so much interesting ideas here.



# Minxing Liu- MCDS

I'm a second year Master student in MCDS program, under LTI, SCS.

I'm interested in building scalable machine learning algorithms and systems, which is quite popular now! This summer I interned at Microsoft Bing & Cortana.

I took 10605 last year, and learned a lot of interesting topics on several sub-fields of Machine Learning. I'm sure you'll have fun in this course and find it quite useful.



# Ning Dong - MIIS

I am second year master student.  
I am interested in scalable  
machine learning and intelligent  
systems.

I interned in Google this summer  
working in Youtube Thumbnails.

I took 10605 last year and it was  
excellent experience. Glad to be  
TA of this course and start the  
journey again!



# Sarguna Janani Padmanabhan (MCDS)

I am a second year student  
pursuing Masters in  
Computational Data Science  
in LTI.

I took 10-605 in Fall'16 and  
absolutely enjoyed the  
course, heavily using the  
concepts learnt in the course  
at my summer internship  
with Facebook.



# Tao Lin

- 2nd year MCDS student in LTI
- Enjoyed 605 last year
- Interned at Google to visualize large-scale data
- Contributed to Spark



# Yifan Yang- MCDS in LTI

I'm a second year Masters student in LTI, School of Computer Science.

I'm working on the Stanford Question Answer Dataset (SQuAD) as my capstone project. The goal is to improve deep learning model to have a better understanding on text.



# Yuhan Mao – MCDS in LTI

I'm a second year Masters student in MCDS, LTI, School of Computer Science.

I have an obsession to large data and enjoy implementing ML algorithms on it to gain the knowledge beneath it.

This summer I interned at Amazon. Using my knowledge in large data, I cut the computational cost for their DNN recommenders about half a million dollars per year. Many topics I learnt in this course like DNN and Spark are much helpful.





## Rose Catherine Kanjirathinkal

I'm a 4<sup>th</sup> year Ph.D. student in LTI. My research focuses on improving Recommender Systems using Deep Learning as well as Knowledge Graphs. I am interested in different techniques that explain recommendations to the user.

Before joining CMU, I worked at IBM Research – India, Bangalore, from 2009 to 2014 (Question Answering, Résumé mining, Big Data).

And before that, I completed my Masters at Indian Institute of Technology Bombay (IIT-B).



# What/How

I kind of like language tasks, especially for this task:

- The data (usually) makes sense
- The models (usually) make sense
- The models (usually) are complex, so
  - More data actually helps
  - Learning simple models *vs* complex ones is sometimes computationally different

# What/How

- Programming Languages and Systems:
  - Python
  - Hadoop and Spark
- Resources:
  - Your desktop/laptop (to start)
    - You'll need approximately 0.5Tb space.
  - Stoat hadoop cluster:
    - 104 worker nodes, with 8 cores, 16 GB RAM, 4 1TB.
    - 30 worker nodes, with 8 cores, 16 GB RAM, 250Gb+
  - Amazon Elastic Cloud
    - Amazon EC2 [<http://aws.amazon.com/ec2/>]
    - Allocation: \$50 worth of time per student

# What/How: 10-605

- **Lecture** course for MS and advanced undergrads
- 60% assignments (6/7)
  - Mostly biweekly programming assignments
    - Not a lot of lines of code, but it will take you time to get them right
  - You can drop one but:
    - HW1a HW1b count as one homework not two
    - HW 1ab and HW2 are related
    - HW4 and HW5 are related
- 15% midterm
- 20% final
- 5% class participation & quizzes

# What/How: ~~10-605~~ 805

- Project course for PhDs and MS students
  - MS students, send me your cv for *permission* to enroll in 10-805 by Friday 9/1
  - Alternative: sign up for 10-605 for now; there's often space in 10-805 later on, and I can transfer you.
  - Don't sign up for 10-805 because it's easier to get in!
- 40% assignments (4/7)
- 15% midterm
- 40% project (no final)
  - open-ended, self-directed project
  - final deliverable is a conference-length paper
- 5% class participation & quizzes

# What/How: 10-605 students working on a project with an 10-805 student

- Lecture course for MS and advance undergrads
- 50% assignments (5/7)
- 15% midterm
- 30% project (no final)
- 5% class participation & quizzes
- There will be some match-making we do but it's up to the 805 students to decide how much help they want/need.

# What/How: 10-805 and 11-805

- Projects
  - 2012: everyone did projects and 5 HWs
    - Guest lectures from industry, experts
  - 2013: projects were optional
    - not everyone wanted to do projects in
    - I wanted to give more programming depth
  - 2014: nobody did projects
    - I had too many students
  - Spring/fall 2015, fall 2016
    - 805/605 split

# What/How: 601 co-req

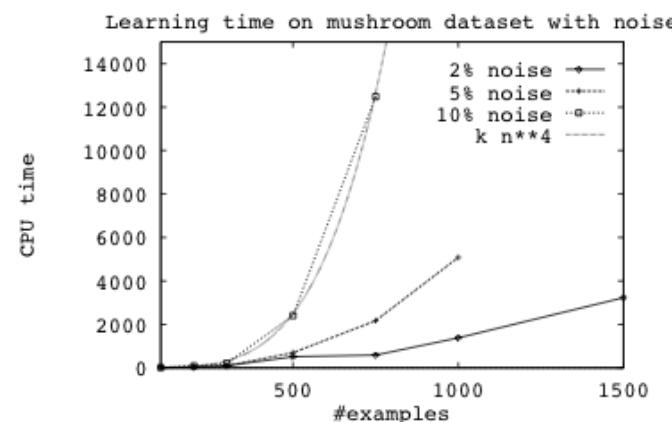
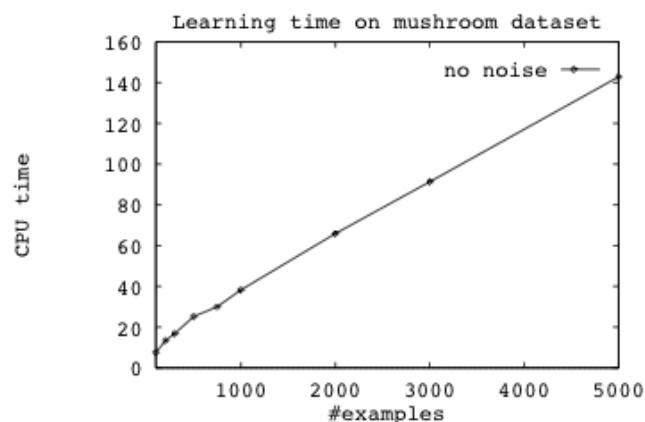
- You should have as a prereq or co-req one of the MLD's intro ML courses: 10-401, 10-601, 10-701, 10-715
- Lectures are designed to *complement* that material
  - computational aspects vs informational aspects
- This is a hard requirement
- Sorry exceptions don't scale well ☹
  - We're gonna check co-reqs
  - I can't remove you except by failing you

# What/How: cheating vs working together

- I have a long and explicit policy
  - stolen from Roni Rosenfeld - read the web page
  - tl;dr: transmit information like they did in the stone age, brain-to-brain, and document it
  - do not copy anything digitally
  - exceptions (eg projects) will be explicitly stated
- Autolab makes it easy to run duplication-detectors on code
- If you are caught:
  - *everybody involved will fail the course* by default
  - *every* infraction *always* gets reported up to the Office of Academic Integrity, the head of your program, the dean of your school, ....
  - a second offense is very bad

# **BIG DATA HISTORY: FROM THE DAWN OF TIME TO THE PRESENT**

# Big ML c. 1993 (Cohen, “Efficient...Rule Learning”, IJCAI 1993)



```
$ ripper ..\tdata/talks
```

Final hypothesis is:

```
talk_announcement :- WORDS ~ talk, WORDS ~ Subject_talk (54/1).
talk_announcement :- WORDS ~ '2d416' (26/3).
talk_announcement :- WORDS ~ system, WORDS ~ 'To_1126@research' (4/0).
talk_announcement :- WORDS ~ mh, WORDS ~ time (5/1).
talk_announcement :- WORDS ~ talk, WORDS ~ used (3/0).
talk_announcement :- WORDS ~ presentations (2/1).
default non_talk_announcement (390/1).
```

Benchmark	CPU Time					
	No Pruning	REP	Grow		MDLGrow	
kr-vs-knn	10.8 ± 0.6	18.5 ± 1.8	13.2	± 0.6	13.4	± 0.6
bridge-t/d	12.7 0.9	27.6 3.3	10.4	0.9	8.1	0.7
thyroid-hypo	72.6 6.5	56.6 9.4	46.4	6.2	48.1	6.3
bridge-mtr1	22.1 0.6	76.7 9.6	16.5	1.5	10.6	0.6
mushroom	35.6 0.8	78.3 7.8	44.5	1.4	45.3	1.7
thyroid-allbp	144.8 7.7	164.5 12.6	99.5	4.6	100.7	5.8
bridge-span	29.5 0.8	176.2 18.6	31.9	2.3	13.3	0.8
bridge-rel-l	44.1 1.0	294.1 36.9	34.0	2.9	14.1	1.2
bridge-type	38.9 1.1	370.6 25.2	40.5	2.3	21.2	1.0
sonar	561.0 12.9	399.2 15.1	368.2	12.0	370.6	12.1
segment	815.7 23.9	1264.0 86.6	728.2	29.6	733.6	27.8
mushroom*	217.2 10.1	4081.7 485.4	276.7	23.4	135.1	6.9
kr-vs-knn*	154.2 11.9	5549.3 1255.3	206.6	23.5	53.5	4.0
rds	3189.1 84.9	15155.2 1282.4	2210.0	52.0	879.9	42.4
Average for Benchmark Set 2	382.03	<b>2695.63</b>	<b>402.00</b>		<b>239.38</b>	
Average for Benchmark Set 1	108.4	384.0	105.9		100.5	

Table 3: Comparing runtimes

# More on this paper

## *Algorithm*

- Phase 1: build rules
  - Discrete greedy search:
  - Starting with empty rule set, add conditions greedily
- Phase 2: prune rules
  - starting with phase 1 output, remove conditions

```
talk_announcement :-
```

```
    .
```

```
    .
```

```
    .
```

```
    .
```

```
    .
```

```
    .
```

```
    .
```

```
    .
```

```
    .
```

```
    .
```

```
default non_talk_announcement .
```

# More on this paper

## *Algorithm*

- Phase 1: build rules
  - Discrete greedy search:
  - Starting with empty rule set, add conditions greedily
- Phase 2: prune rules
  - starting with phase 1 output, remove conditions, greedily

```
talk_announcement :- WORDS ~ talk, WORDS ~ Subject_talk, WORDS ~ p_comma (54/0).  
talk_announcement :- WORDS ~ '2d416', WORDS ~ be (19/0).  
talk_announcement :- WORDS ~ show, WORDS ~ talk (7/0).  
talk_announcement :- WORDS ~ mh, WORDS ~ time, WORDS ~ research (4/0).  
talk_announcement :- WORDS ~ system, WORDS ~ 'To_1126@research' (3/0).  
talk_announcement :- WORDS ~ '2d416', WORDS ~ memory (3/0).  
talk_announcement :- WORDS ~ interfaces, WORDS ~ From_p_exclam_point (2/0).  
talk_announcement :- WORDS ~ presentations, WORDS ~ From_att (2/0).  
default non_talk_announcement .
```

# More on this paper

## *Algorithm*

- Phase 1: build rules
  - Discrete greedy search:
  - Starting with empty rule set, add conditions greedily
- Phase 2: prune rules
  - starting with phase 1 output, remove conditions, greedily

```
talk_announcement :- WORDS ~ talk, WORDS ~ Subject_talk (54/1).  
talk_announcement :- WORDS ~ '2d416' (26/3).  
talk_announcement :- WORDS ~ system, WORDS ~ 'To_1126@research' (4/0).  
talk_announcement :- WORDS ~ mh, WORDS ~ time (5/1).  
talk_announcement :- WORDS ~ talk, WORDS ~ used (3/0).  
talk_announcement :- WORDS ~ presentations (2/1).  
default non_talk_announcement (390/1).
```

# More on this paper

## *Algorithm*

- Fit the POS,NEG example
- While POS isn't empty:
  - Let  $R$  be "if True  $\rightarrow$  pos"
  - While NEG isn't empty:
    - LI** • Pick the "best" [i] condition  $c$  of the form " $x_i=True$ " or " $x_i=false$ "
    - Add  $c$  to the LHS of  $R$
    - Remove examples that don't satisfy  $c$  from NEG
  - Add  $R$  to the rule set [ii]
  - Remove examples that satisfy  $R$  from POS
- Prune the rule set:
  - ...

**cubic!**

[i] "Best" is wrt some statistics on  $c$ 's coverage of POS,NEG

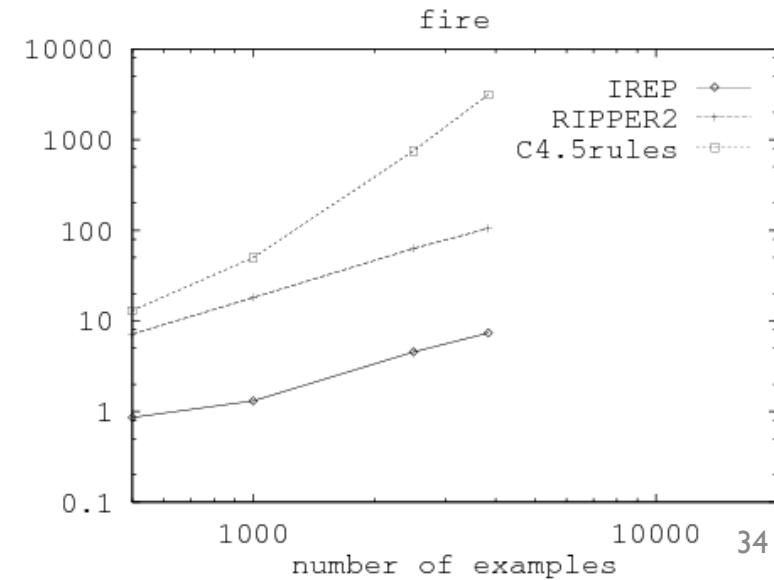
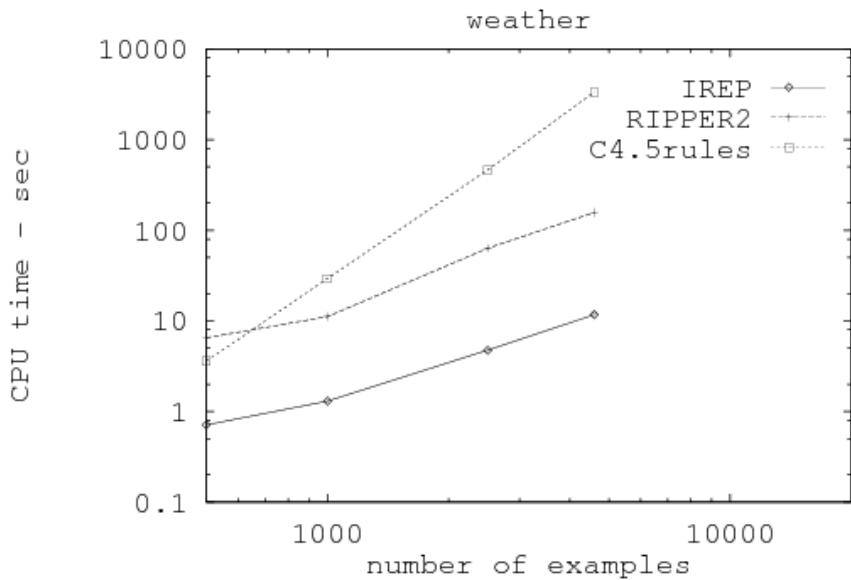
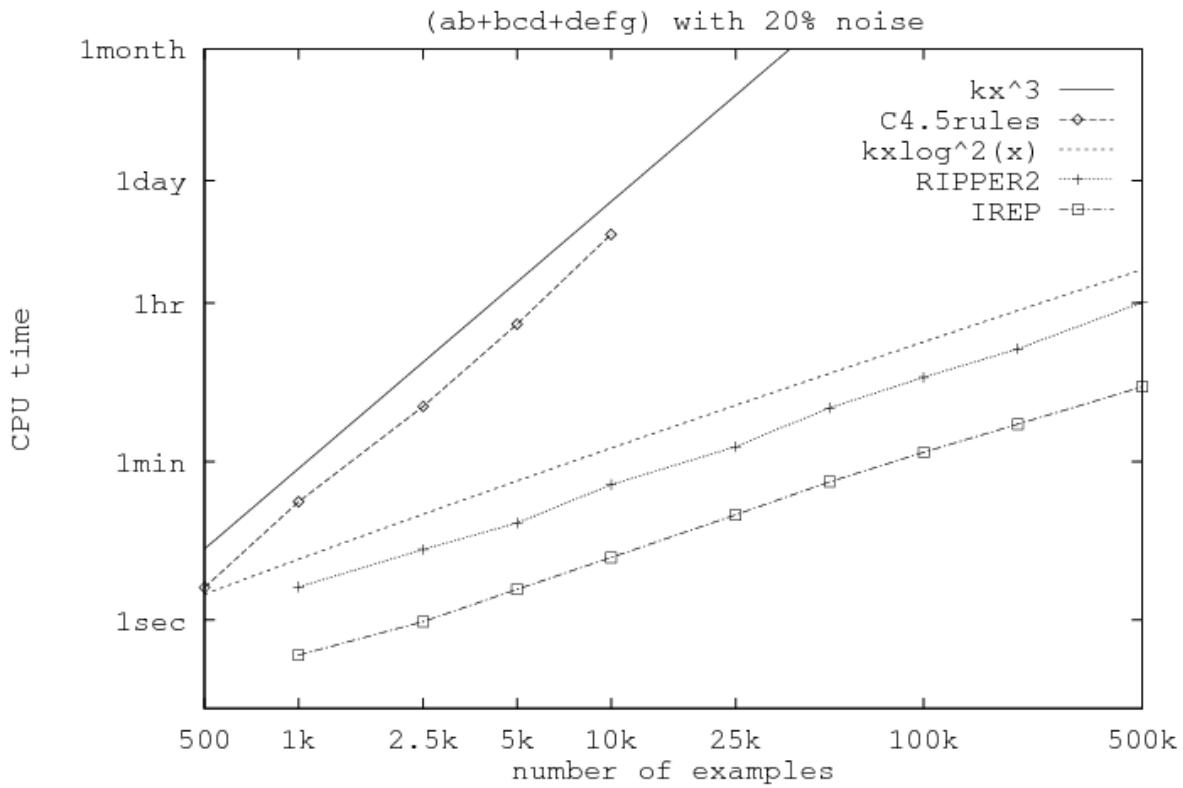
[ii]  $R$  is now of the form "if  $x_1=_$  and  $x_2=_$  and ...  $\rightarrow$  pos"

## *Analysis*

- The total number of iterations of L1 is the number of conditions in the rule set – call it  $m$
- Picking the "best" condition requires looking at all examples
  - say there are  $n$  of these
- Time is at least  $m*n$
- The problem:
  - When there are noisy positive examples the algorithm builds rules that cover just 1-2 of them
  - So with huge noisy datasets you build huge rulesets

**quadratic**

# Related paper from 1995...



# So in mid 1990's.....

- Experimental datasets were small
- Many commonly used algorithms were *asymptotically* “slow”
- Not many people really cared

# Big ML c. 2001 (Banko & Brill, “Scaling to Very Very Large...”, ACL 2001)

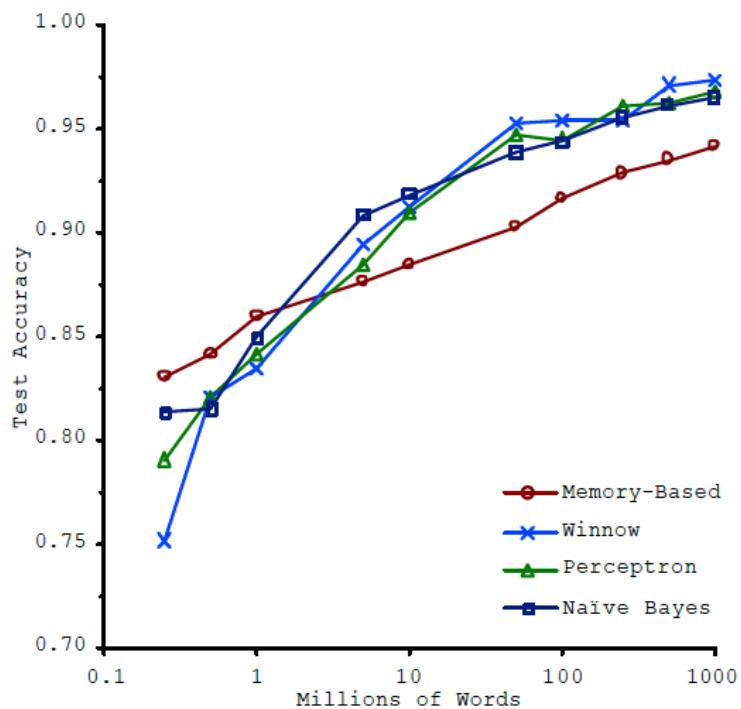


Figure 1. Learning Curves for Confusion Set Disambiguation

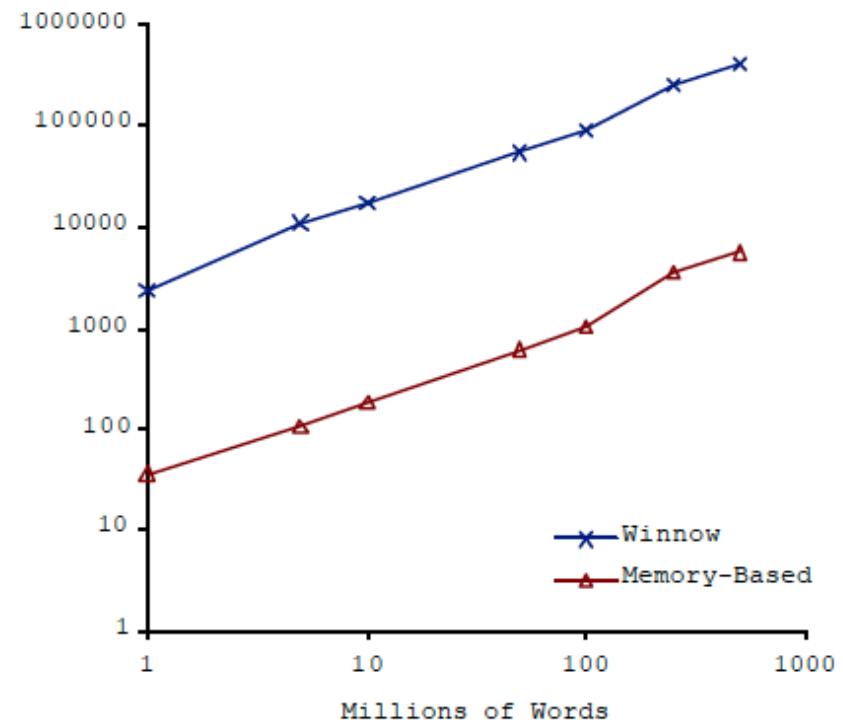
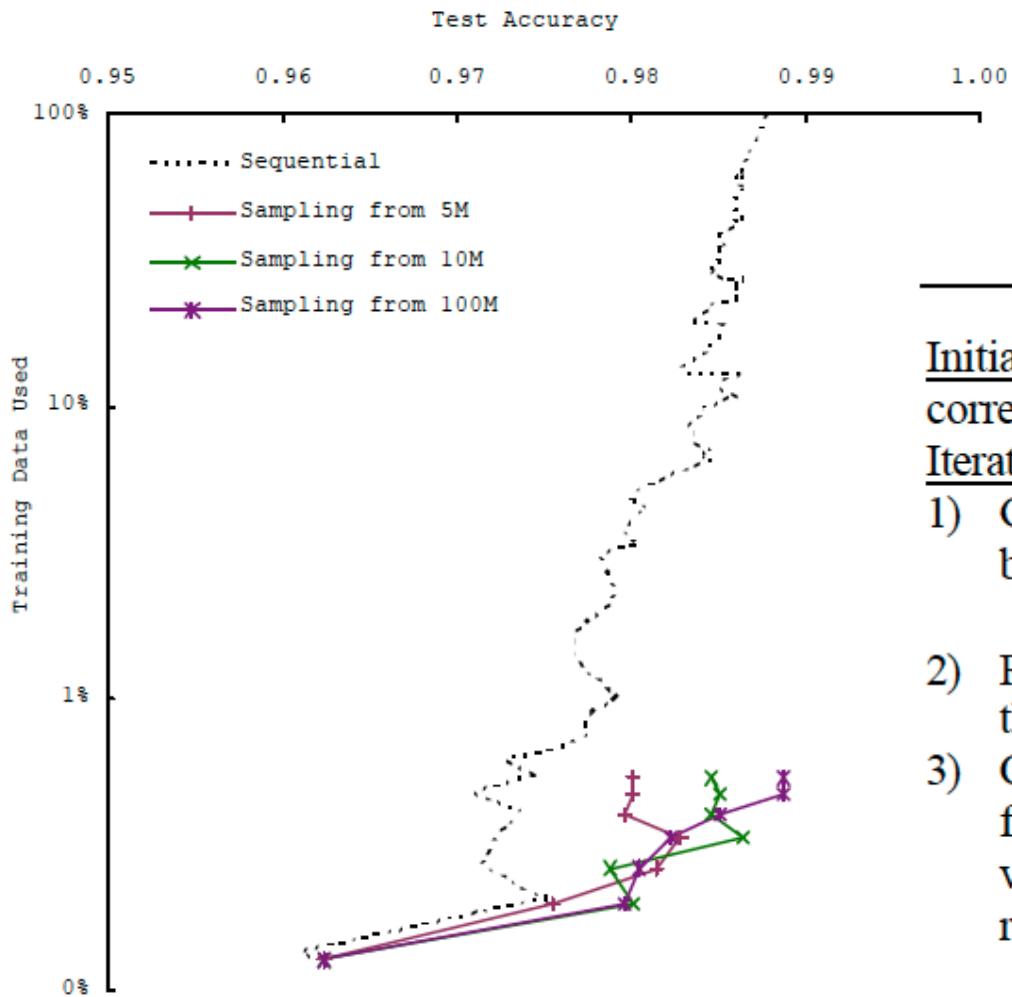


Figure 2. Representation Size vs. Training Corpus Size

Task: distinguish pairs of easily-confused words  
 (“affect” vs “effect”) in context

# Big ML c. 2001 (Banko & Brill, "Scaling to Very Very Large...", ACL 2001)



Initialize: Training data consists of  $X$  words correctly labeled

Iterate:

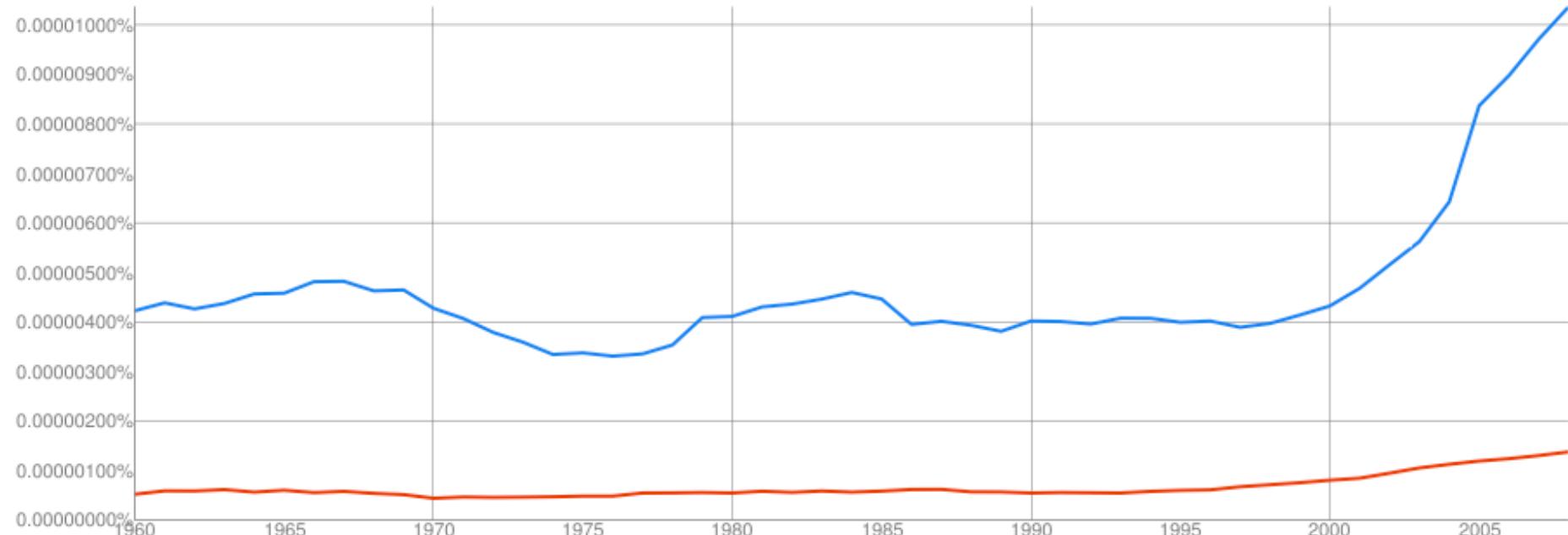
- 1) Generate a committee of classifiers using bagging on the training set
- 2) Run the committee on unlabeled portion of the training set
- 3) Choose  $M$  instances from the unlabeled set for labeling - pick the  $M/2$  with the greatest vote entropy and then pick another  $M/2$  randomly – and add to training set

Figure 4. Active Learning with Large Corpora

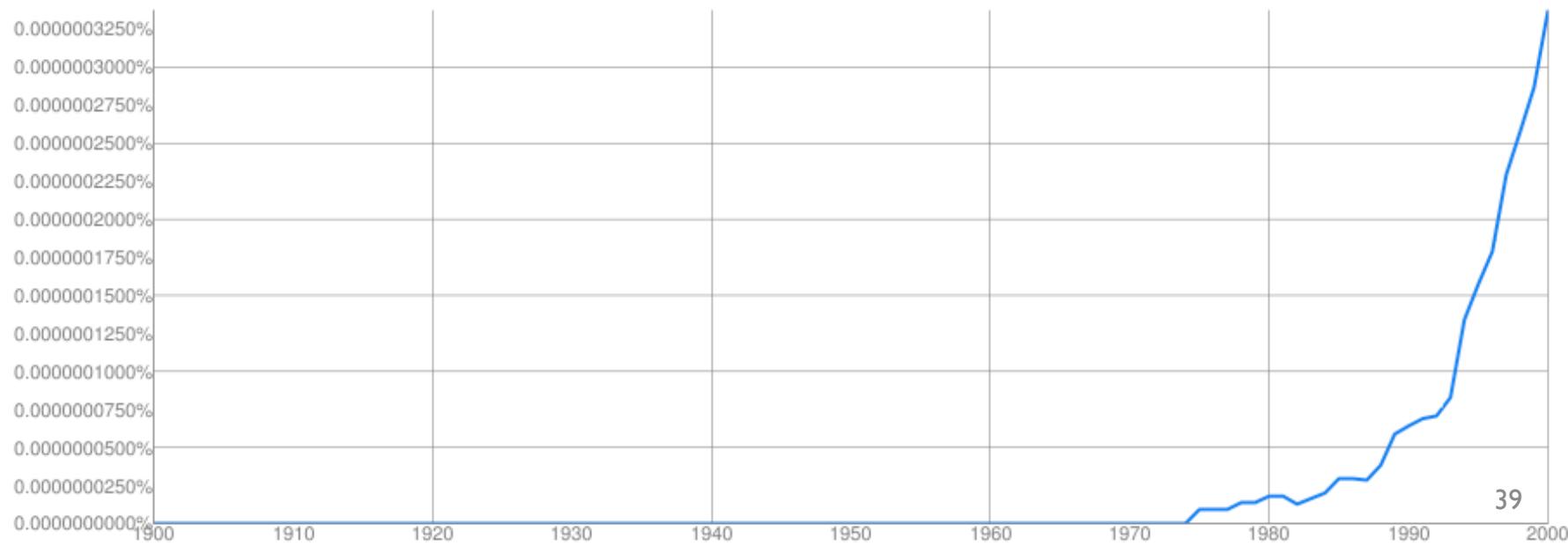
# Why More Data Helps: A Demo

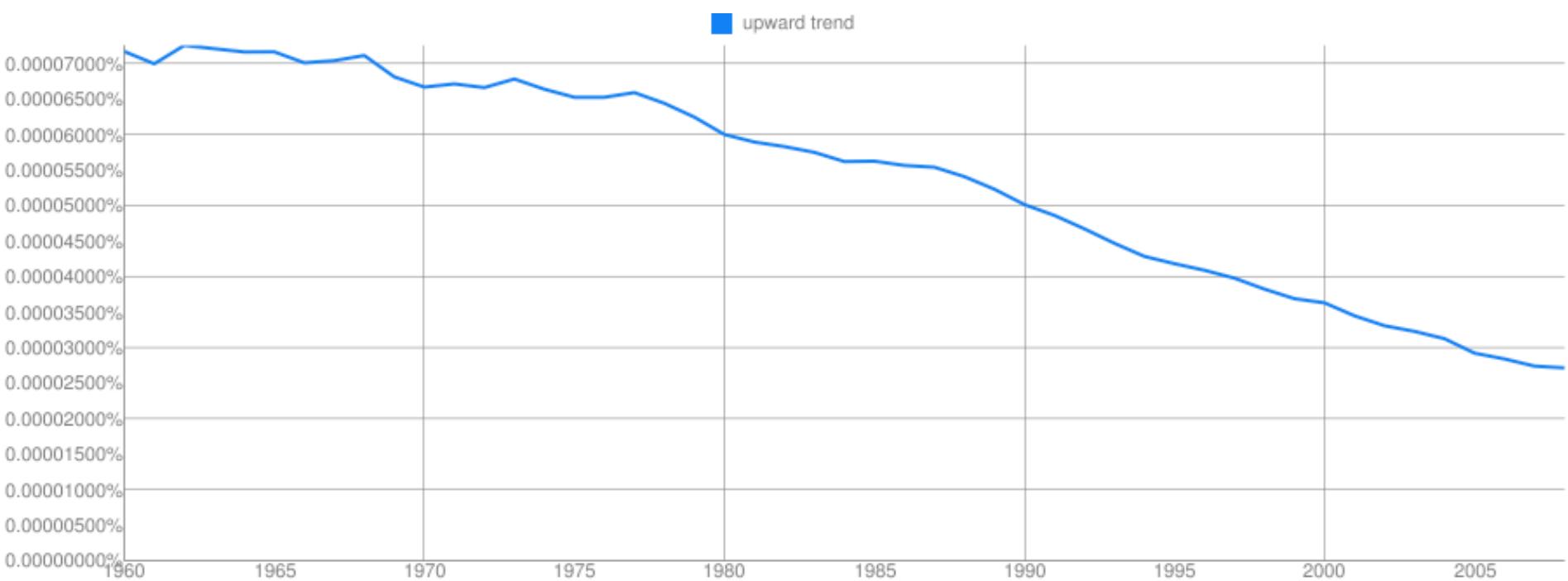
- Data:
  - All 5-grams that appear  $\geq$  40 times in a corpus of 1M English books
    - approx 80B words
    - 5-grams: 30Gb compressed, 250-300Gb uncompressed
    - Each 5-gram contains frequency distribution over *years*

■ merry Christmas ■ happy holidays



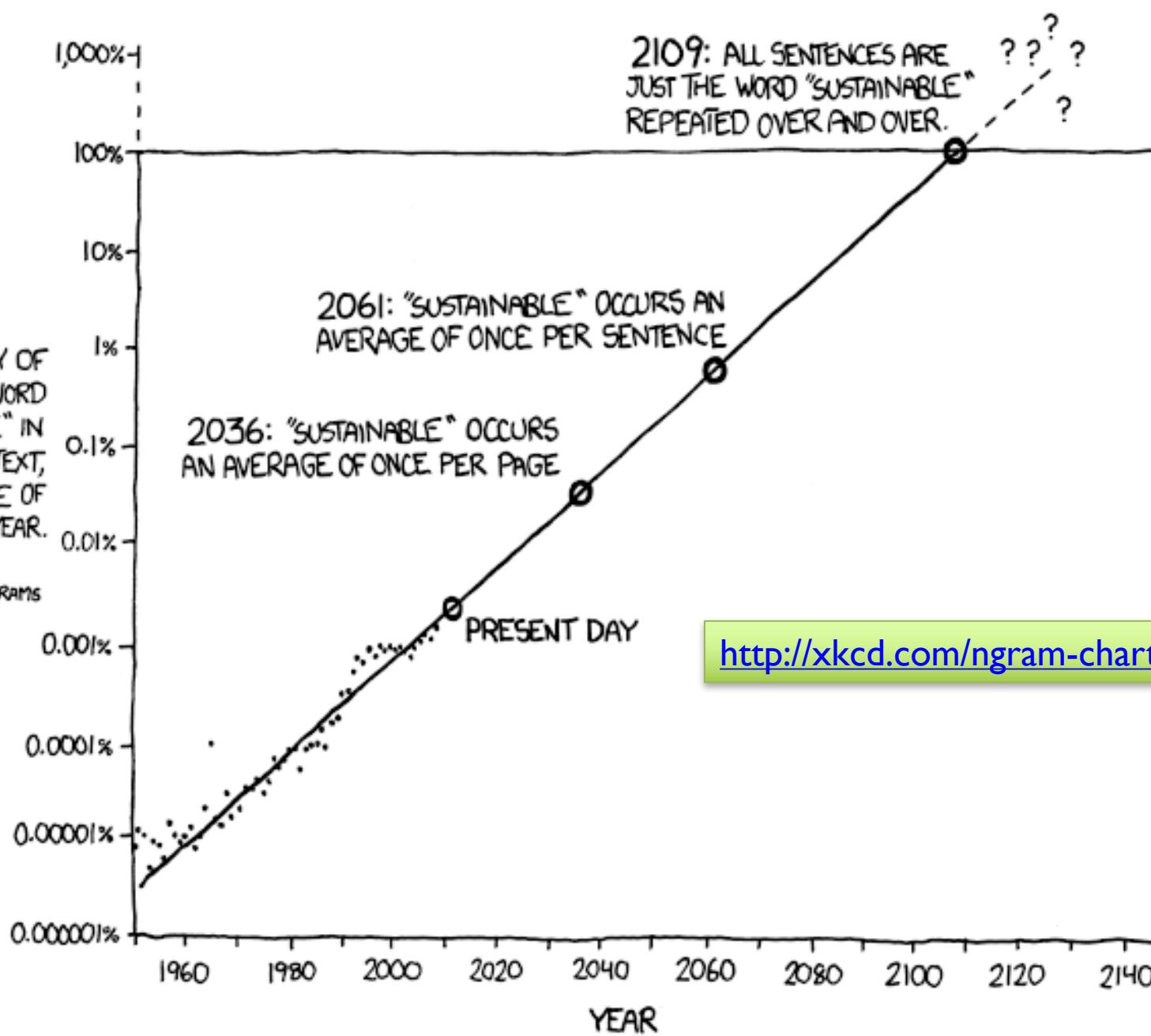
■ explosion in popularity





FREQUENCY OF  
USE OF THE WORD  
"SUSTAINABLE" IN  
US ENGLISH TEXT,  
AS A PERCENTAGE OF  
ALL WORDS, BY YEAR.

SOURCE: GOOGLE NGRAMS



<http://xkcd.com/ngram-charts/>

THE WORD "SUSTAINABLE" IS UNSUSTAINABLE.

# Why More Data Helps: A Demo

- Data:
  - All 5-grams that appear  $\geq 40$  times in a corpus of 1M English books
    - approx 80B words
    - 5-grams: 30Gb compressed, 250-300Gb uncompressed
    - Each 5-gram contains frequency distribution over *years*
  - Wrote code to compute
    - $\Pr(A,B,C,D,E|C=\text{affect} \text{ or } C=\text{effect})$
    - $\Pr(\text{any subset of } A,\dots,E|\text{any other fixed values of } A,\dots,E \text{ with } C=\text{affect} \text{ V effect})$
  - Demo:
    - `/Users/wcohen/Documents/code/pyhack/bigml`
    - eg: `python ngram-query.py data/aeffect-train.txt _B effect _-`

# Big ML c. 2001 (Banko & Brill, “Scaling to Very Very Large...”, ACL 2001)

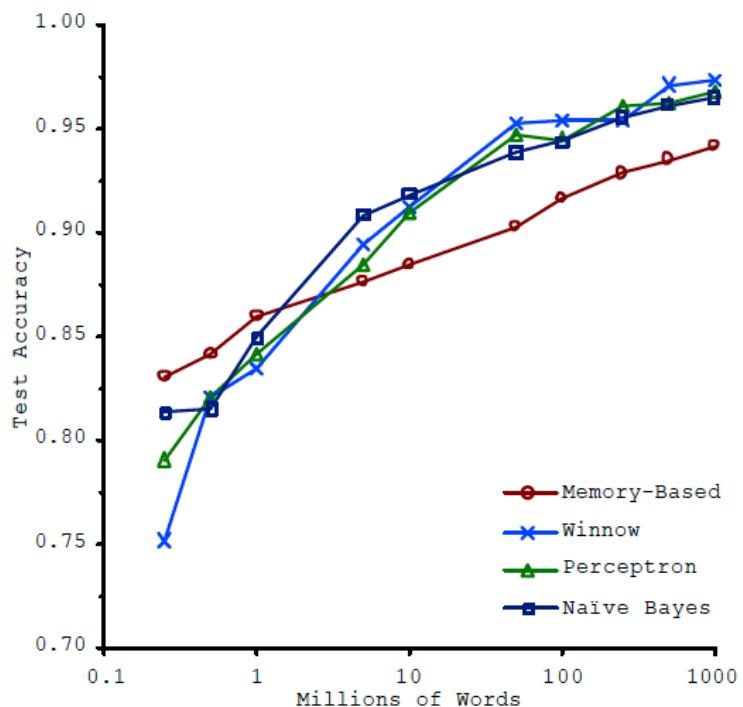


Figure 1. Learning Curves for Confusion Set Disambiguation

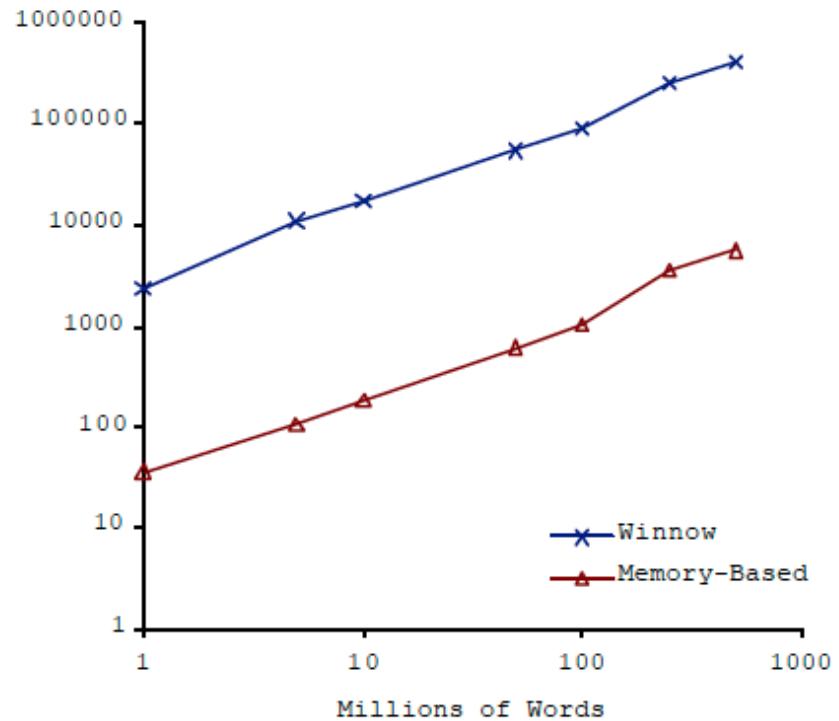


Figure 2. Representation Size vs. Training Corpus Size

Task: distinguish pairs of easily-confused words  
("affect" vs "effect") in context

# Why More Data Helps

- Data:
  - All 5-grams that appear  $\geq 40$  times in a corpus of 1M English books
    - approx 80B words
    - 5-grams: 30Gb compressed, 250-300Gb uncompressed
    - Each 5-gram contains frequency distribution over *years*
  - Wrote code to compute
    - $\Pr(A,B,C,D,E|C=\text{affect} \text{ or } C=\text{effect})$
    - $\Pr(\text{any subset of } A,\dots,E|\text{any other fixed values of } A,\dots,E \text{ with } C=\text{affect V effect})$
- Observations [from **playing with data**]:
  - Mostly **effect** not **affect**
  - Most common word before **affect** is **not**
  - After **not effect** most common word is **a**
  - ...

# So in 2001.....

- We're learning:
  - “there's no data like more data”
  - For many tasks, there's no real *substitute* for using lots of data

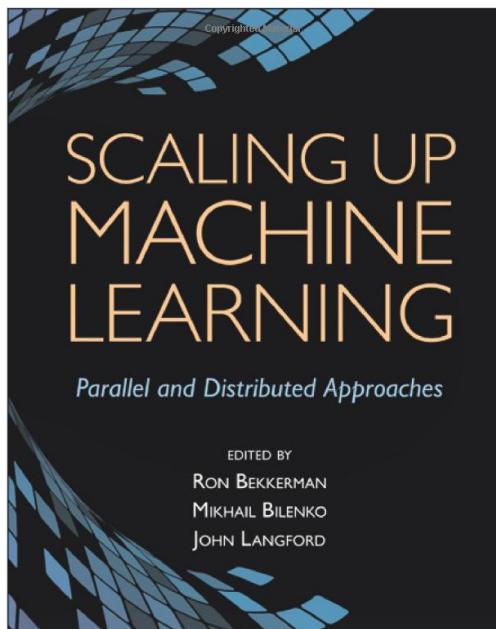
# ...and in 2009

Eugene Wigner's article "*The Unreasonable Effectiveness of Mathematics in the Natural Sciences*" examines why so much of physics can be neatly explained with simple mathematical formulas such as  $f = ma$  or  $e = mc^2$ . Meanwhile, sciences that involve human beings rather than elementary particles have proven more resistant to elegant mathematics. Economists suffer from physics envy over their inability to neatly model human behavior. An informal, incomplete grammar of the English language runs over 1,700 pages.

Perhaps when it comes to natural language processing and related fields, we're doomed to complex theories that will never have the elegance of physics equations. But if that's so, we should stop acting as if our goal is to author extremely elegant theories, and instead embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data.

Norvig, Pereira, Halevy, "The Unreasonable Effectiveness of Data", 2009

# ...and in 2012



[Arthur Gretton](#), [Michael Mahoney](#), [Mehryar Mohri](#), [Ameet Talwalkar](#)

Gatsby Unit, UCL; Stanford; Google Research; UC Berkeley

**Workshop: Low-rank Methods for Large-scale Machine Learning**

7:30am - 6:30pm Saturday, December 11, 2010

[Joseph Gonzalez](#), [Sameer Singh](#), [Graham Taylor](#), [James Bergstra](#), [Alice Zheng](#), [Misha Bilenko](#), [Yucheng Low](#), [Yoshua Bengio](#), [Michael Franklin](#), [Carlos Guestrin](#), [Andrew McCallum](#), [Alexander Smola](#), [Michael Jordan](#), [Sugato Basu](#)

Carnegie Mellon University; University of Massachusetts, Amherst; New York University; Harvard; Microsoft Research; Microsoft Research; Carnegie Mellon University; University of Montreal; UC Berkeley; Carnegie Mellon University; UMass Amherst; Yahoo! Research; University of California; Google Research

**Workshop: Big Learning: Algorithms, Systems, and Tools for Learning at Scale**

**Location:** Montebajo: Theater

Dec 2011

SMLA Workshop 2010

*29 June - 01 July, 2010, Bradford, UK*

**International Workshop on  
Scalable Machine Learning and Applications (SMLA-10)  
In conjunction with [CIT 2010](#)**

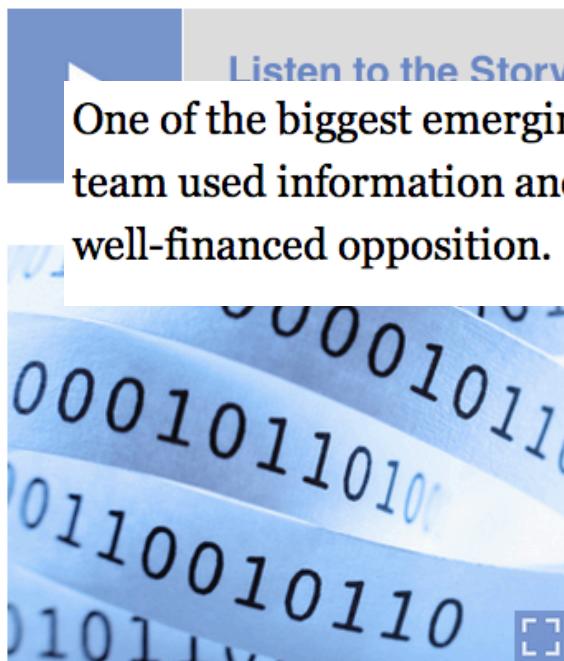
# ...and in 2013

news > opinion > commentary

## Forget YOLO: Why 'Big Data' Should Be The Word Of The Year

by GEOFF NUNBERG

December 20, 2012 10:58 AM



Listen to the Story 

+ Playlist

One of the biggest emerging stories about the campaign that has ended is how Mr. Obama's team used information and technology to outmatch and outwit a galvanized and incredibly well-financed opposition.

probably  
"frankens" it was the buzz of Silicon Valley like  
*Wired* and *The Economist*, and it was the buzz of Silicon Valley and Davos. And if the phrase wasn't as familiar to many people as "Etch A Sketch" and "47 percent," Big Data had just as much to do with President Obama's victory as they did.

Whether it's explicitly mentioned or not, the Big Data phenomenon has been all

Adam Gryko/Studio

it will be around a lot longer than "gangnam style." It's about intrusions on our data sweeps or the ads that track us as we wander around the Web. It has even turned statistics into a sexy major. So if you haven't heard the phrase yet, there's still time — it will be around a lot longer than "gangnam style."

# ...and in 2014

← → C  **WIRED** GEAR SCIENCE ENTERTAINMENT BUSINESS SECURITY DESIGN OPINION MAGA

**INNOVATION INSIGHTS** | [analytics](#) [doctors](#) [EHR](#) [salaries](#)

## Tell Your Kids to Be Data Scientists, Not Doctors

BY LINDA BURTCH, BURTCH WORKS 06.17.14 | 5:45 PM | PERMALINK

[Share 42](#) [Tweet 48](#) [g+1 8](#) [in Share 124](#) [Pin it](#)

Home News & Commentary Authors Slideshows Video Reports White Papers Events University

STRATEGIC CIO SOFTWARE SECURITY CLOUD MOBILE BIG DATA INFRASTRUCTURE

### BIG DATA // BIG DATA ANALYTICS

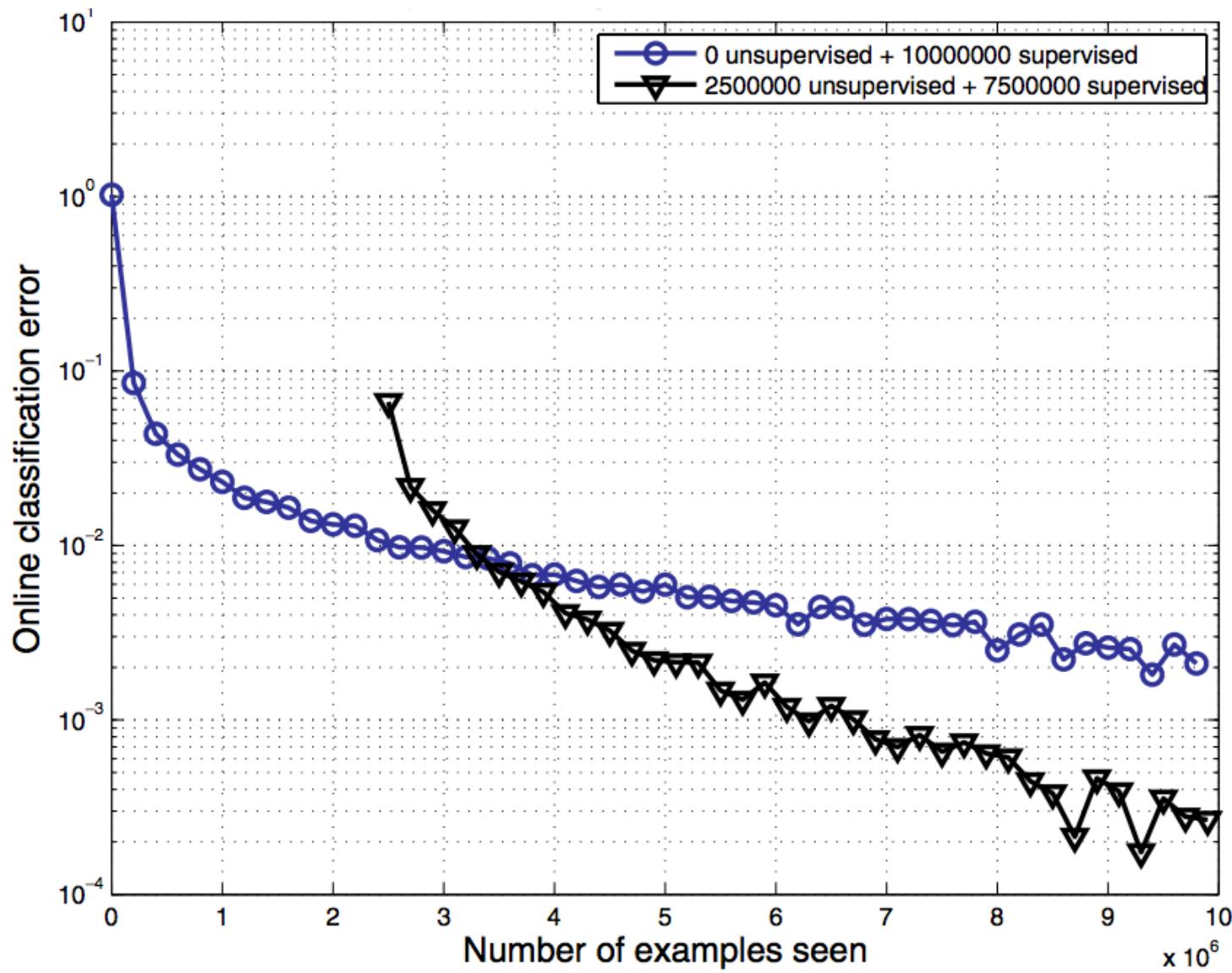
COMMENTARY  
12/2/2013  
09:06 AM

## Data Scientist: The Sexiest Job No One Has



The data scientist has been called the sexiest job of the 21st century, but it's largely going unfilled. That's a huge problem for the business world.

# Bengio, Foundations & Trends, 2009



### 3-layer net, budget of 10000000 iterations

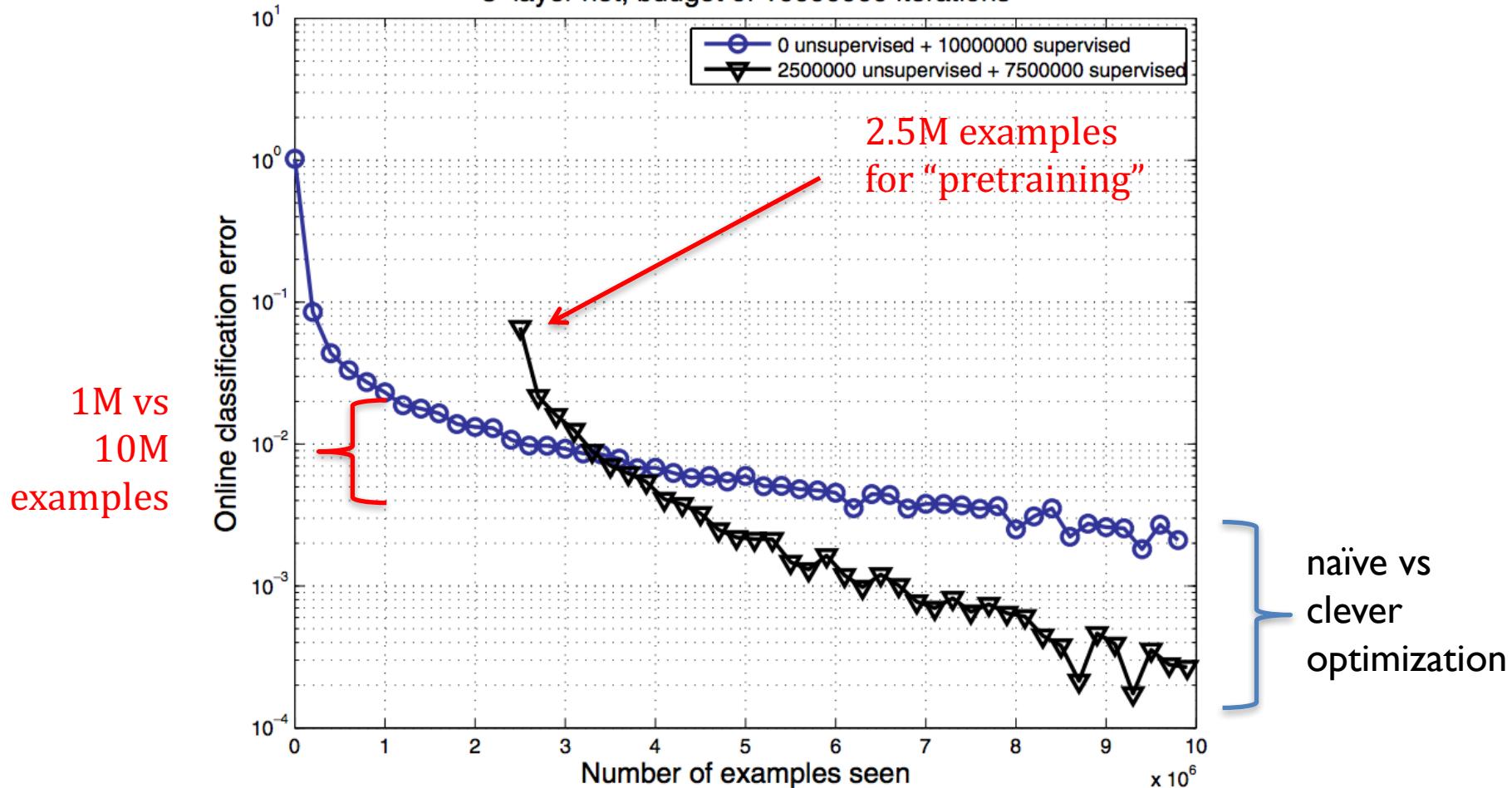


Fig. 4.2 Deep architecture trained online with 10 million examples of digit images, either with pre-training (triangles) or without (circles). The classification error shown (vertical axis, log-scale) is computed online on the next 1000 examples, plotted against the number of examples seen from the beginning. The first 2.5 million examples are used for unsupervised pre-training (of a stack of denoising auto-encoders). The oscillations near the end are because the error rate is too close to 0, making the sampling variations appear large on the log-scale. Whereas with a very large training set regularization effects should dissipate, one can see that without pre-training, training converges to a poorer apparent local minimum: unsupervised pre-training helps to find a better minimum of the online error. Experiments were performed by Dumitru Erhan.

# Today....

- Commonly used deep learning datasets:
  - Images/videos:
    - ImageNet: 20k+ categories, 14M+ images
    - MS COCO: 91 categories, 2.5M labels, 328k images
    - YouTube-M: 8M urls, 4800 classes, 0.5M hours
  - Reading comprehension:
    - Children's book test: 600k + context/query pairs
    - CNN/Daily mail: ~300k docs, 1.2M cloze questions
  - Other:
    - Ubuntu dialog: 7M+ utterances, 1M+ dialogs
    - ...

# REVIEW: ASYMPTOTIC COMPLEXITY

# How do we use very large amounts of data?

- Working with big data is *not*\* about
  - code optimization
  - learning details of todays hardware/software:
    - GraphLab, Hadoop, Spark, parallel hardware, ....
- Working with big data *is* about
  - Understanding the cost of what you want to do
  - Understanding what the tools that are available offer
  - Understanding how much can be accomplished with linear or nearly-linear operations (e.g., sorting, ...)
  - Understanding how to organize your computations so that they effectively use whatever's fast
  - Understanding how to test/debug/verify with large data

\* according to William

# Asymptotic Analysis: Basic Principles

Usually we only care about positive  $f(n), g(n)$ ,  $n$  here...

$$f(n) \in O(g(n)) \text{ iff } \exists k, n_0 : \forall n > n_0, f(n) \leq k \cdot g(n)$$

$$f(n) \in \Omega(g(n)) \text{ iff } \exists k, n_0 : \forall n > n_0, f(n) \geq k \cdot g(n)$$

# Asymptotic Analysis: Basic Principles

Less pedantically:

$$f(n) = O(g(n)) \text{ iff } \exists k, n_0 : \forall n > n_0, f(n) \leq k \cdot g(n)$$

$$f(n) = \Omega(g(n)) \text{ iff } \exists k, n_0 : \forall n > n_0, f(n) \geq k \cdot g(n)$$

Some useful rules:

$$O(n^4 + n^3) = O(n^4) \quad \text{Only highest-order terms matter}$$

$$O(3n^4 + 127n^3) = O(n^4) \quad \text{Leading constants don't matter}$$

$$O(\log n^4) = O(4 \cdot \log n) = O(\log n)$$

Degree of something in a log doesn't matter

# Back to rule pruning....

## Algorithm

- Fit the POS,NEG exampleWhile POS isn't empty:
  - Let  $R$  be "if True  $\rightarrow$  pos"
  - While NEG isn't empty:
    - Pick the "best" [1] condition  $c$  of the form " $x_i=True$ " or " $x_i=false$ "
    - Add  $c$  to the LHS of  $R$
    - Remove examples that don't satisfy  $c$  from NEG
  - Add  $R$  to the rule set [2]
  - Remove examples that satisfy  $R$  from POS
- Prune the rule set:
  - For each condition  $c$  in the rule set:
    - Evaluate the accuracy of the ruleset w/o  $c$  on heldout data
  - If removing any  $c$  improves accuracy
    - Remove  $c$  and repeat the pruning step

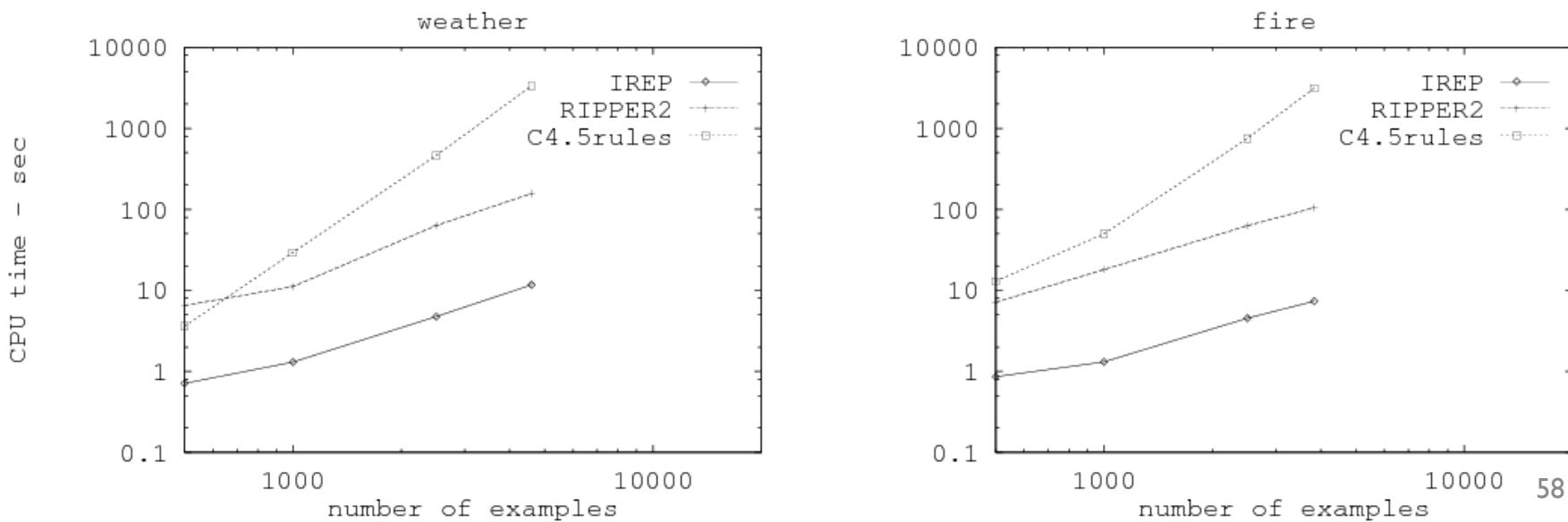
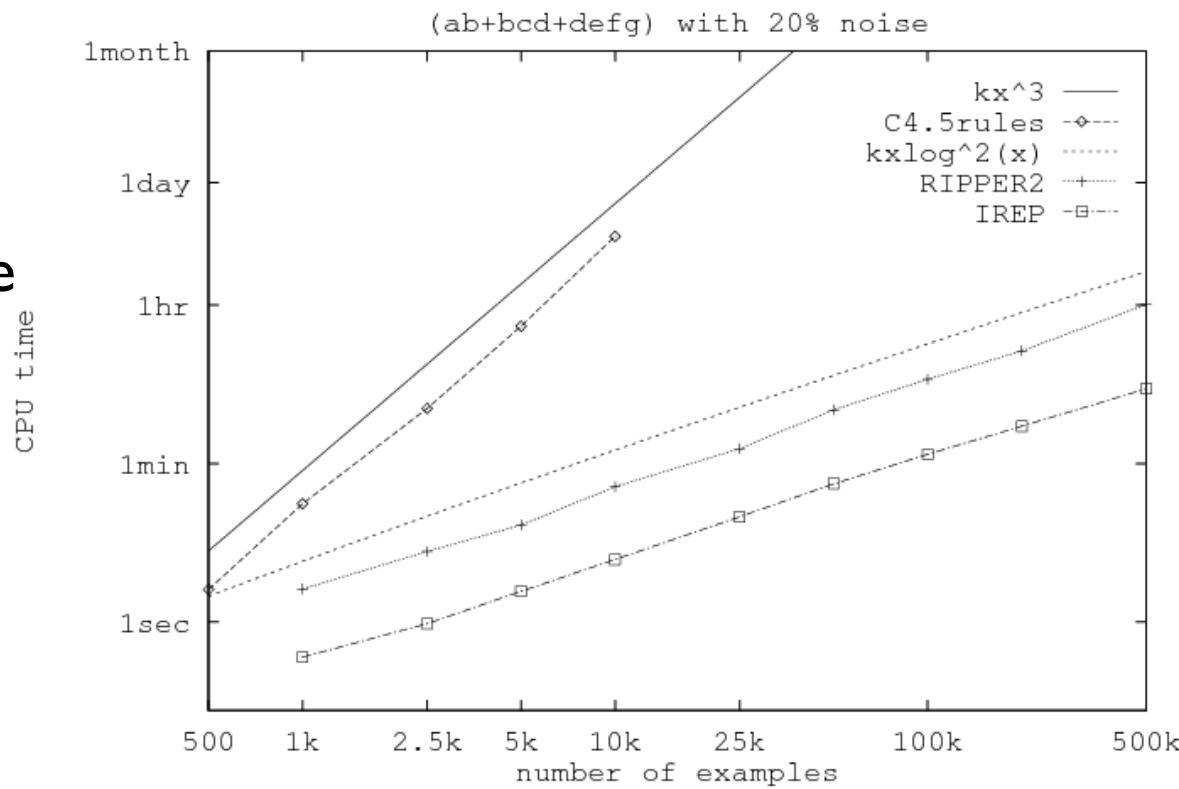
## Analysis

- Assume  $n$  examples
- Assume  $m$  conditions in rule set
- Growing rules takes time at least  $\Omega(m^*n)$  if evaluating  $c$  is  $\Omega(n)$
- When data is clean  $m$  is small, fitting takes linear time
- When  $k\%$  of data is noisy,  $m$  is  $\Omega(n^*0.01*k)$  so growing rules takes  $\Omega(n^2)$
- Pruning a rule set with  $m = 0.01*kn$  extra conditions is *very* slow:  $\Omega(n^3)$  if implemented naively

[1] "Best" is wrt some statistics on  $c$ 's coverage of POS,NEG

[2]  $R$  is now of the form "if  $x_1=_$  and  $x_2=_$  and ...  $\rightarrow$  pos"

*Empirical  
analysis of  
complexity:  
plot run-time  
on a log-log  
plot and  
measure the  
slope (using  
linear  
regression)*



# Where do asymptotics break down?

- When the constants are too big
  - or  $n$  is too small
- When we can't predict what the program will do
  - Eg, how many iterations before convergence?  
Does it depend on data size or not?
  - This is when you need experiments
- When there are different types of operations with different costs
  - We need to understand what we should count

# What do we count?

- Compilers don't warn Jeff Dean. Jeff Dean warns compilers.
- Jeff Dean builds his code before committing it, but only to check for compiler and linker bugs.
- Jeff Dean writes directly in binary. He then writes the source code as a documentation for other developers.
- Jeff Dean once shifted a bit so hard, it ended up on another computer.
- When Jeff Dean has an ergonomic evaluation, it is for the protection of his keyboard.
- gcc -O4 emails your code to Jeff Dean for a rewrite.
- When he heard that Jeff Dean's autobiography would be exclusive to the platform, Richard Stallman bought a Kindle.
- Jeff Dean puts his pants on one leg at a time, but if he had more legs, you'd realize the algorithm is actually only  $O(\log n)$

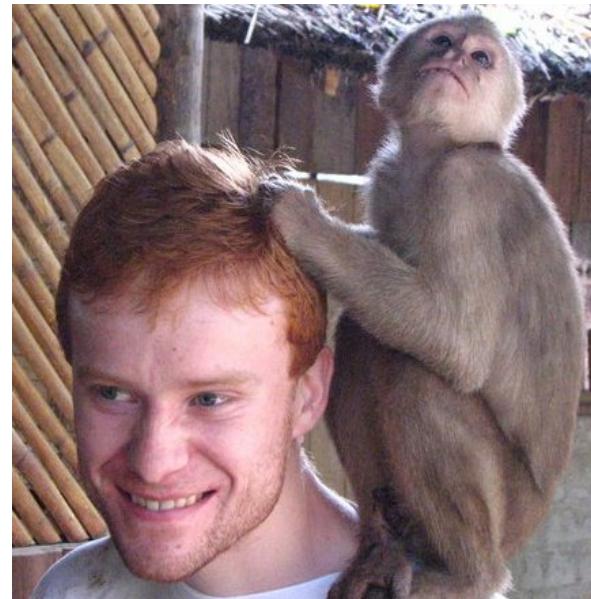


# Numbers (Jeff Dean says) Everyone Should Know

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

# Update: Colin Scott, UCB

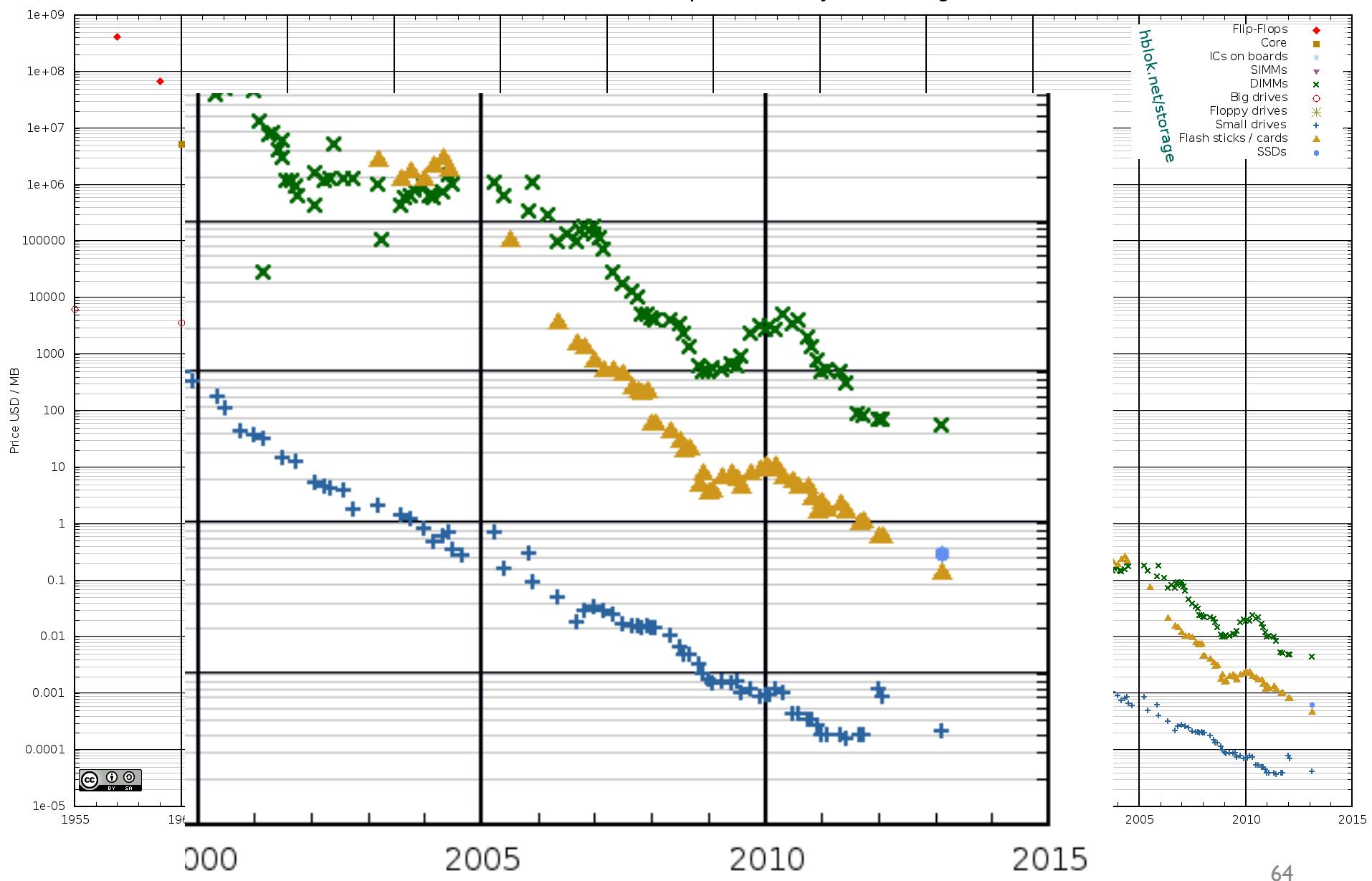
[file:///Users/wcohen/Documents/code/interactive\\_latencies/interactive\\_latency.html](file:///Users/wcohen/Documents/code/interactive_latencies/interactive_latency.html) - \*may need to open this from shell



# What's Happening with Hardware?

- Clock speed: stuck at 3Ghz for ~ 10 years
- Net bandwidth doubles ~ 2 years
- Disk bandwidth doubles ~ 2 years
- SSD bandwidth doubles ~ 3 years
- Disk seek speed doubles ~ 10 years
- SSD latency nearly saturated

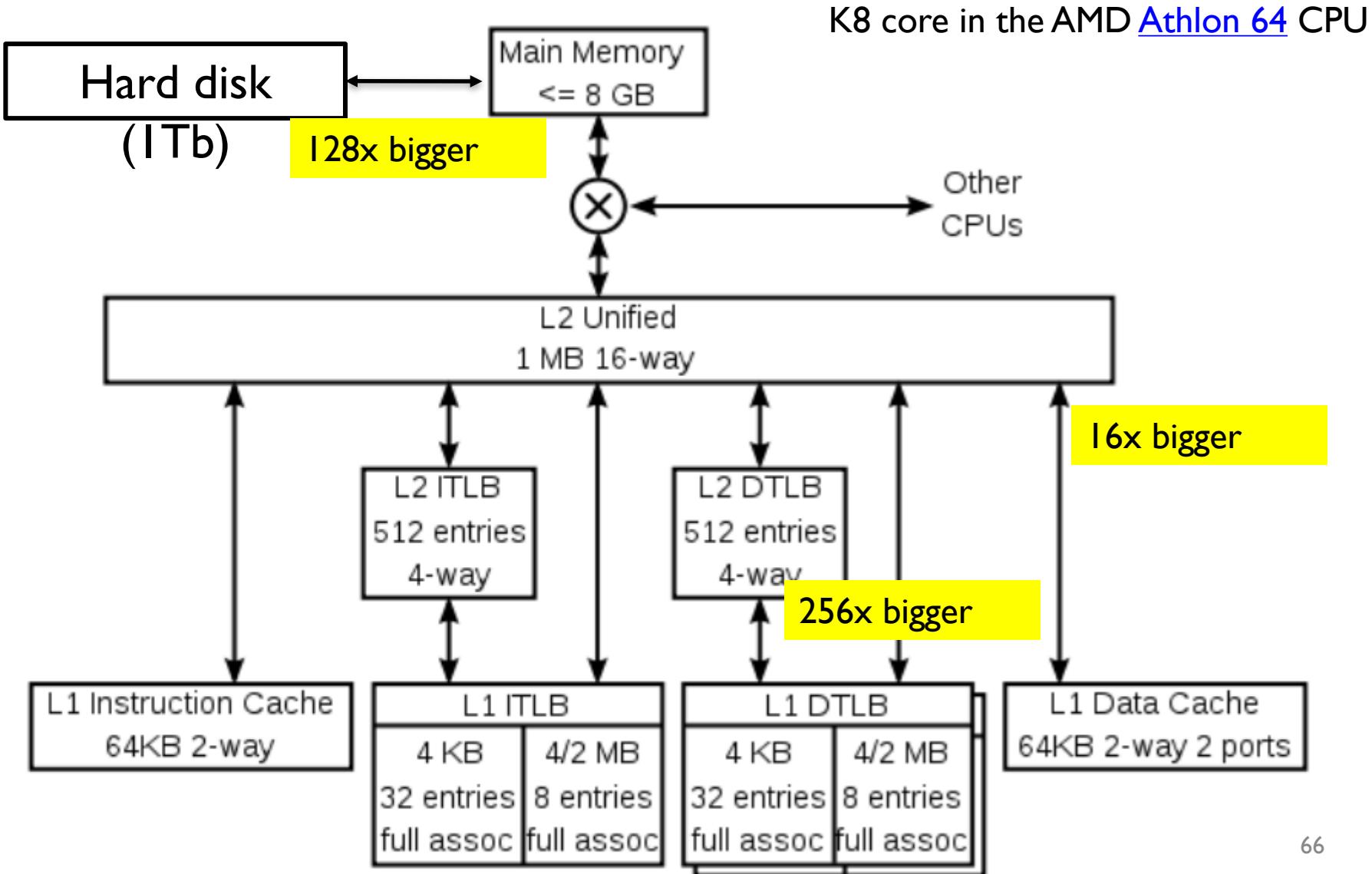
## Historical Cost of Computer Memory and Storage



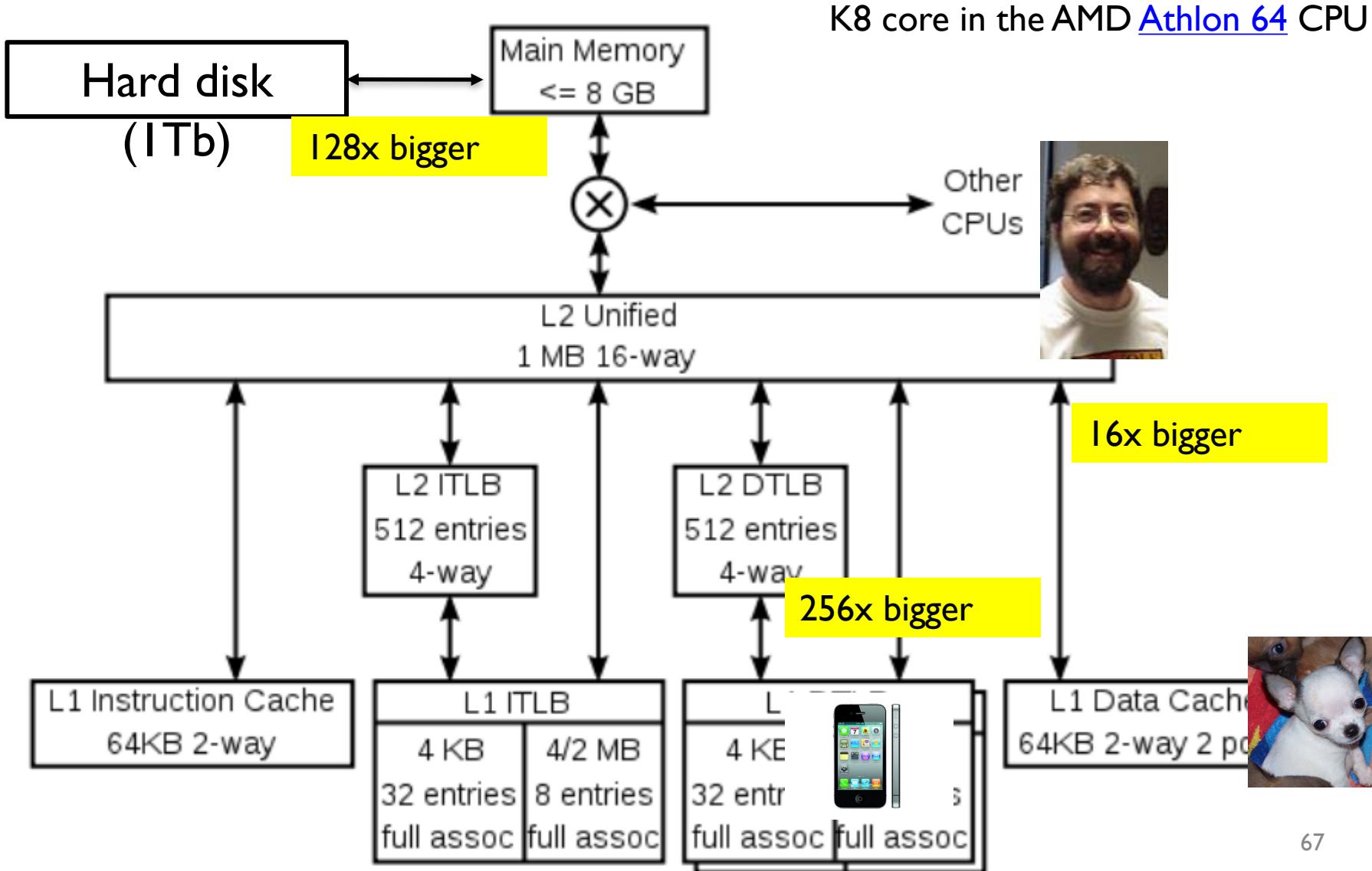
# Numbers (Jeff Dean says) Everyone Should Know

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

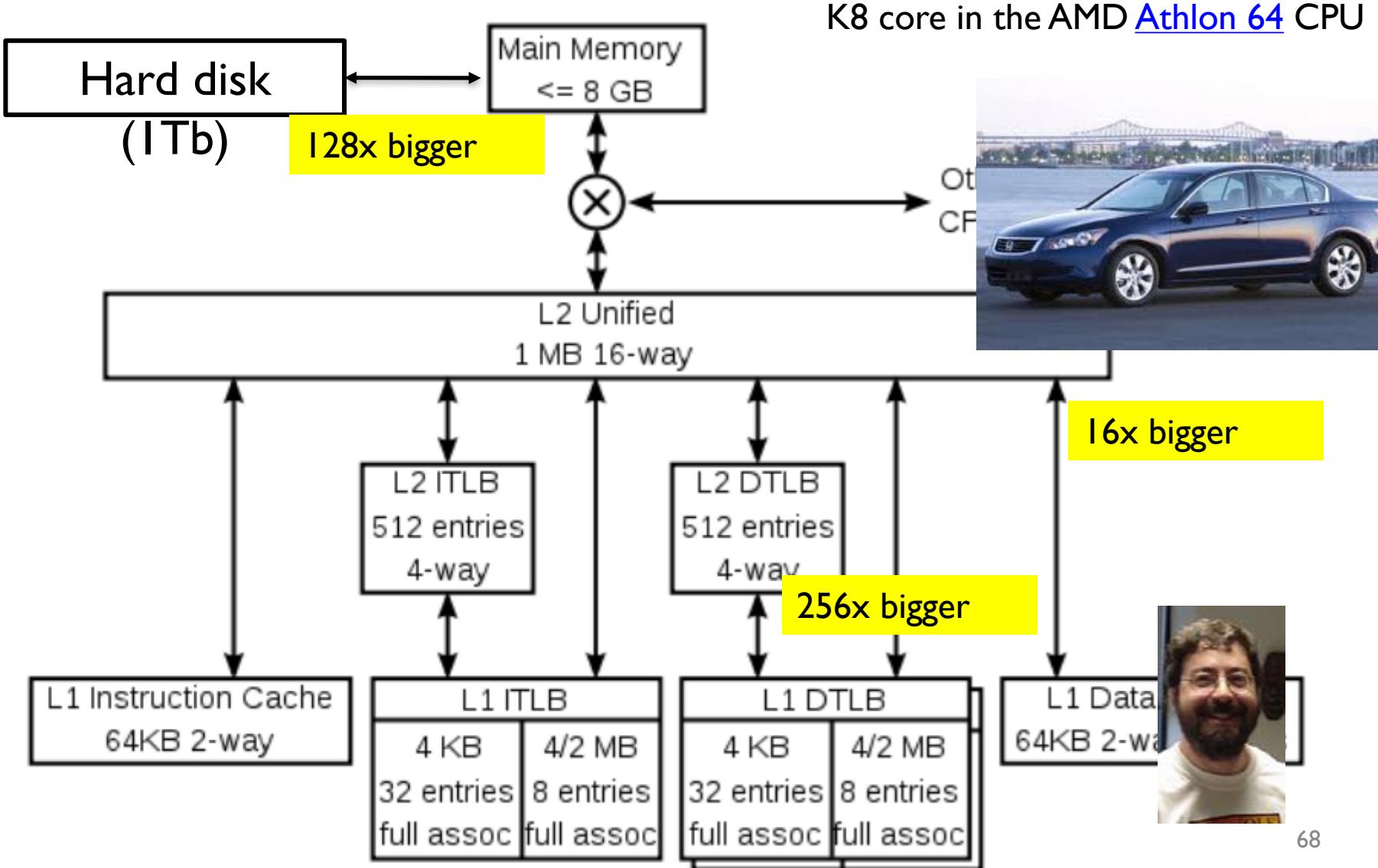
# A typical CPU (not to scale)



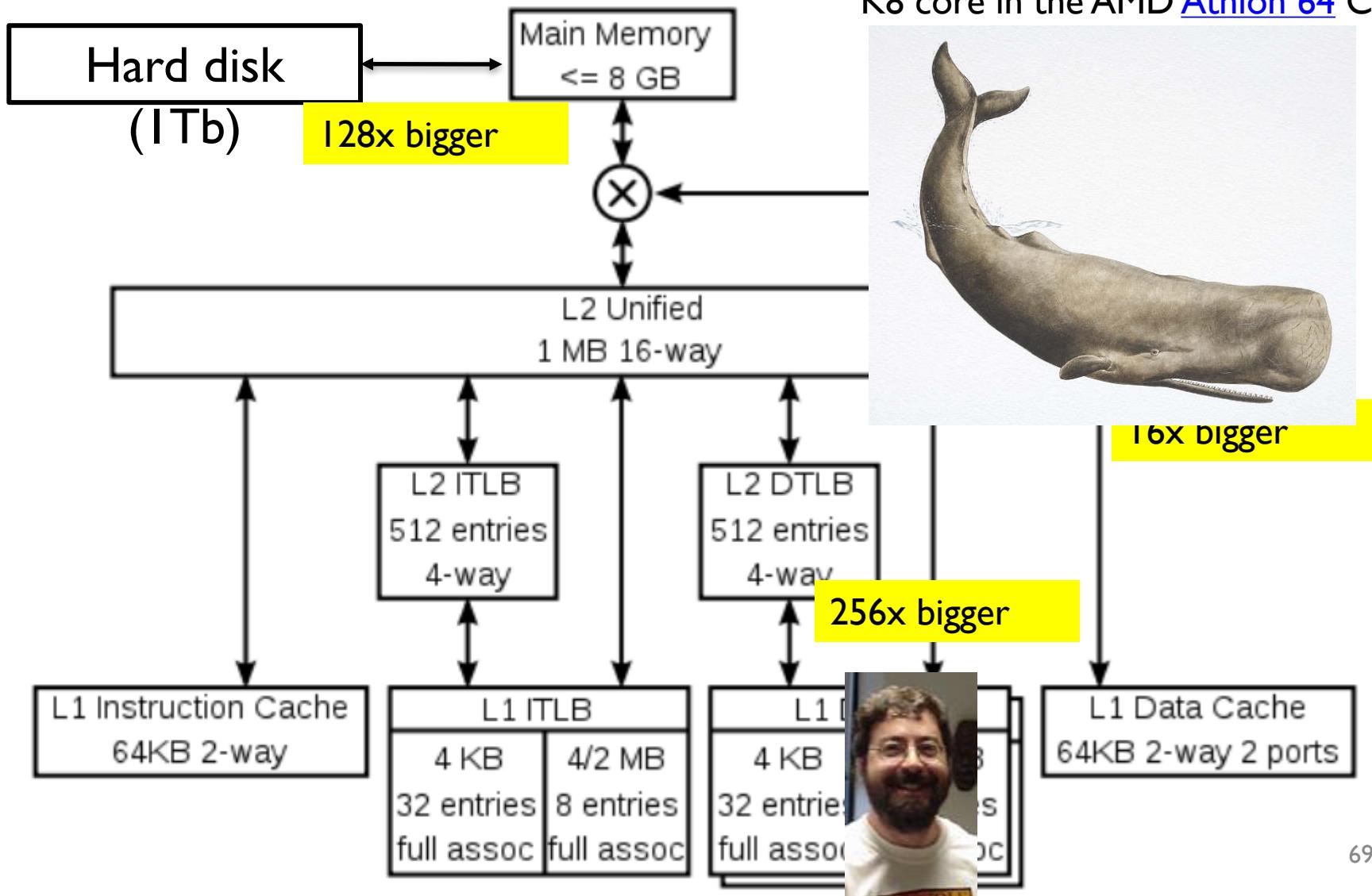
# A typical CPU (not to scale)



# A typical CPU (not to scale)



# A typical CPU (not to scale)



# A typical disk



# Numbers (Jeff Dean says) Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns $\approx 10x$
Mutex lock/unlock	100 ns
Main memory reference	100 ns $\approx 15x$
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

40x

≈ 100,000x

# What do we count?

- Compilers don't warn Jeff Dean. Jeff Dean warns compilers.
- ....
- Memory access/instructions are *qualitatively different* from disk access
- Seek are *qualitatively different* from sequential reads on disk
- Cache, disk fetches, etc work best when you stream through data *sequentially*
- Best case for data processing: stream through the data *once* in *sequential order*, as it's found on disk.



# Other lessons -?



## Encoding Your Data

- CPUs are fast, memory/bandwidth are precious, ergo...
    - Variable-length encodings
    - Compression
    - Compact in-memory representations
  - Compression very important aspect of many systems
    - inverted index posting list formats
    - storage systems for persistent data
- \* but not important enough for this class's assignments....

# What/How

- Next lecture: probability review and Naïve Bayes.
- Homework:
  - Watch the review lecture I linked to on the wiki
- I'm *not* going to repeat it