Data and Application

Tutorial of Parameter Server

Mu Li
CSD@CMU & IDL@Baidu
muli@cs.cmu.edu

About me

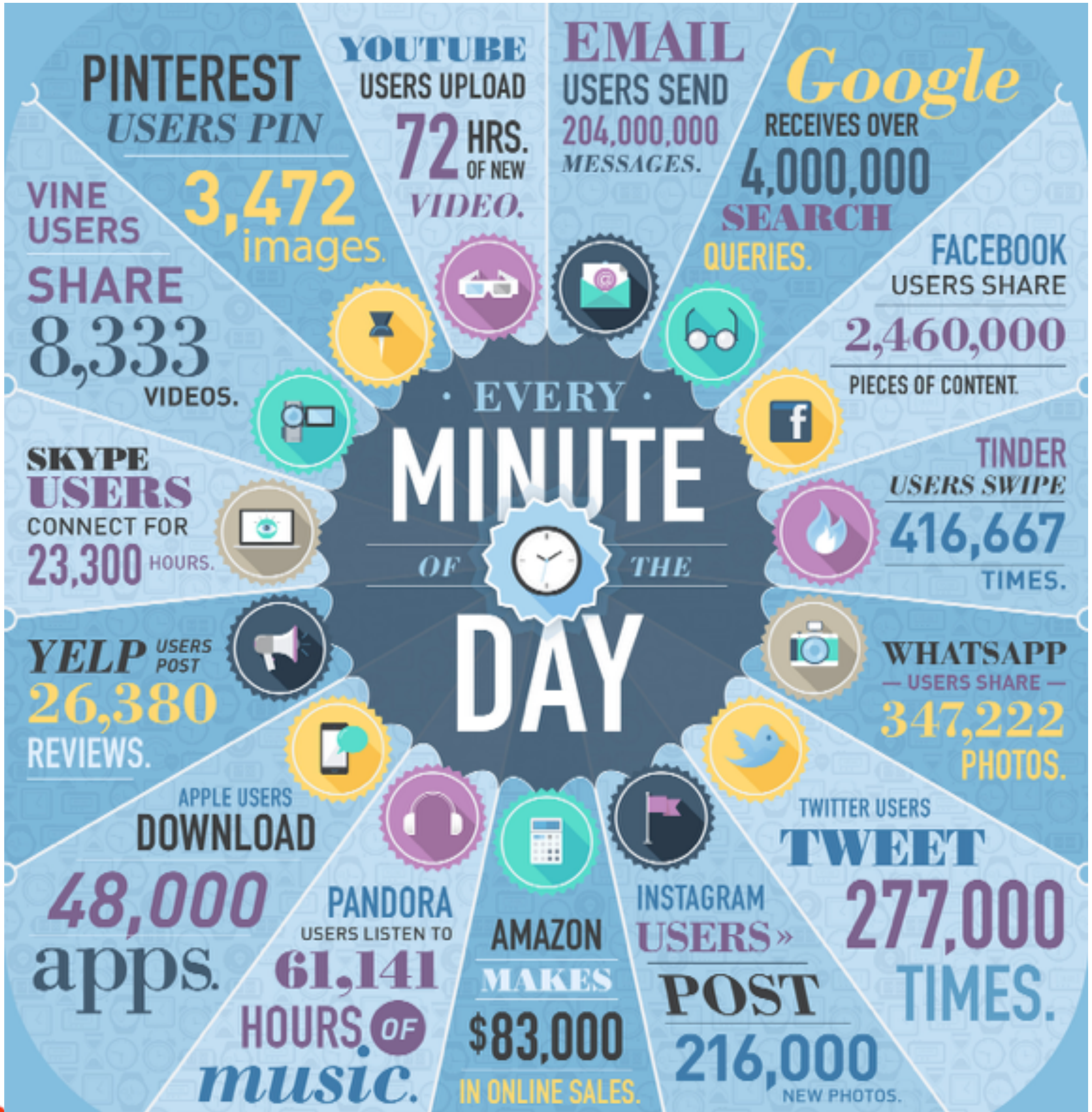
- Ph.D student working with Alex Smola and Dave Andersen
  - large scale machine learning theory, algorithm, application, and distributed system
- Senior architect at Baidu
  - the largest search engine at China, >60% market share
  - working on distributed machine learning systems for computational ads

About this tutorial

- Focus on the design and implementation of large scale machine learning systems
- Parallel and distributed algorithms
- Several coding exercises
- Provide real datasets and machines

Application & Data

There are lots of data

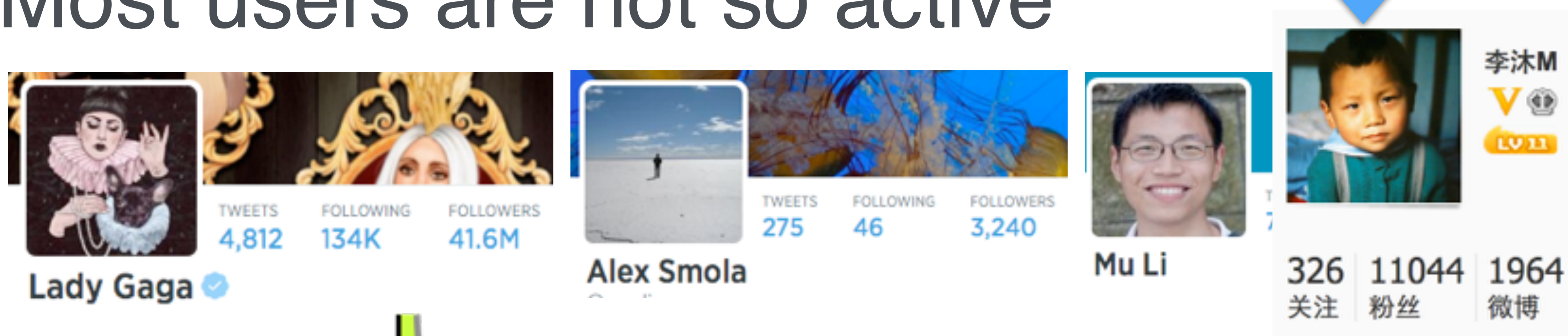


- Text
- Images
- Voices
- Videos
- All about user activities: personalization

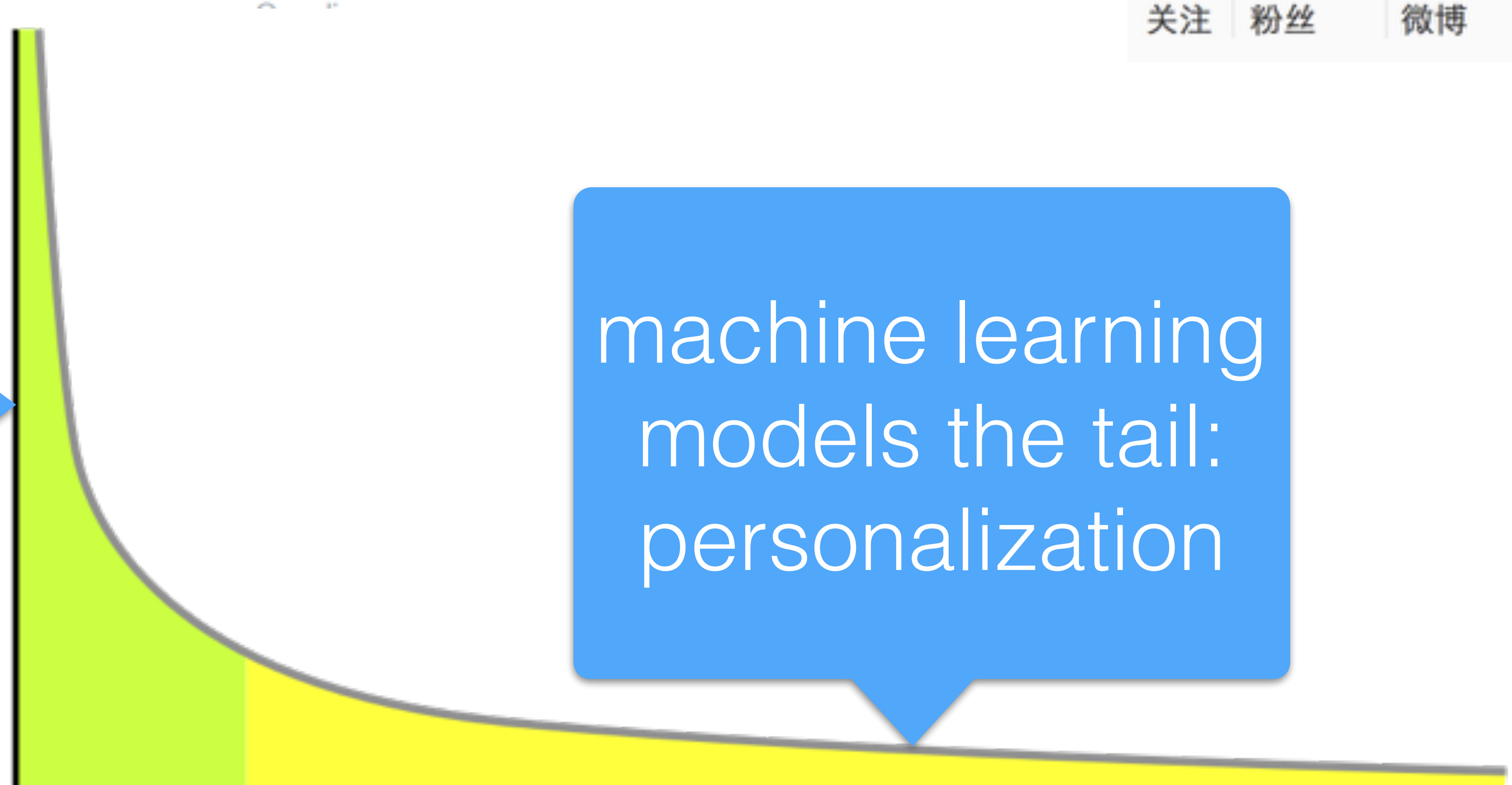
Data are sparse

- Most categories have only fe
- Most users are not so active

not true, indeed more active than Alex :)



simple statistic tools model the head well

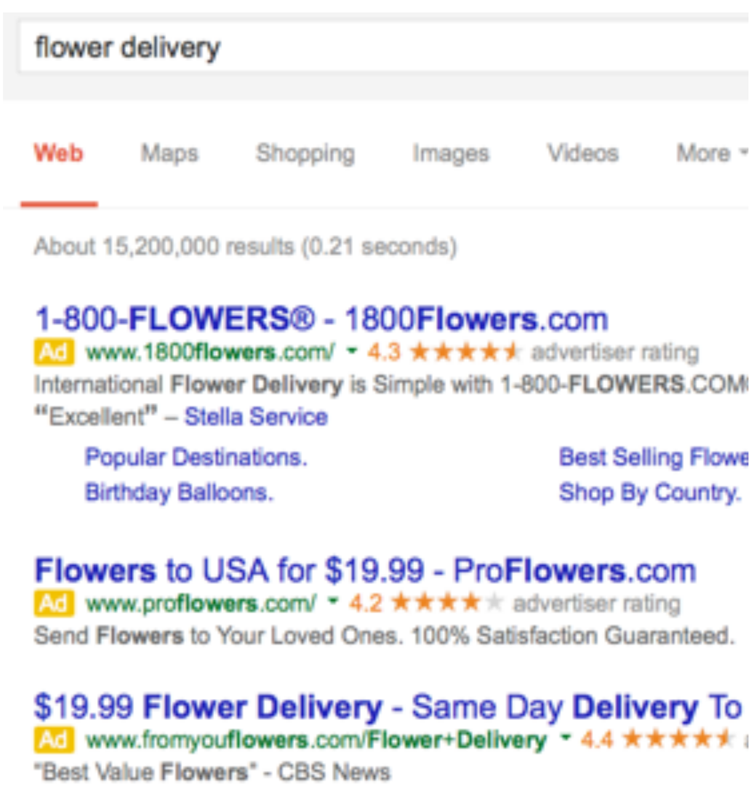


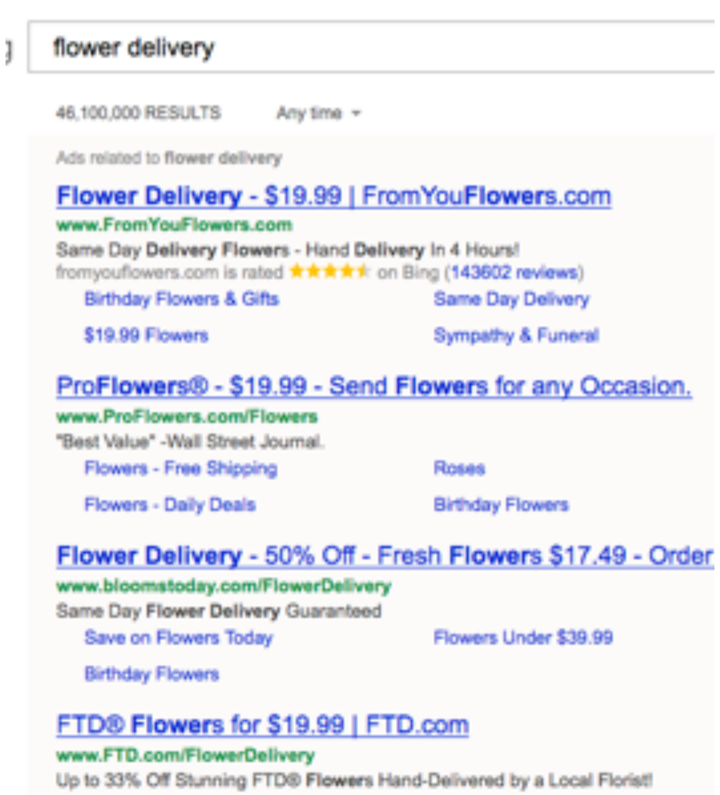
machine learning models the tail: personalization

Online Advertising

- The major revenue source of internet search companies

query: “flower delivery” results from baidu, google, bing





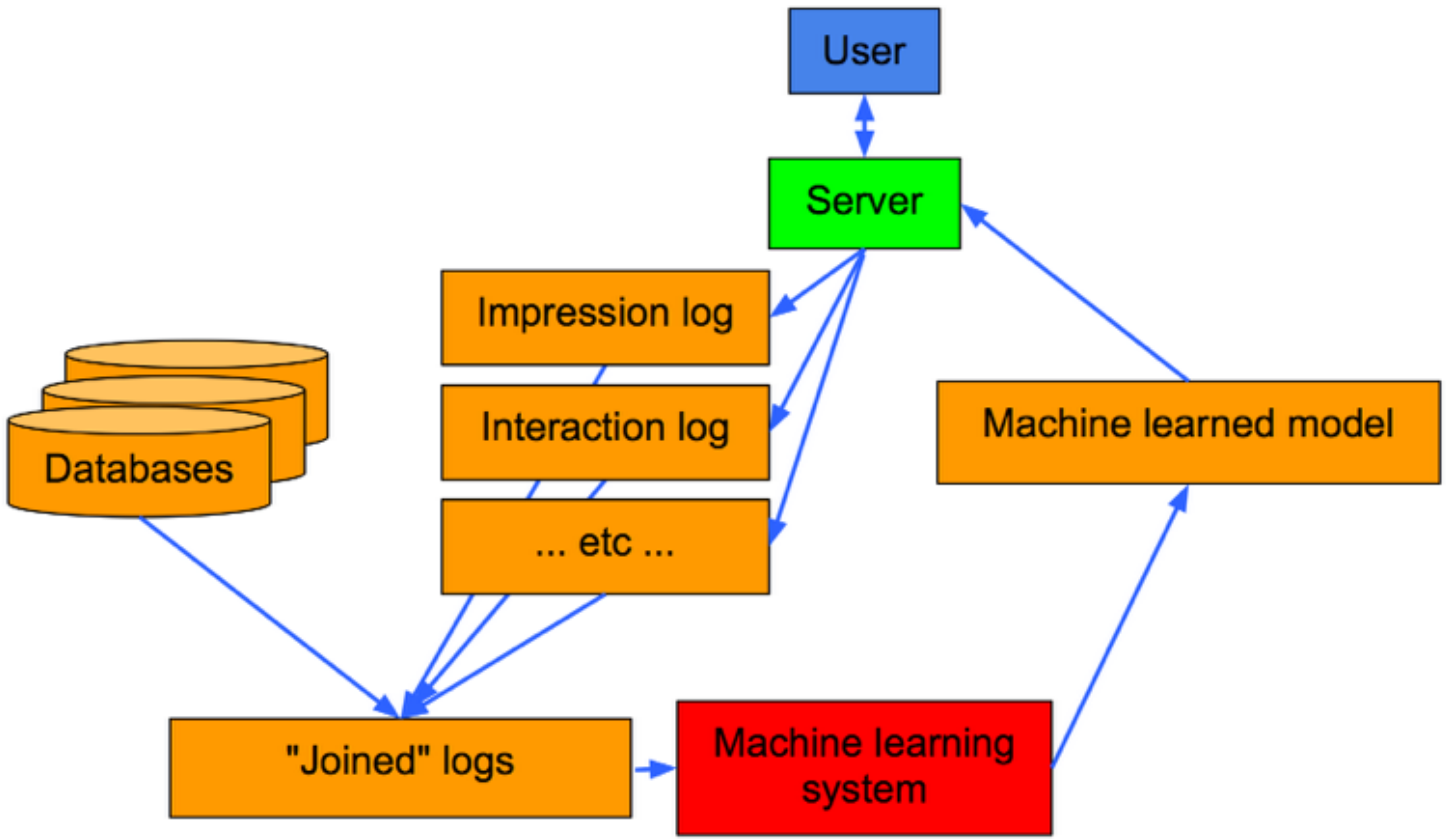
Computational Advertising

- Search companies charge advertisers if their Ads were clicked by users
- Display position is the scarce resource
- Ads are ranked by

$$p(click \mid Ad, user, scene) \times bid\_price(Ad)$$

- bid prices are given (studied by electronic mechanism design)
- our goal: predict the click-though rate

System architecture



from Google Sibyl

Machine Learning Approach

- Represent {Ad, user, scene} as a feature vector *x*, let *y* (clicked or not clicked) be the label, then model *p(y|x)*
  - A common way

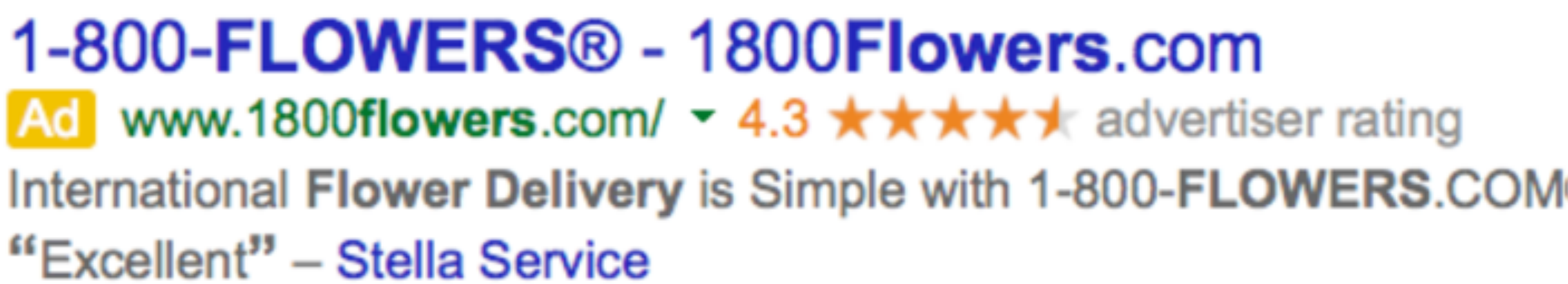
$$p(y|x, w) = \frac{1}{1 + \exp(y\langle x, w\rangle)}$$

  - then learn *w* by logistic regression
- Also increasing interests on deep learning

Feature Engineering

- Feature engineering is the most effective way to improve the model performance
  - even still true for deep learning
- Easy way to add domain knowledge into the model
- Often contain multiple feature groups
  - three major sources: ads, users, advertisers

N-grams



- uni-gram: international, flower, delivery, …
- bi-gram: international flower, flower delivery, …
- tri-gram: international flower delivery, …
- for short text, even desirable generate all possible n-grams, then filter out unimportant ones

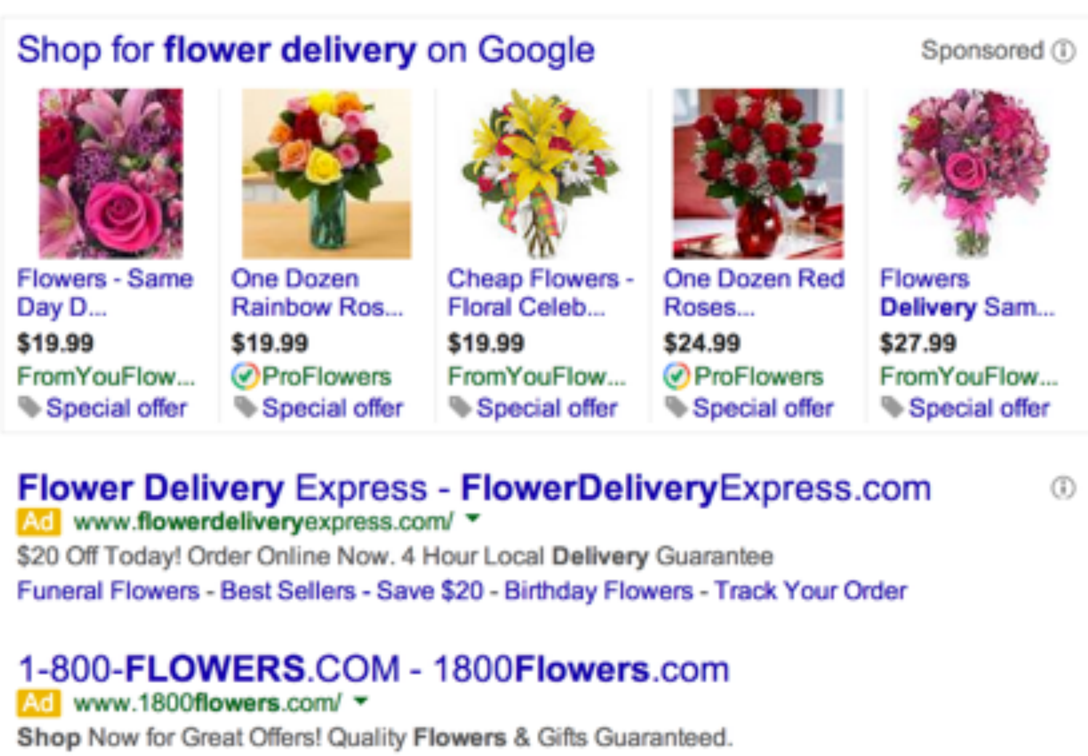
Style

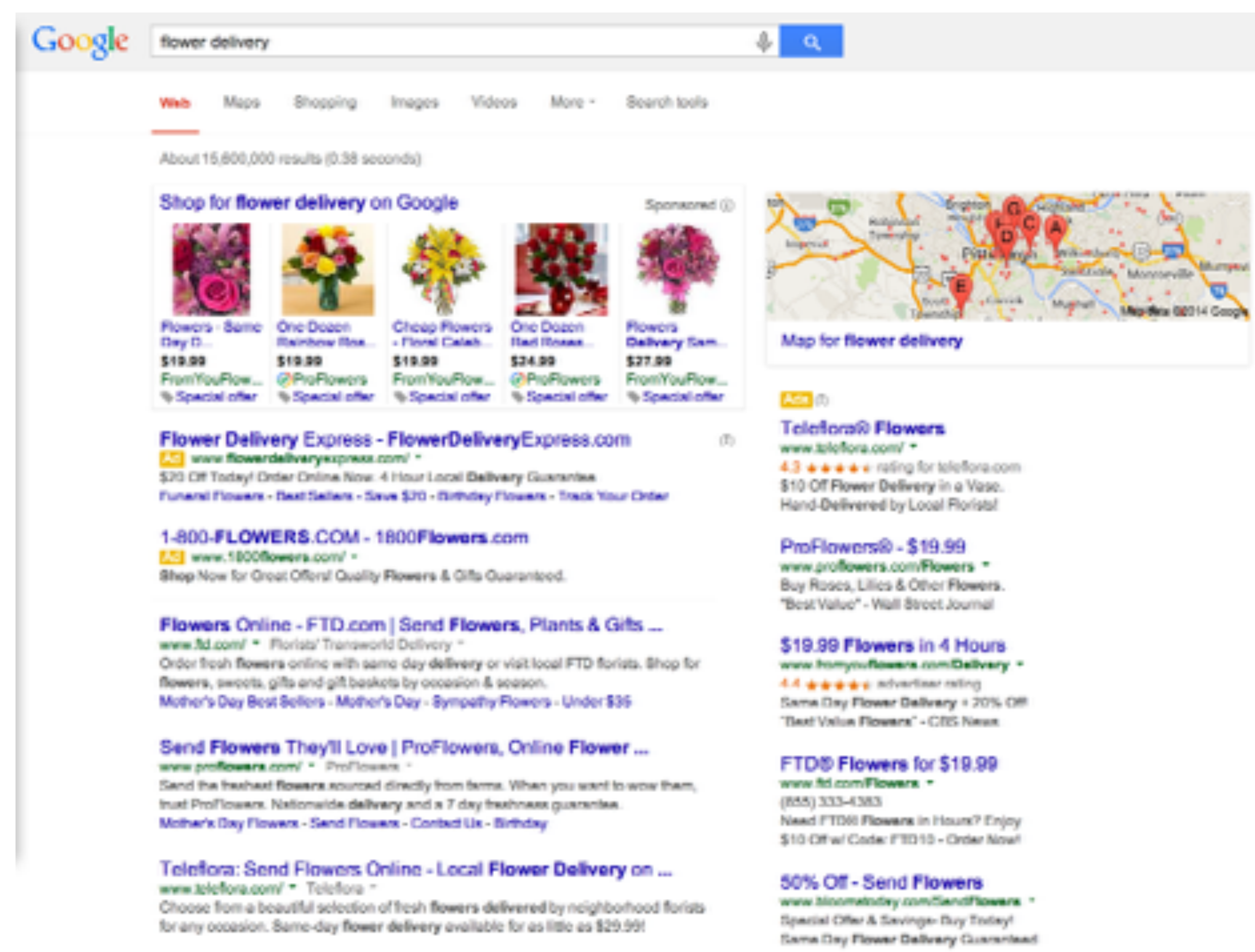
Bold text

1-800-**FLOWERS**.COM - 1800**Flowers**.com
Ad www.1800**flowers**.com/
Shop Now for Great Offers! Quality **Flowers** & Gifts Guaranteed.

Layout

Images



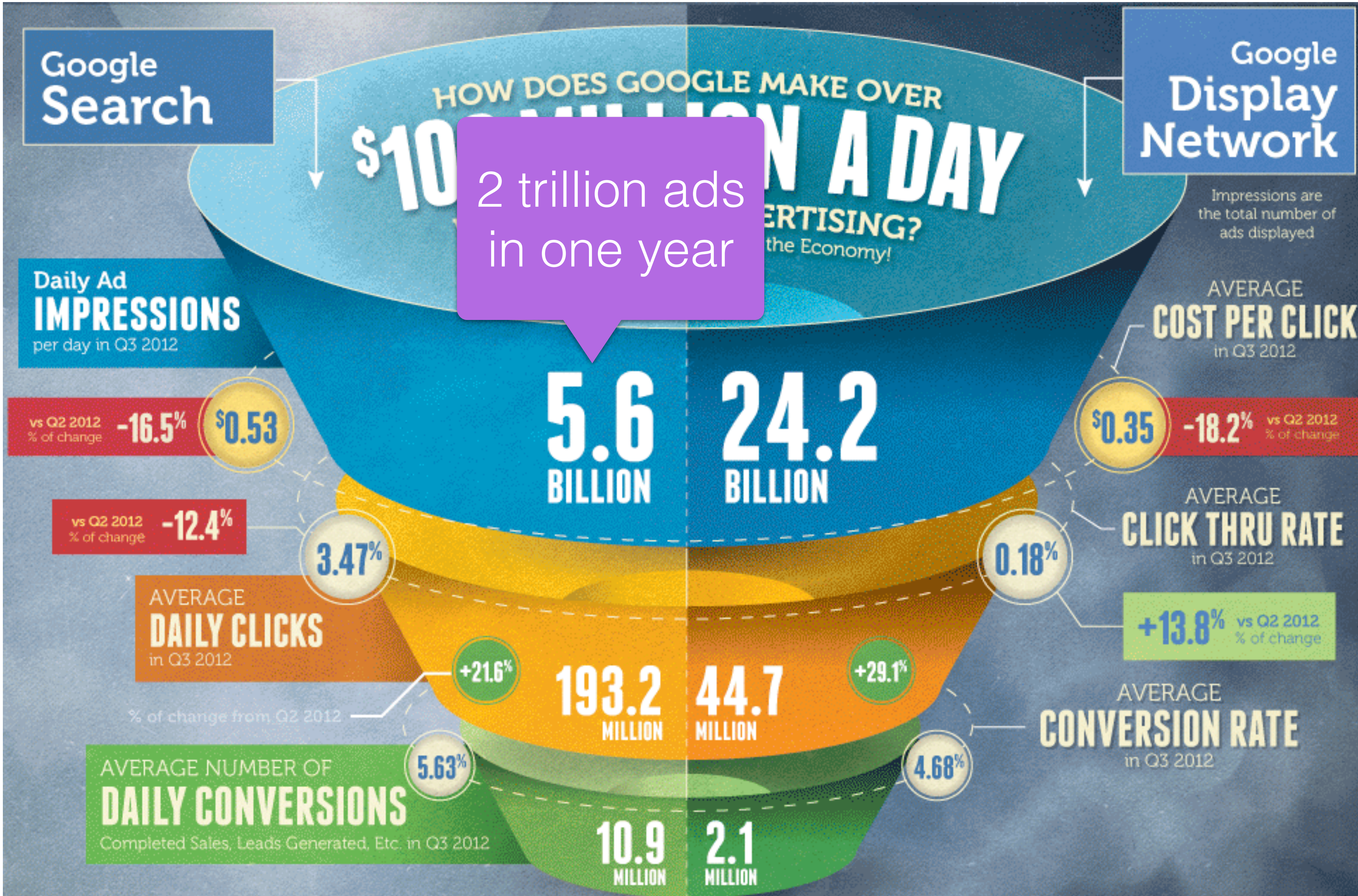


Personalization

- Users profile
  - gender, age, location, …
- Advertiser profile
  - category, reputation, …
- Session
  - a sequence of activities

Feature combination

- Given two feature groups
  - {(a,1), (b,0)}
  - {(A, 0), (B, 1)}
- Produce a combination group
  - and: {(aA, 0), (aB, 1), (bA, 0), (bB,0)}
  - or: {(aA, 1), (aB, 1), (bA, 0), (bB,1)}
- Approximate the polynomial kernel, but much more efficient
- Guide by domain knowledge or heuristic search



2 trillion ads in one year

Data Scale of Ad-ctr

- Only 1 year search log produces 2 trillions examples
  - sub-sampling? not always works because of the personalization
- Feature size = #ngram + #users + #sessions + #combination
  - often at the same scale of #samples
- A training task some years ago

```
Sun Sep 23 [redacted] global # of instances [120,746,552,096]
Sun Sep 23 [redacted] global # of features  [60,801,353,282]
```

Industry Dataset Size

- 100 billions of samples
- 10 billions of features
- 1T—1P training data

Training data Google

>5 years ago

Product	Examples	Compressed Raw data	Training data	Compression	Features per example	bytes per feature
A	59.9B	9.98TB	2.00TB	4.99x	54.9	0.67
B	7.6B	2.67TB	0.71TB	3.78x	94.9	1.07
C	197.5B	66.66TB	15.54TB	4.29x	77.7	1.11
D	129.1B	61.93TB	17.24TB	3.59x	100.57	1.46

Where to store the data

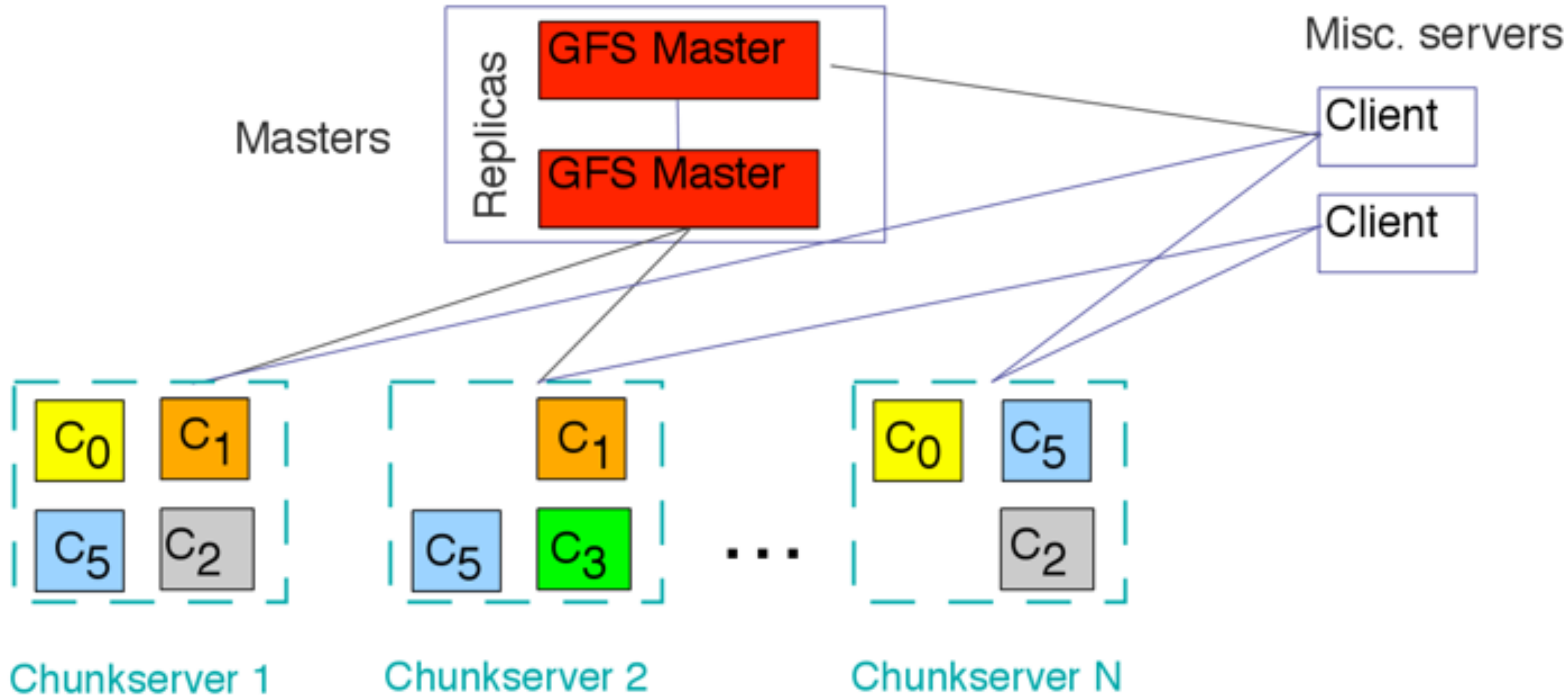
- Lots of disks
- Fail at any time

Access patterns

- Files are large 100MB—10GB
- Sequential read and append

```
1100958974 Jun 30 17:17 part-13073.gz
1102681642 Jun 30 17:18 part-13074.gz
1102654752 Jun 30 17:18 part-13075.gz
1099726070 Jun 30 17:18 part-13076.gz
1101590533 Jun 30 17:18 part-13077.gz
1100016199 Jun 30 17:19 part-13078.gz
1100637016 Jun 30 17:19 part-13079.gz
1102254166 Jun 30 17:20 part-13080.gz
1103309165 Jun 30 17:21 part-13081.gz
1101572961 Jun 30 17:21 part-13082.gz
1100632933 Jun 30 17:22 part-13083.gz
1101068910 Jun 30 17:22 part-13084.gz
1099448455 Jun 30 17:22 part-13085.gz
1100856891 Jun 30 17:23 part-13086.gz
1101677645 Jun 30 17:23 part-13087.gz
1102511398 Jun 30 17:23 part-13088.gz
1099531212 Jun 30 17:24 part-13089.gz
1100965668 Jun 30 17:24 part-13090.gz
1100877289 Jun 30 17:24 part-13091.gz
1100353449 Jun 30 17:25 part-13092.gz
1102797472 Jun 30 17:25 part-13093.gz
1100592927 Jun 30 17:25 part-13094.gz
1100954490 Jun 30 17:26 part-13095.gz
```

Google File System



- Data are replicated
  - write success only if all replicas are done
- Request data:
  - ask master for the location
  - ask chunk server for the data
- New generations: Colossus

HDFS

- Open source implementation of GFS
- operations:
  - haddop fs -ls, -get, -put, -head, -cat, …
  - libhdfs: C API
  - mount to local filesystem
- A little bit slower than GFS (personal experience)
- Large delay
  - hadoop fs -ls /xxx (8000 files)

```
real    0m16.992s
user    0m3.905s
sys     0m0.287s
```

- Sometimes reading the training data uses more times than training