Machine Learning Systems Design

Lecture 3: Data engineering

What's your best new president meme?

Dave Itzkoff @ditzkoff

Out on the town having the time of my life with a bunch of friends

11:09 AM · Jan 20, 2021

Stanford University

CS 329 | Chip Huyen

Logistics

- OHs started this week
- Assignment 1 out (due Wed, Jan 27 @ 11:59PM PT)
- Final project instruction out
    - Meet with course staff for brainstorming (sign-up sheet out this sun)
    - Group size?
- Lecture note?

Zoom poll: Do you find the lecture notes helpful?

1. Yes
2. Yes if shorter
3. No

Agenda

1. Mind vs. data
2. Data engineering 101
3. Breakout exercise
4. Batch processing vs. stream processing
5. Creating training datasets
6. Programmatic labeling
7. Weak supervision & more

1. Mind vs. data

WHO WOULD WIN?

Intelligent model architectures that took researchers their entire PhDs to design

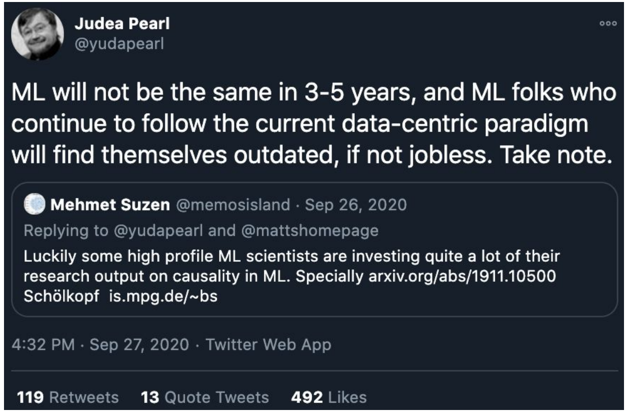
Terabytes of data scraped from Reddit in a week

Zoom poll!

Mind

"Data is profoundly dumb."

Judea Pearl, [Mind over data - The Book of Why]



"Huge computation and massive amount of data, … with simple learning device, … [create] incredibly bad learners. … Structure allows us to design systems that can learn more from less data."

Chris Manning, [Deep Learning and Innate Priors]

Data

"General methods that leverage computation are ultimately the most effective, and by a large margin … Human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation."

Richard Sutton, [Bitter Lesson]

"We don't have better algorithms. We just have more data."

Peter Norvig, [The Unreasonable Effectiveness of Data]

"Imposing structure requires us to make certain assumptions, which are invariably wrong for at least some portion of the data."

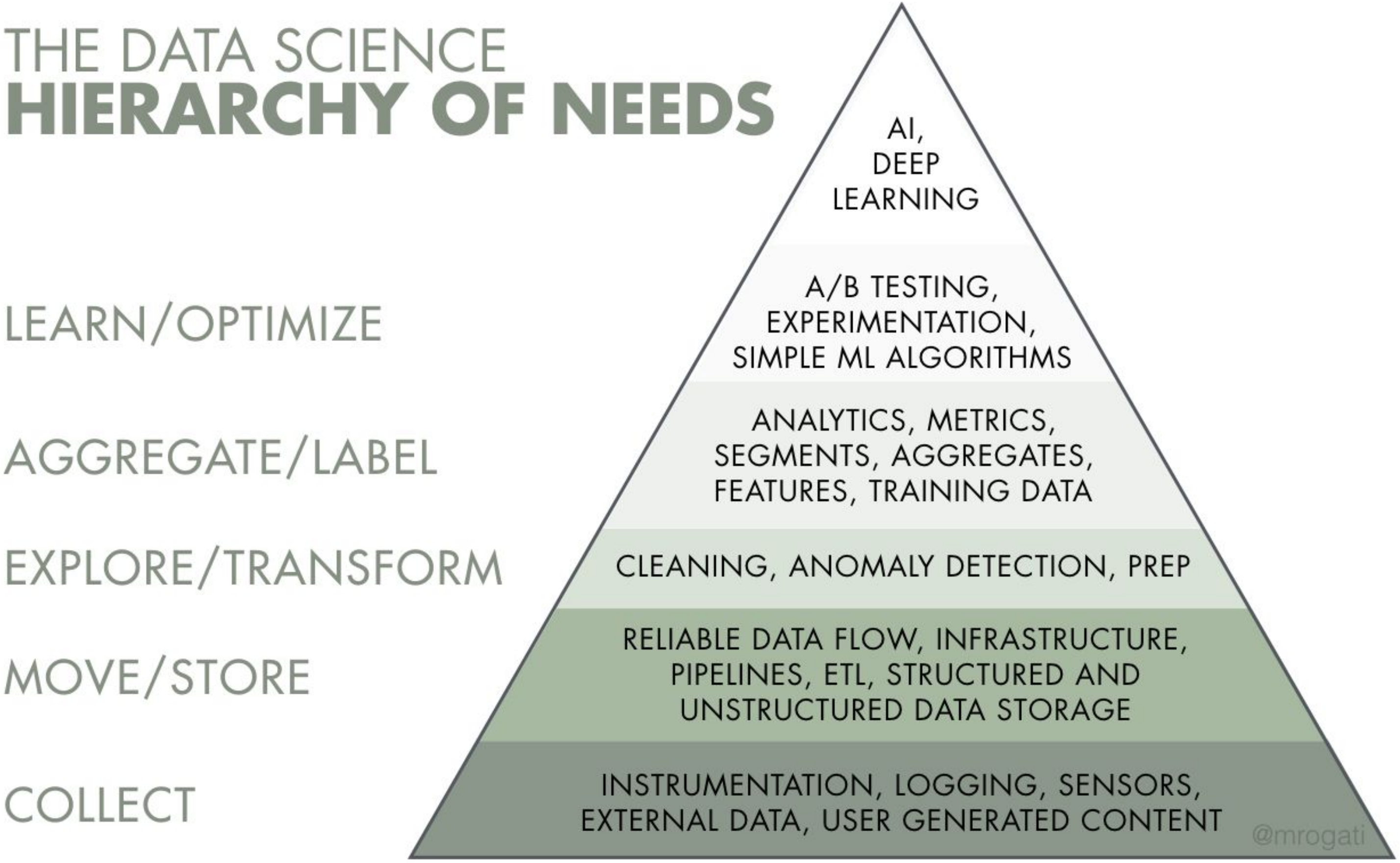
Yann LeCun, [Deep Learning and Innate Priors]

Data is necessary.

The debate is whether *finite** data is sufficient.

* If we had infinite data (and infinite memory), we can solve arbitrarily complex problems by just looking up the answers.

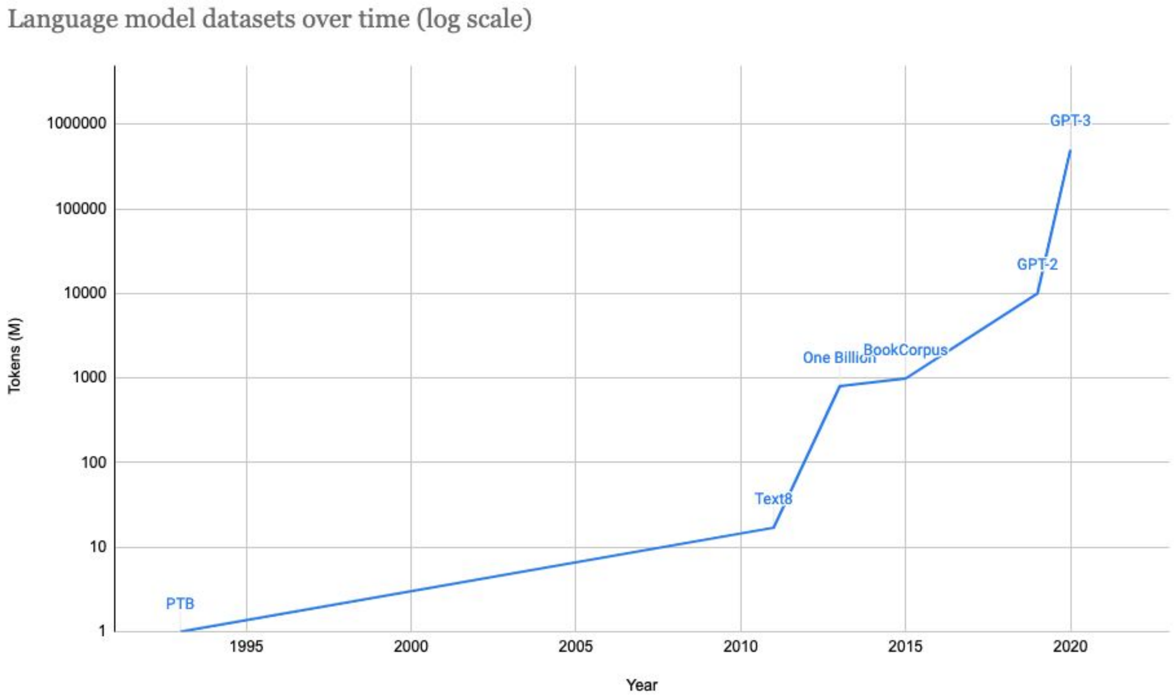
A lot of data $\neq$ infinite data.



The AI Hierarchy of Needs (Monica Rogati, 2017)

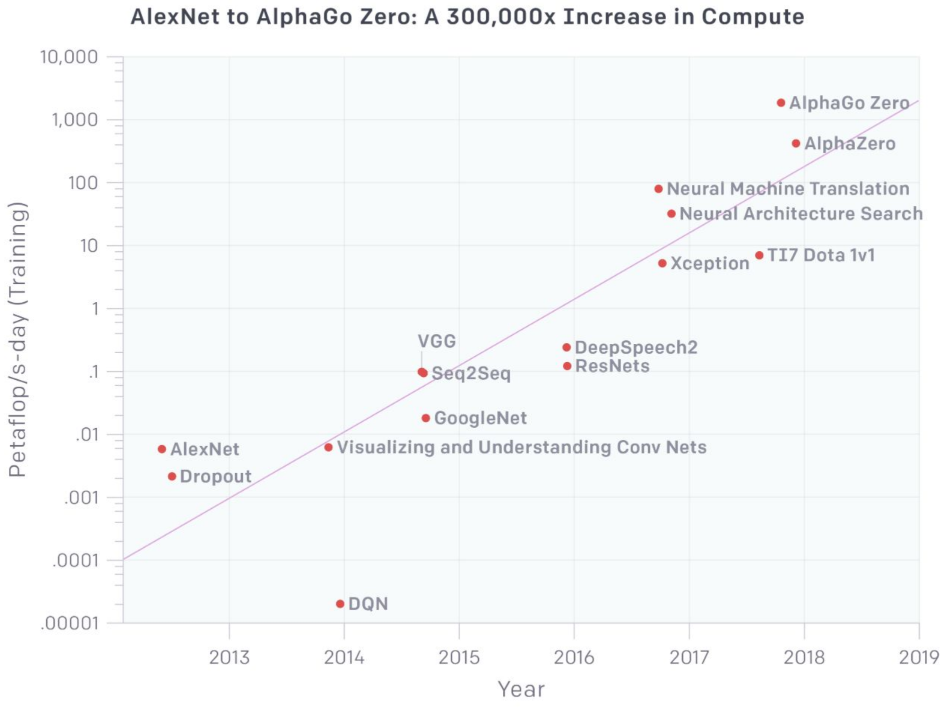
Datasets for language models

Dataset	Year	Tokens (M)
Penn Treebank	1993	1
Text8	2011	17
One Billion	2013	800
BookCorpus	2015	985
GPT-2 (OpenAI)	2019	10,000
GPT-3 (OpenAI)	2020	500,000



More data (generally) needs more compute

"amount of compute used in the largest AI training runs has doubled every 3.5 months"



AI and Compute (OpenAI 2018)

Let’s see what a language model trained on 500B tokens can do

2. Data engineering 101

Very basic. For details, take a database class!

Data basics: data sources

- User generated: inputs, clicks
- Systems generated: logs, metadata, predictions
- Enterprise applications data: inventory, customer relationships
- Third-party data

Third party data: creepy but fascinating

- Types of data
  - social media, income, job
- Demographic group
  - men, age 25-34, work in tech
- More available with Mobile Advertiser ID
- Useful for learning features
  - people who like A also like B

Remote working

Millions of people decided to #stayhome and work remotely to limit the spread of coronavirus. Use our Remote working segment to easily reach them and show software or products that will help them stay effective.

61 M profiles

How did we build the segment?

—

Our segment includes profiles of users who recently read articles, watched videos or used mobile apps which refers to:

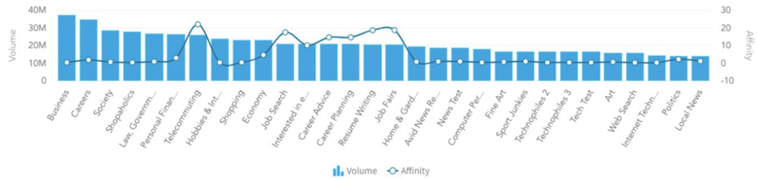
- remote working
- effective ways of working from home
- tools for remote workers
- homeschooling and e-learning

—

If you want to reach remote workers, try to extend your target group by selecting the top interests, which include Telecommuting, Career Planing or Personal Finance.

Data point affinity and volume

Data point intersection volume and affinity, sorted by volume.



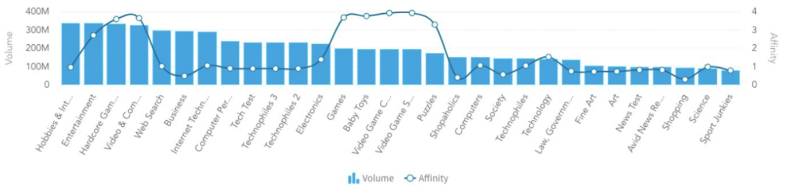
Top interests

—

They love computing and electronic entertainment. If you want to reach players, try targeting at their top interests.

Data point affinity and volume

Data point intersection volume and affinity, sorted by volume.



Data basics: formats to store

- How to store both data and labels?
  - `{‘image’: [[200,155,0], [255,255,255], ...], ‘label’: ‘car’, ‘id’: 1}`
- How to store a model?
- How to store any complex object?

Data basics: data serialization

- Converting a data structure or object state into a format that can be stored or transmitted and reconstructed later

	Format	**Binary/Text**	**Human-readable?**	**Example use cases**
	JSON	Text	Yes	Everywhere
Row-based	CSV	Text	Yes	Everywhere
Column-based	Parquet	Binary	No	Hadoop, Amazon Redshift
	Avro	Binary primary	No	Hadoop
	Protobuf	Binary primary	No	Google, TensorFlow (TFRecord)
	Pickle	Text, binary	No	Python, PyTorch serialization

Data basics: column-based vs. row-based

Column-based:

- stored and retrieved column-by-column
- good for accessing features

Row-based:

- stored and retrieved row-by-row
- good for accessing samples

	Column 1	Column 2	Column 3
Sample 1	…	…	…
Sample 2	…	…	…
Sample 3	…	…	…

Data basics: text vs. binary files

Benefits of column-based:

flexible data access: can access only columns required (e.g. if your data has 1000 columns and you only want 5 columns, load only 5 columns)

Column-based:

- stored and retrieved column-by-column
- good for accessing features

	Column 1	Column 2	Column 3
Sample 1	...	...	...
Sample 2	...	...	...
Sample 3	...	...	...

You can unload the result of an Amazon Redshift query to your Amazon S3 data lake in Apache Parquet, an efficient open columnar storage format for analytics. Parquet format is up to 2x faster to unload and consumes up to 6x less storage in Amazon S3, compared with text formats. This enables you to save data transformation and enrichment you have done in

amazon REDSHIFT

Data basics: column-based vs. row-based

Pandas DataFrame: column-based

- accessing a row much slower than accessing a column and NumPy

```
# Get the column `date`, 1000 loops
%timeit -n1000 df["Date"]

# Get the first row, 1000 loops
%timeit -n1000 df.iloc[0]
```

```
1.78 µs ± 167 ns per loop (mean ± std. dev. of 7 runs, 1000 loops each)
145 µs ± 9.41 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

NumPy ndarray: row-based by default

- can specify to be column-based

```
df_np = df.to_numpy()
%timeit -n1000 df_np[0]
%timeit -n1000 df_np[:,0]
```

```
147 ns ± 1.54 ns per loop (mean ± std. dev. of 7 runs, 1000 loops each)
204 ns ± 0.678 ns per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```


https://github.com/chiphuyen/just-pandas-things

Data basics: text vs. binary files

	Text files	Binary files
Examples	CSV, JSON	Parquet
Pros	Human readable	Compact
To store the number 1000000?	7 characters -> 7 bytes	If stored as int32, only 4 bytes

You can unload the result of an Amazon Redshift query to your Amazon S3 data lake in Apache Parquet, an efficient open columnar storage format for analytics. Parquet format is up to 2x faster to unload and consumes up to 6x less storage in Amazon S3, compared with text formats. This enables you to save data transformation and enrichment you have done in

Data basics: column-based vs. row-based

Column-based:

- stored and retrieved column-by-column
- good for accessing features
- good for using data for analytic tasks

	Column 1	Column 2	Column 3
Sample 1	...	...	...
Sample 2	...	...	...
Sample 3	...	...	...

Row-based:

- stored and retrieved row-by-row
- good for accessing samples
- good for managing transactions as they come in

Data basics: OLTP vs. OLAP

OnLine Transaction Processing

OnLine Analytical Processing

OLTP: OnLine Transaction Processing

- How to handle a large number of small transactions?
  - e.g. ordering food, ordering rides, buying things online, transferring money
- Requirements:
  - Atomicity: all the steps in a transaction fail or succeed as a group
    - If payment fails, don't assign a driver
  - Isolation: concurrent transactions happen as if sequential
    - Don't assign the same driver to two different requests that happen at the same time
  - Fast response time (e.g. milliseconds)
- Operations:
  - INSERT, UPDATE, DELETE

See ACID: Atomicity, Consistency, Isolation, Durability

Row

```
INSERT INTO RideTable(RideID, Username, DriverID, City, Month, Price)
VALUES ('10', 'memelord', '3932839', 'Stanford', 'July', '20.4');
```

OLAP: OnLine Analytical Processing

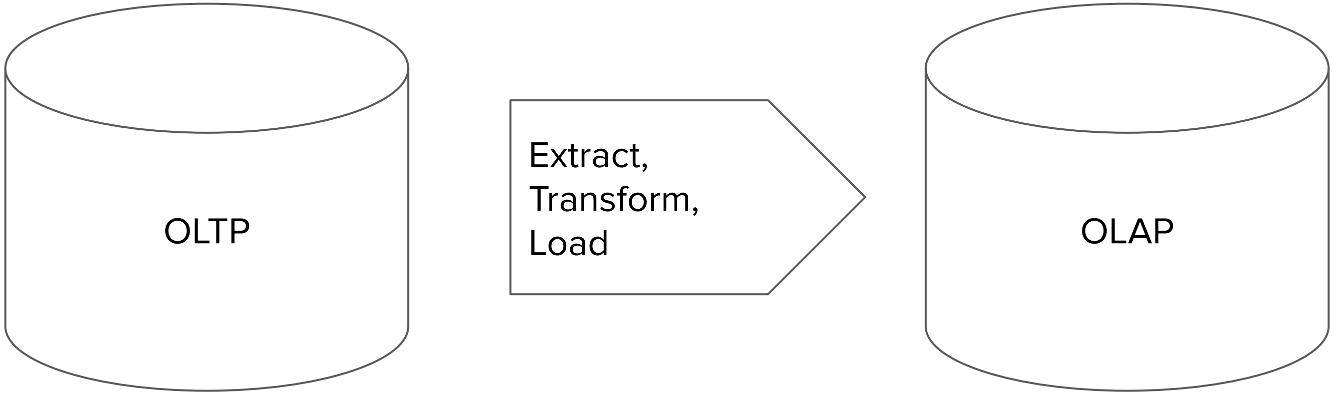
- How to get aggregated information from a large amount of data?
  - e.g. what’s the average ride price last month for riders at Stanford?
- Requirements:
  - Can handle complex queries on large volumes of data
  - Okay response time (seconds, minutes, even hours)
- Operations:
  - Mostly SELECT

Column

```
SELECT AVG(Price)
FROM RideTable
WHERE City = 'Stanford' AND Month = 'July';
```

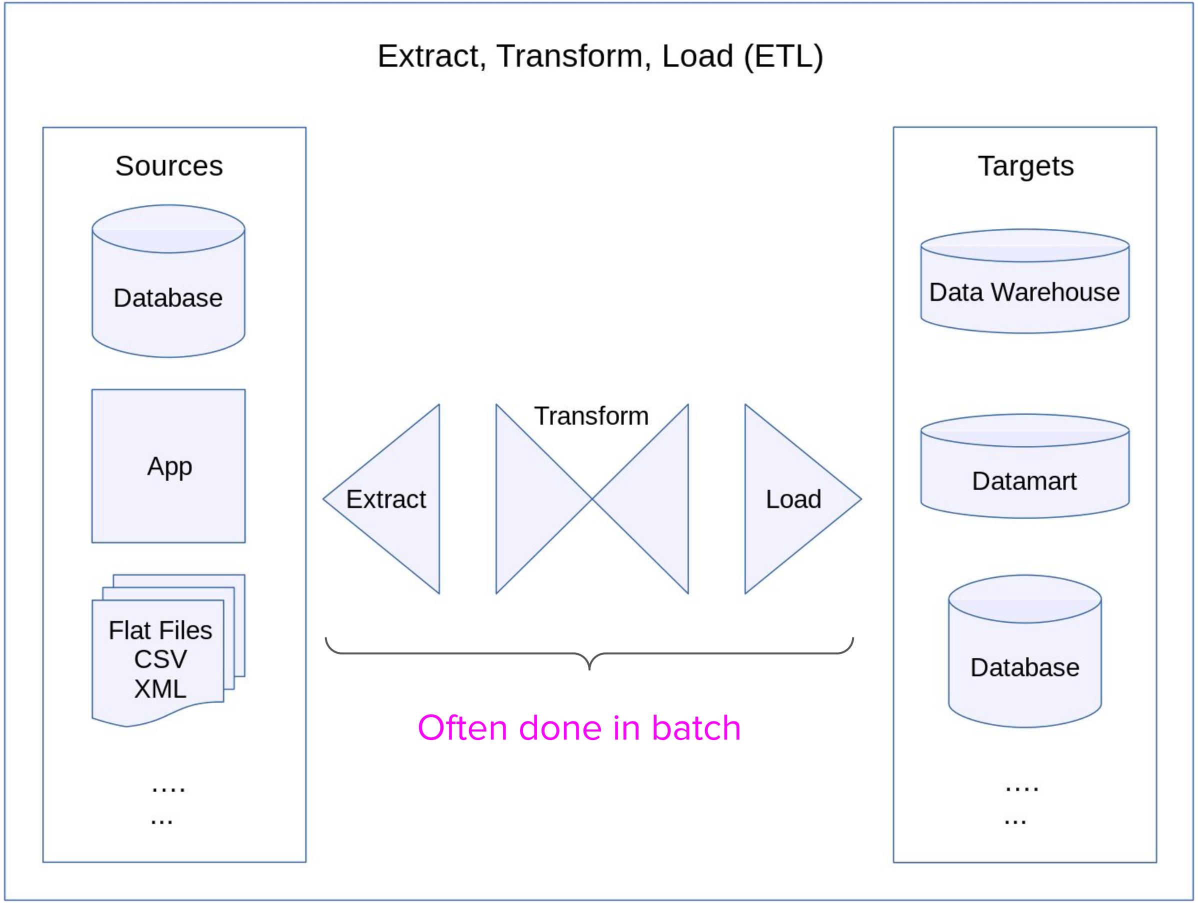
Data basics: ETL

Data basics: ETL (Extract, Transform, Load)



Transform: the meaty part

- cleaning, validating, transposing, deriving values, joining from multiple sources, deduplicating, splitting, aggregating, etc.



https://commons.wikimedia.org/wiki/File:Extract,_Transform,_Load_Data_Flow_Diagram.svg

Data basics: structured vs. unstructured data

Structured	**Unstructured**
Schema clearly defined	Whatever
Easy to search and analyze	Fast arrival (e.g. no need to clean up first)
Can only handle data with specific schema	Can handle data from any source
Schema changes will cause a lot of trouble	No need to worry about schema changes
Data warehouse	Data lake

Data basics: structured vs. unstructured data

Structured	**Unstructured**
Schema clearly defined	Whatever
Easy to search and analyze	Fast arrival (e.g. no need to clean up first)
Can only handle data with specific schema	Can handle data from any source
Schema changes will cause a lot of trouble	No need to worry about schema changes
Data warehouse	Data lake

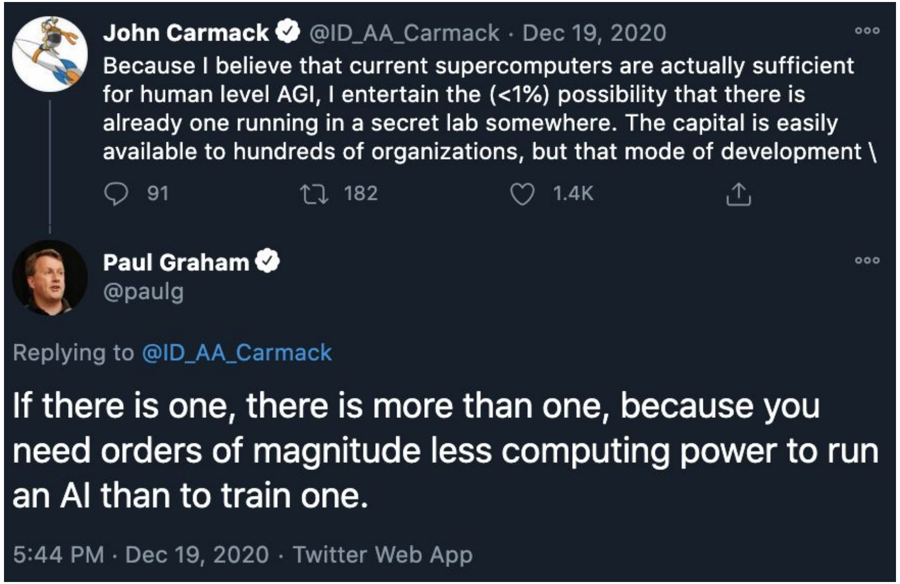
Structured -> $\underbrace{\text{unstructured}}_{\text{want more flexibility}}$ -> $\underbrace{\text{structured}}_{\text{tools \& infra standardized}}$

ETL -> ELT -> ETL

3. Breakout exercise

Breakout exercise (group of 5, 5 mins)

- Where do you stand on mind vs. data debate?
- Would data be sufficient or would we need a breakthrough in ML to achieve human-level AI?
- Is human-level AI even possible?
- Is AGI possible?



4. Batch processing vs. online processing

Online predictions: solution

1. Fast inference
    a. model that can make predictions in the order of milliseconds

will cover in a later lecture!

2. Real-time pipeline
    a. a pipeline that can process data, input it into model, and return a prediction in real-time

Real time pipeline: ride-sharing example

To detect whether a transaction is fraud, need features from:

- this transaction
- user’s recent transactions (e.g. 7 days)
- credit card’s recent transactions
- recent in-app frauds
- etc.

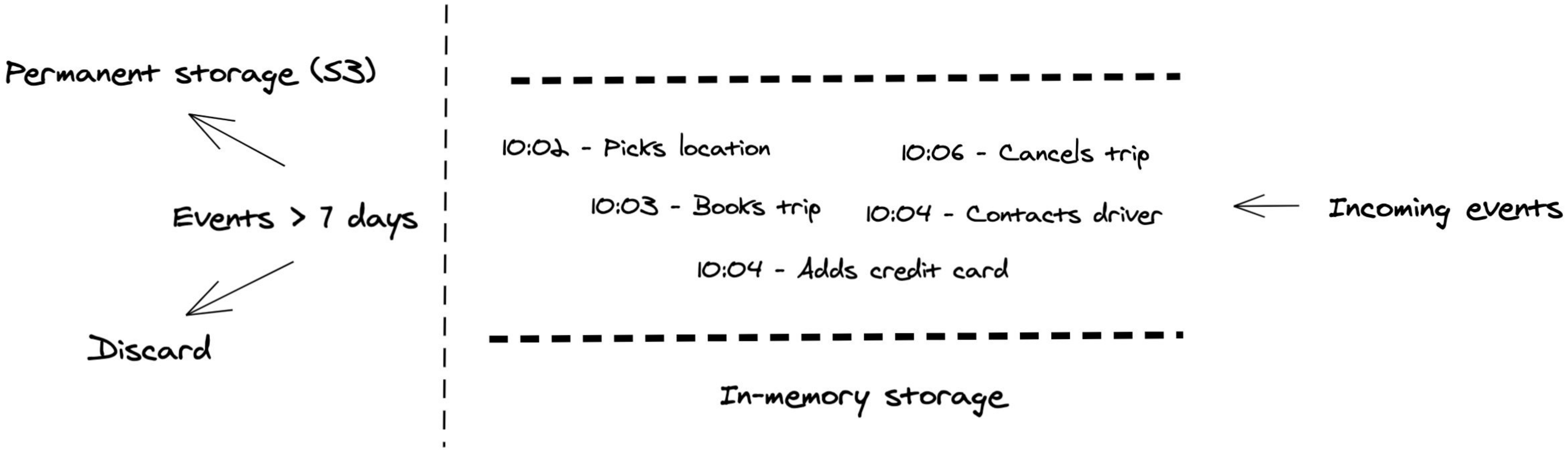
Real time pipeline: ride-sharing example

To detect whether a transaction is fraud, need features from:

- this transaction
- user's recent transactions (e.g. 7 days)
- credit card's recent transactions
- recent in-app frauds
- etc.

How to quickly access these features?

Stream storage



Stream storage

972 companies reportedly use **Kafka** in their tech stacks, including **Uber, Spotify,** and **Shopify**.

Uber | Spotify | Shopify | Slack | Robinhood | LaunchDarkly | Nubank | The New York Times | Alibaba Travels

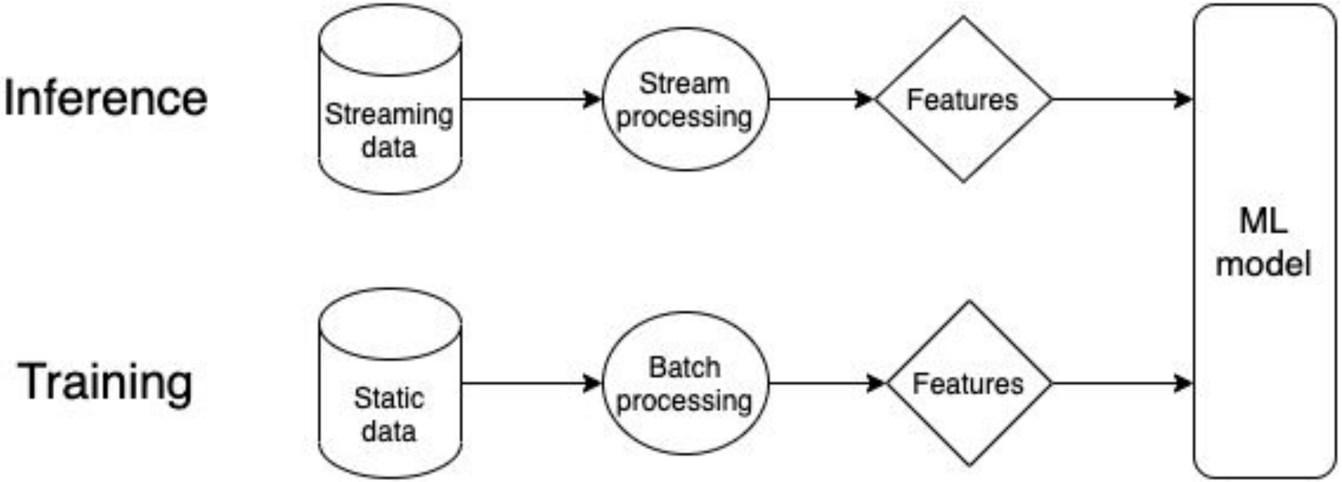
233 companies reportedly use **Amazon Kinesis** in their tech stacks, including **Amazon, Instacart,** and **Lyft**.

Amazon | Instacart | Lyft | LaunchDarkly | Accenture | Figma | trivago | N26 | Pratilipi

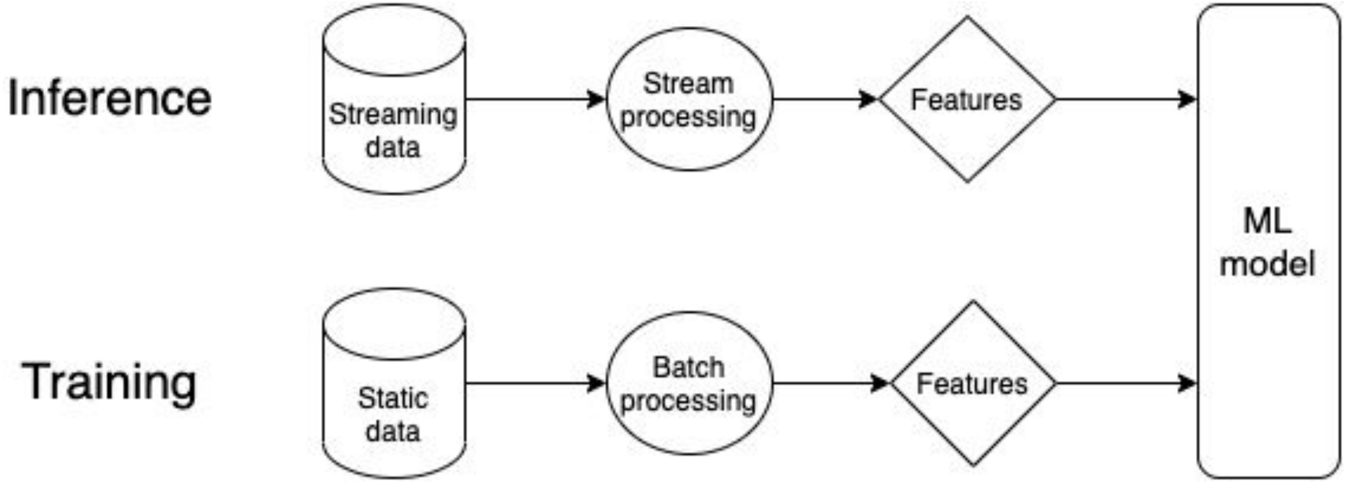
Need static + dynamic features

Static data	Streaming data
CSV, PARQUET, etc.	Kafka, Kinesis, etc.
Bounded: know when a job finishes	Unbounded: never finish
Static features: ● age, gender, job, city, income ● when account was created ● rating	Dynamic features ● locations in the last 10 minutes ● recent activities
Can be processed in batch ● e.g. SQL, MapReduce	Processed as events arrive ● e.g. Apache Flink, Samza

One model, two pipelines

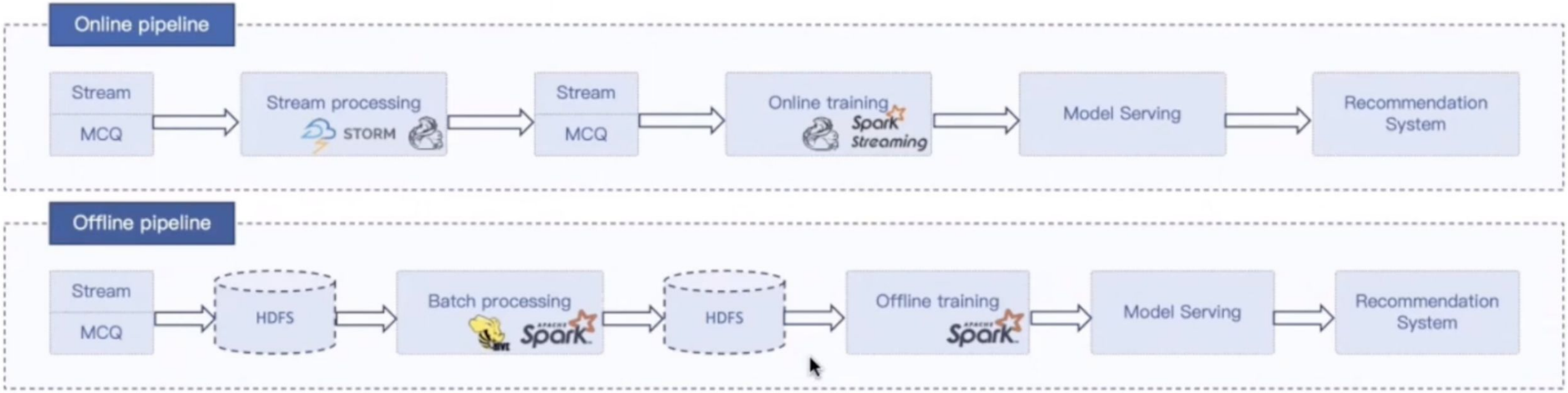


One model, two pipelines



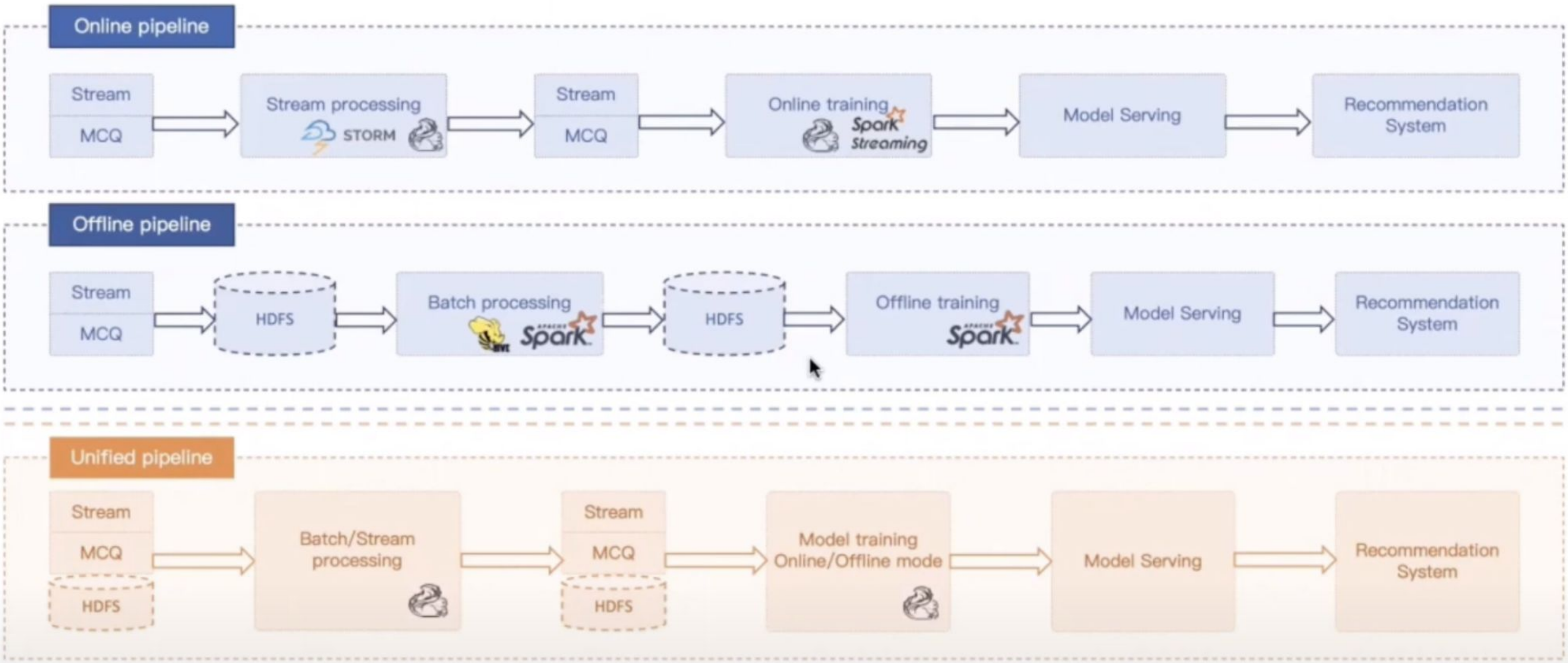
⚠️⚠️ A common source of errors in production ⚠️⚠️

One model, two pipelines: example



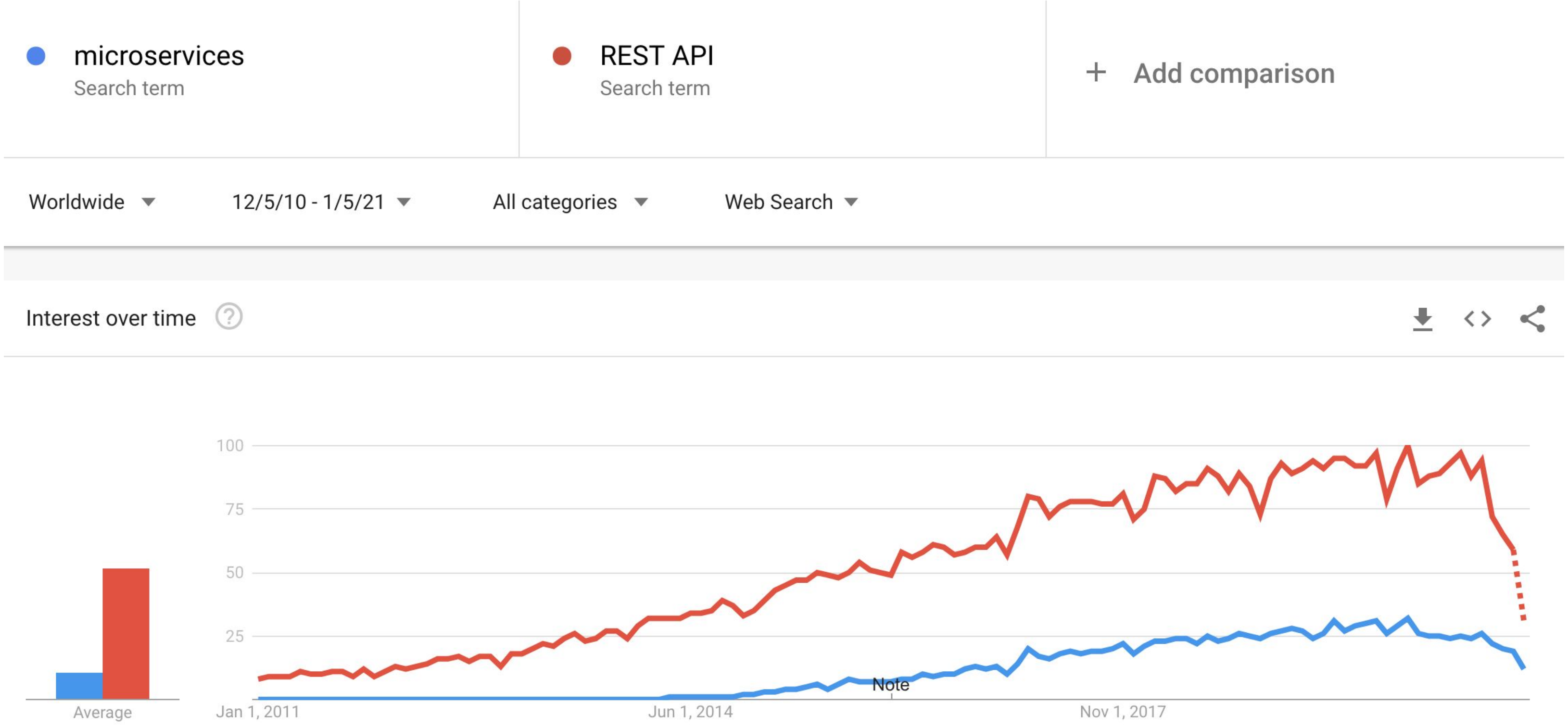
Machine learning with Flink in Weibo (Qian Yu, QCon 2019)

Apply unified Flink APIs to both online and offline ML pipelines



[Machine learning with Flink in Weibo](#) (Qian Yu, QCon 2019)

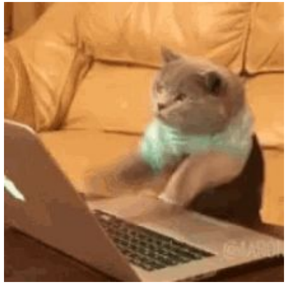
Microservices ft. REST APIs



REST APIs: request-driven

Request
POST, GET, etc.

Response

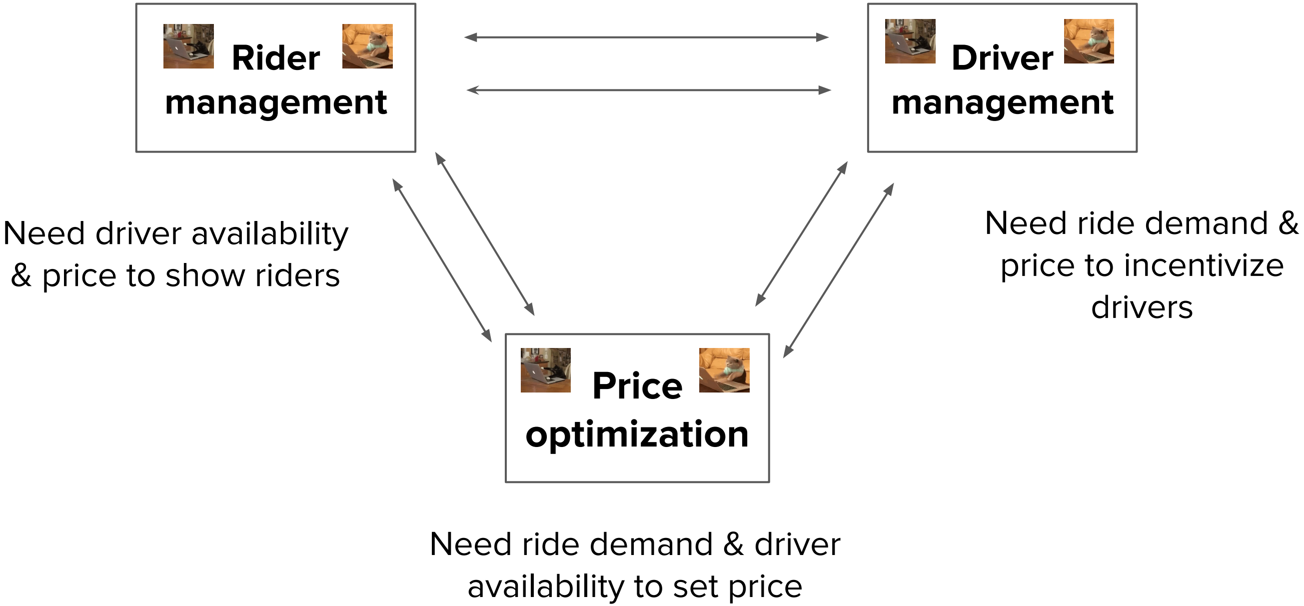


Client

Server

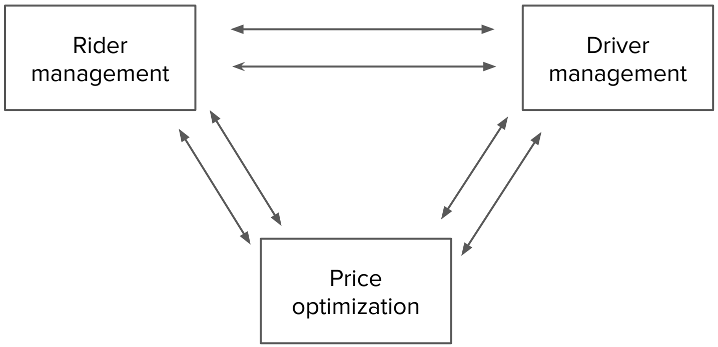
Server has to listen for the request to register

Inter-service communication



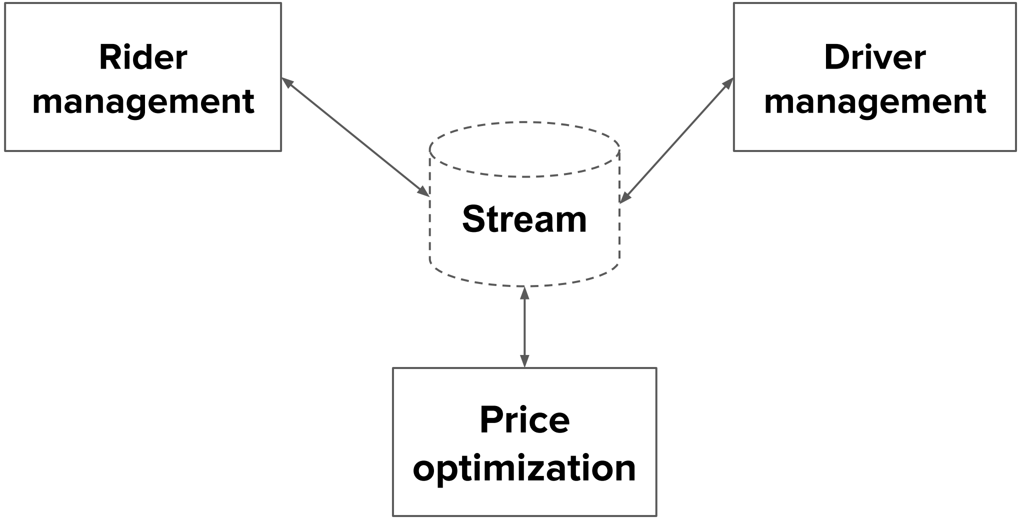
Request-driven: problems

- How to map data transformations through the entire system?
- How to debug?



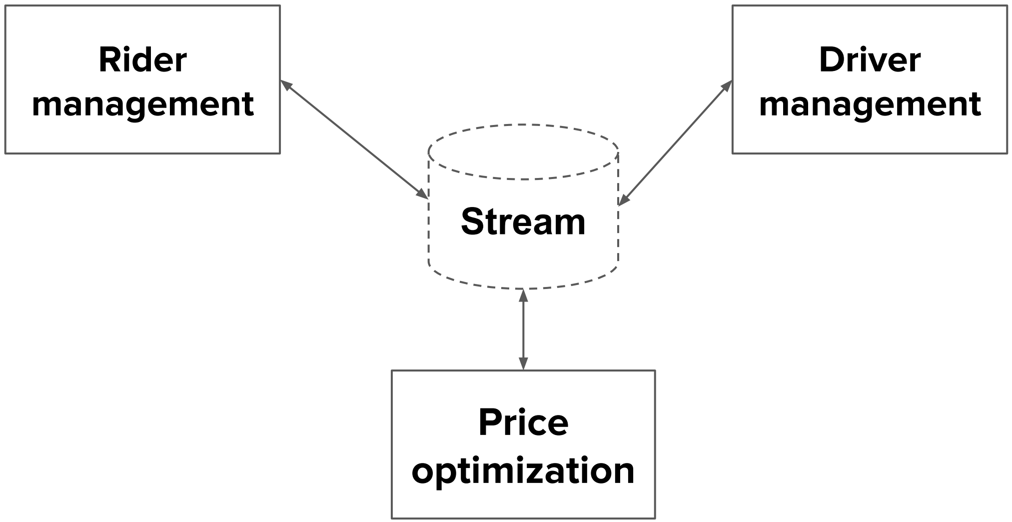
Event-driven: publish/subscribe

- All services publish to a stream
- All services subscribe to this stream to get info they need



Event-driven: pubsub*

- Data flows through this stream, can monitor data transformations through the entire system



There are other streaming models, e.g. message queue

Barriers to stream processing

1. Companies don’t see the benefits of streaming
    - Systems not at scale
    - Batch predictions work fine
    - Online predictions would work better but they don’t know that

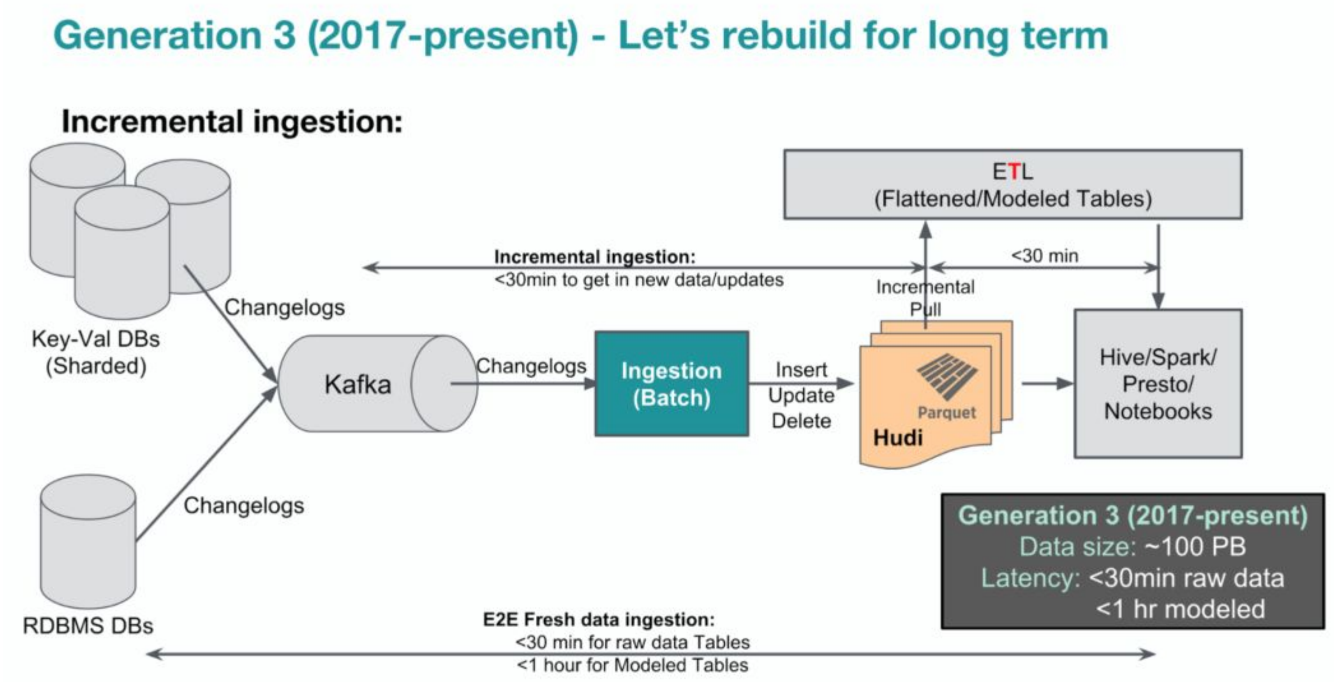
Barriers to stream processing

1. Companies don’t see the benefits of streaming
2. High initial investment on infrastructure
3. Mental shift
4. Python incompatibility

Data pipeline: case study with Uber

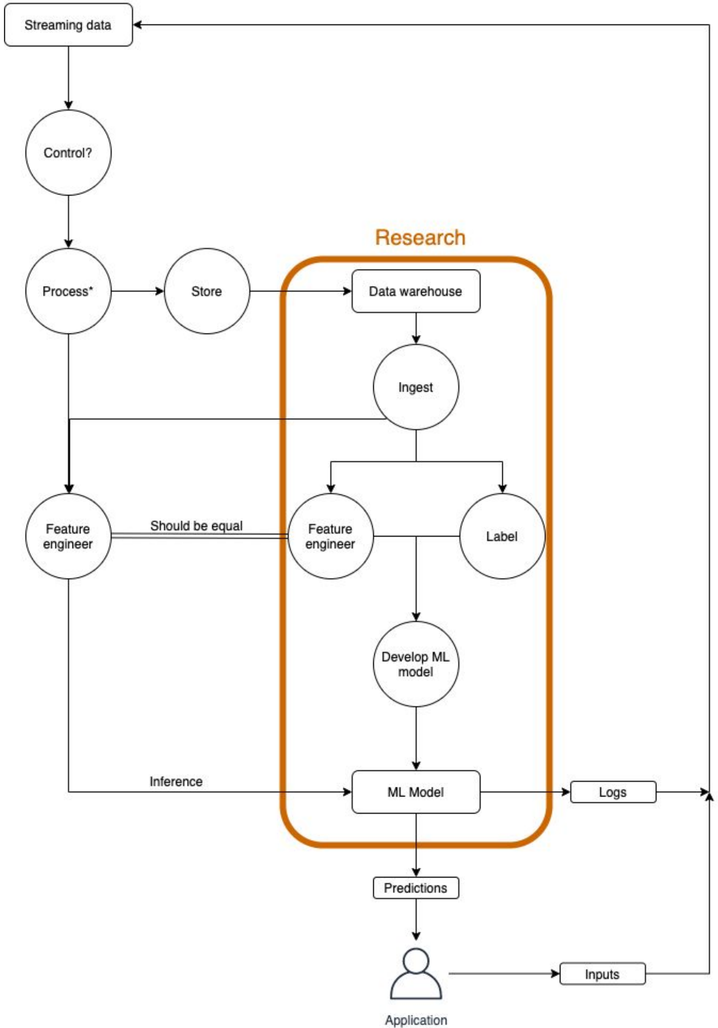
e.g.
JSON blobs

e.g.
transactions



[Uber's Big Data Platform: 100+ Petabytes with Minute Latency]() (Reza Shiftehfar, Uber Engineering Blog 2018)

Data pipeline for online ML systems



5. Creating training datasets

⚠️ Data: full of potential for biases ⚠️

- sampling/selection biases
- under/over-representation of subgroups
- human biases embedded in historical data
- labeling biases
- …

Algorithmic biases not covered (yet)!

Partition: shuffle then split

	Week 1	Week 2	Week 3	Week 4	Week 5
Test split	X11	X21	X31	X41	X51
Valid split	X12	X22	X32	X42	X52
Train split	X13	X23	X33	X43	X53
	X14	X24	X34	X44	X54
	...	...	...	...	...

Aim for similar distributions of labels across splits
e.g. each split has 90% NEGATIVE, 10% POSITIVE

Partition: shuffle then split

	Week 1	ek 2	Weel	Week 4	Week 5
Test split	X11			X41	X51
Valid split	X12	X	2	X42	X52
Train split	X13			X43	X53
	X14	24	X3	X44	X54
	...	...	...	...	...

⚠️ **Not representative of real-world usage!** ⚠️

A better partition

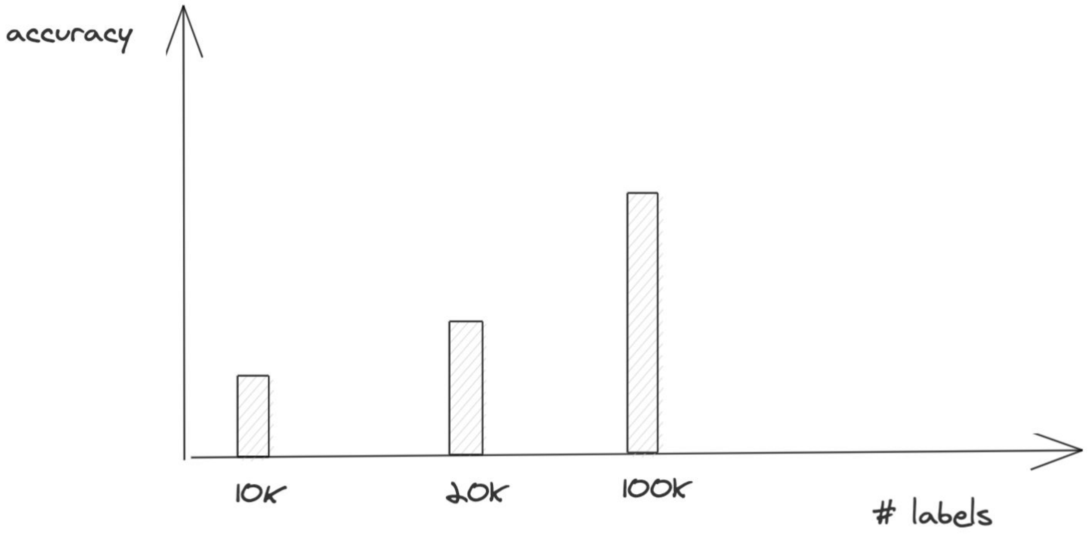
Train split

Week 1	Week 2	Week 3	Week 4	Week 5	
X11	X21	X31	X41	X51	**Valid split**
X12	X22	X32	X42	X52	
X13	X23	X33	X43	X53	**Test split**
X14	X24	X34	X44	X54	
...	...	...	...	...	

Even better partition

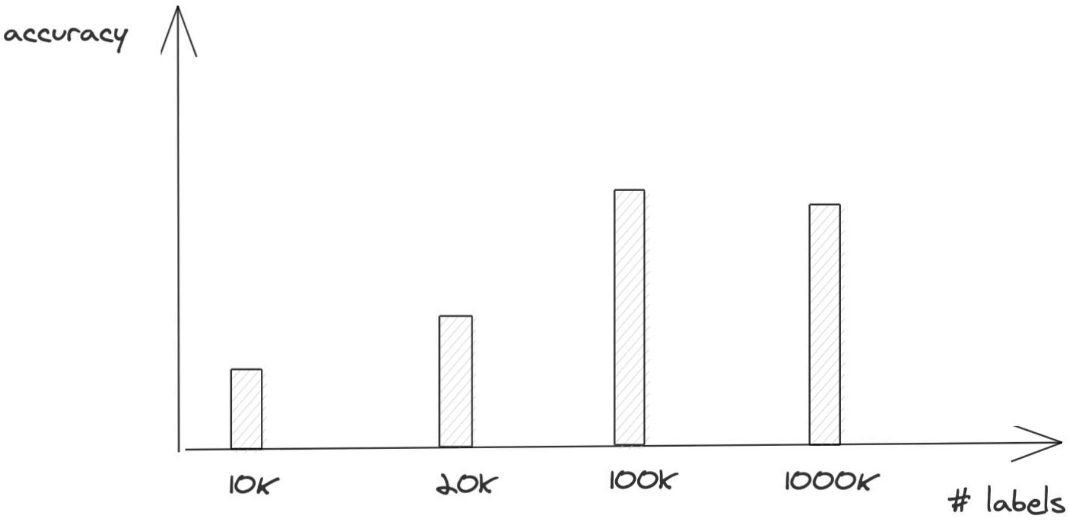
- Multiple train splits
  - Train on data from 6 months, 1 month, 1 week ago and validate on today data to see how worse performance gets over time
- Multiple valid/test splits
  - Evaluate performance on different slices of data
    - Different subgroups
    - Critical slices
      - e.g. object detection for self-driving cars: accuracy on road surfaces with cyclists is more important

⚠️ More data isn't always better ⚠️



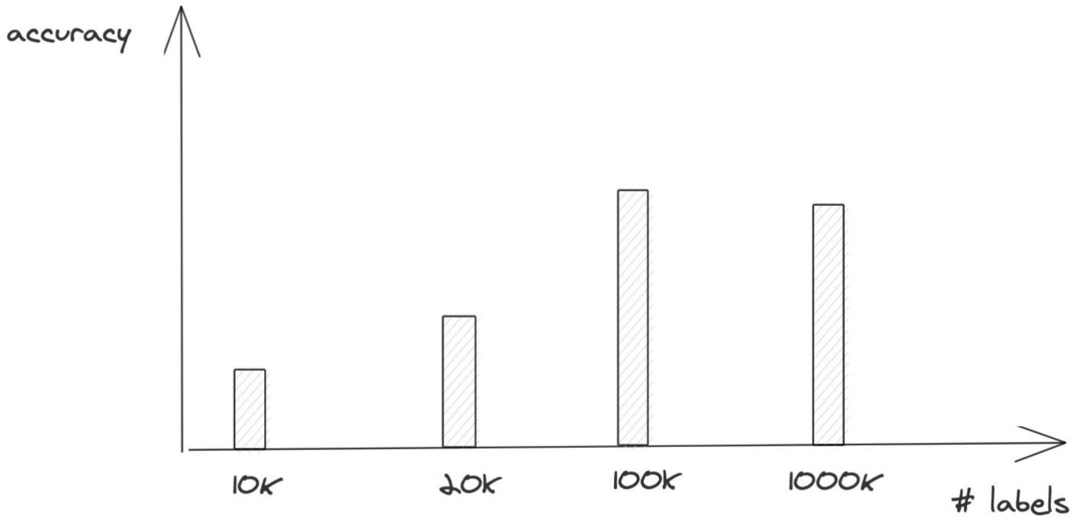
🧠 Idea 🧠: crowdsource data to get 1 million labels!

⚠️ More data isn't always better ⚠️



Why is the model getting worse?

⚠️ Label sources with varying accuracy ⚠️



- 100K labels: internally labeled, high accuracy
- 1M labels: crowdsourced, noisy

Label multiplicity: example

Task: label all entities in the following sentence:

Darth Sidious, known simply as the Emperor, was a Dark Lord of the Sith
who reigned over the galaxy as Galactic Emperor of the First Galactic Empire.

Label multiplicity: example

Zoom poll: which annotator is correct?

Task: label all entities in the following sentence:

Darth Sidious, known simply as the Emperor, was a Dark Lord of the Sith
who reigned over the galaxy as Galactic Emperor of the First Galactic Empire.

Annotator	# entities	Annotation
1	3	[**Darth Sidious**], known simply as the Emperor, was a [**Dark Lord of the Sith**] who reigned over the galaxy as [**Galactic Emperor of the First Galactic Empire**]
2	6	[**Darth Sidious**], known simply as the [**Emperor]**, was a **[Dark Lord**] of the [**Sith**] who reigned over the galaxy as [**Galactic Emperor**] of the [**First Galactic Empire**].
3	4	[**Darth Sidious**], known simply as the [**Emperor]**, was a **[Dark Lord of the Sith**] who reigned over the galaxy as [**Galactic Emperor of the First Galactic Empire**].

Label multiplicity

More expertise required (more difficult to label), more room for disagreement!

If experts can’t agree on a label, time to rethink human-level performance

Label multiplicity: solution

- Clear problem definition
- Annotation training
- Data lineage: track where data/labels come from

Label multiplicity: solution

- Clear problem definition
- Annotation training
- Data lineage: track where data/labels come from
- Learning methods with noisy labels
  - Learning with Noisy Labels (Natarajan et al., 2013)
  - Loss factorization, weakly supervised learning and label noise robustness (Patrini et al., 2016)
  - Cost-Sensitive Learning with Noisy Labels (Natarajan et al., 2018)
  - Confident Learning: Estimating Uncertainty in Dataset Labels (Northcutt et al., 2019)

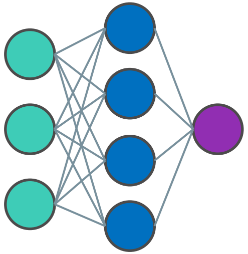
6. Programmatic labeling

Training data is the bottleneck

Data

+

Algorithms



=

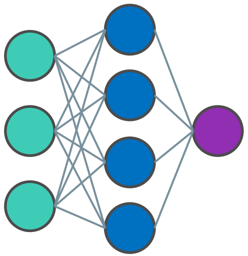
ML Model

Training data is the bottleneck

Data

Key differentiator

Algorithms



```
from transformers \
  import BertModel as model
```

Increasingly commoditized

ML Model

“We don’t have better algorithms. We just have more data.”

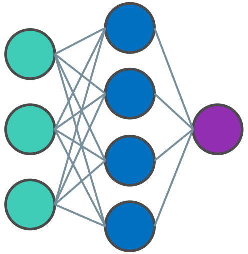
Peter Norvig, [The Unreasonable Effectiveness of Data]

Training data is the bottleneck

Data

- 8 Person-months
- 8-9 pt. differences

Algorithms



- 1-2 days
- <1 pt. differences

ML Model

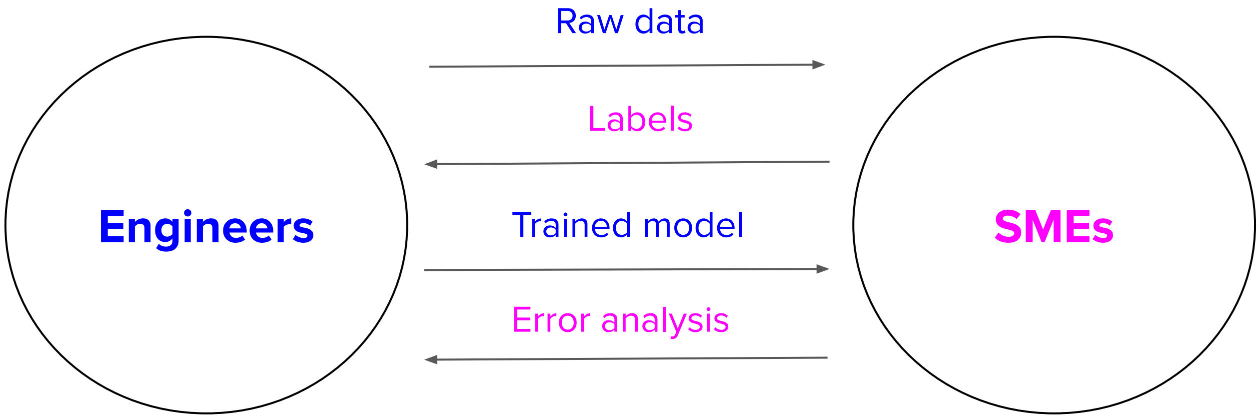
How to get training data in days?

[Cross-Modal Data Programming Enables Rapid Medical Machine Learning](#) (Dunnmon et al., 2019)

Hand labeling data is ...

- **Expensive:** Esp. when **subject matter expertise** required
- **Non-private:** Need to ship data to human annotators
- **Slow:** Time required scales linearly with # labels needed
- **Non-adaptive:** Every change requires re-labeling the dataset

Cross-functional communication



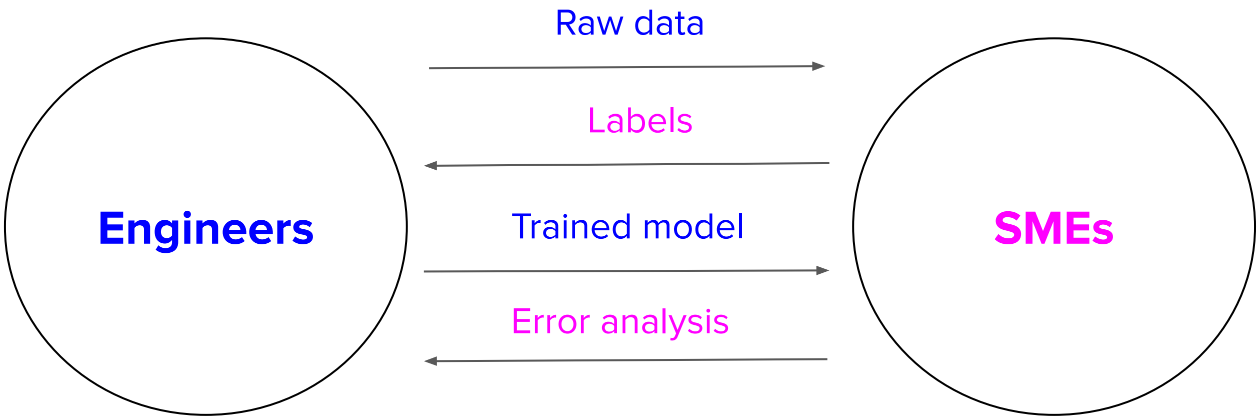

```
def function:
   if X:
      do Y
```

```
If the nurse's note mentions
serious conditions like pneumonia,
the patient's case should be given
priority consideration.
```

Code: version control, reuse, share

How to version, share, reuse **expertise**?

SME as labeling functions




```
def function:
    if X:
        do Y
```

```
def labeling_function(note):
    if "pneumonia" in note:
        return "EMERGENT"
```

Labeling functions (LFs): Encode SME heuristics as functions and use them to label training data *programmatically*

LFs: can express many different types of heuristics

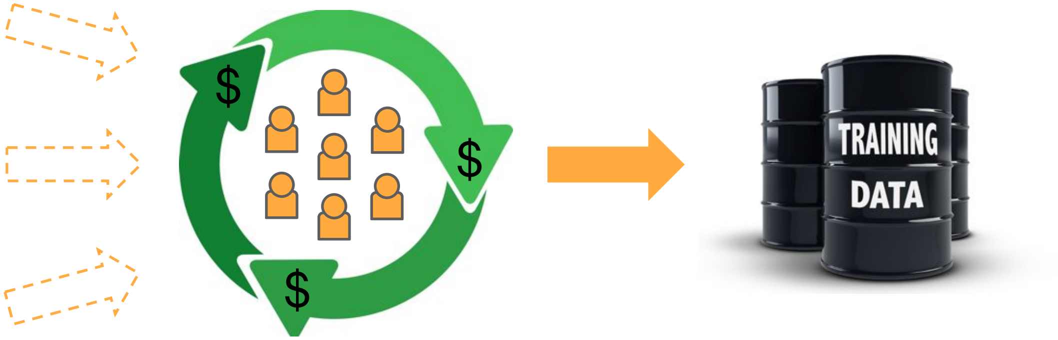
(.*)	Pattern Matching	If a phrase like “send money” is in email
	Boolean Search	If unknown_sender AND foreign_source
	DB Lookup	If sender is in our Blacklist.db
	Heuristics	If SpellChecker finds 3+ spelling errors
	Legacy System	If LegacySystem votes spam
	Third Party Model	If BERT labels an entity “diet”
	Crowd Labels	If Worker #23 votes spam

LFs: can express many different types of heuristics

"If nurse's report says 'malignant', likely to be emergent"

"If it matches a list of patient names…"

"If our legacy model thinks it's emergent…"



Labeling functions: Simple, flexible, interpretable, adaptable, fast

LFs: powerful but noisy

```
def LF_contains_money(x):
    if “money” in x.body.text:
        return “SPAM”
```

```
def LF_from_grandma(x):
    if x.sender.name is “Grandma”:
        return “HAM”
```

```
def LF_contains_money(x):
    if “free money” in x.body.text:
        return “SPAM”
```

From: **Grandma**

“Dear handsome grandson,
Since you can’t be home for Thanksgiving dinner this year, I’m sending you some **money** so you could enjoy a nice meal …”

??

“You have been pre-approved for free **cash** …”

??

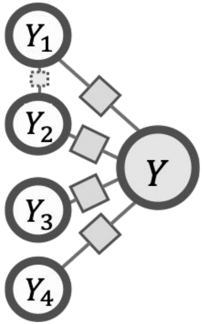
- **Noisy**: Unknown, inaccurate
- **Overlapping**: LFs may be correlated
- **Conflicting**: different LFs give different labels
- **Narrow**: Don’t generalize well

LF labels are combined to generate ground truths

```
def LF_contains_money(x):
   if “money” in x.body.text:
      return “SPAM”
```

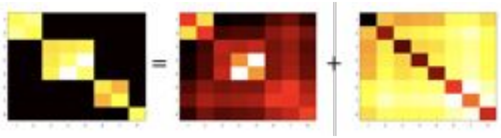
```
def LF_from_grandma(x):
   if x.sender.name is “Grandma”:
      return “HAM”
```

```
def LF_contains_money(x):
   if “free money” in x.body.text:
      return “SPAM”
```



[Intuition]

Look at agreements & disagreements



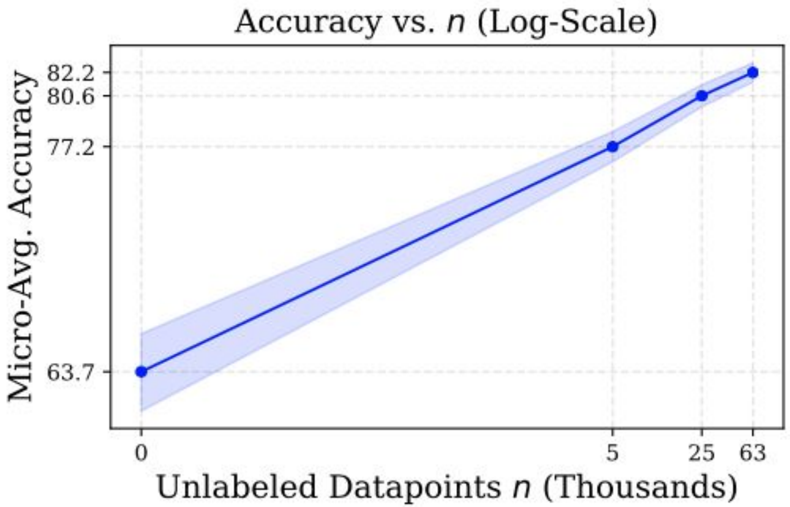
$$(\Sigma^{-1})_O = \Sigma_O^{-1} + zz^T$$

Provably consistent matrix completion-style algorithm over inverse covariance

[Ratner et. al. NeurIPS’16;
Bach et. al. ICML’17;
Ratner et. al. AAAI’19;
Varma et. al. ICML’19l;
Sala et. al. NeurIPS’19;
Fu et. al. ICML’20]

Hand labeling	Programmatic labeling
Expensive: esp. when subject matter expertise required	**Cost saving**: Expertise can be versioned, shared, reused across organization
Non-private: Need to ship data to human annotators	**Privacy**: Create LFs using a cleared data subsample then apply LFs to other data without looking at individual samples.
Slow: Time required scales linearly with # labels needed	**Fast**: Easily scale 1K -> 1M samples
Non-adaptive: Every change requires re-labeling the dataset	**Adaptive**: When changes happen, just reapply LFs!

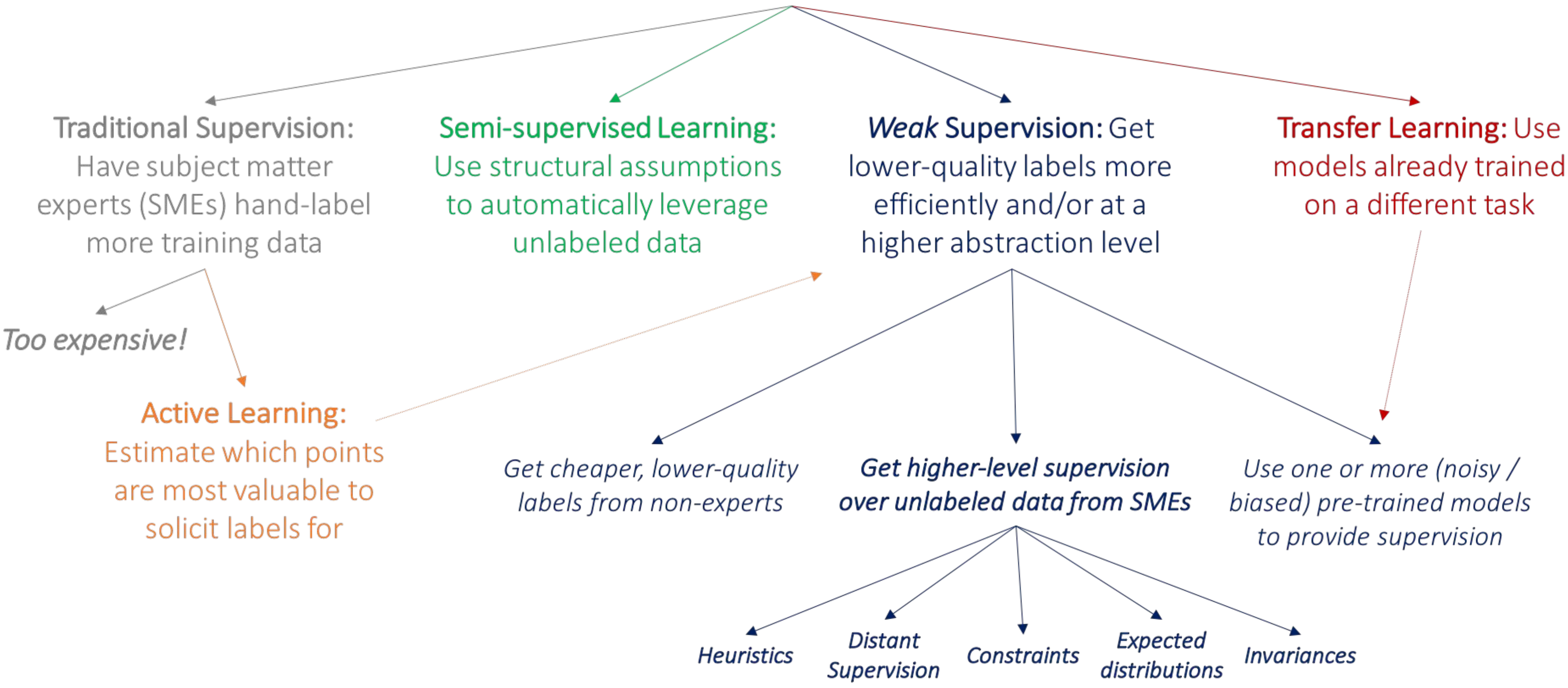
Programmatic labeling: Scale with unlabeled data



Ratner et. al. NeurIPS'16; Ratner et. al. VLDB'18; Ratner et. al. AAAI'19

7. Weak supervision & more

How to get more labeled training data?



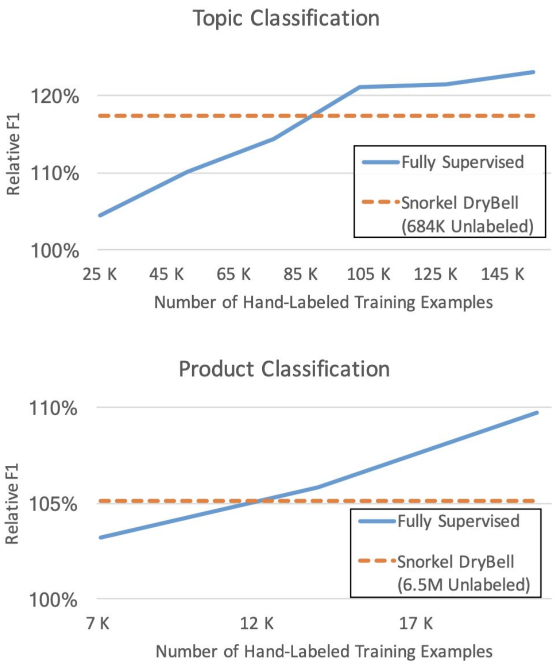
[Weak Supervision: A New Programming Paradigm for Machine Learning](#) (Ratner et al., 2019)

Semi-supervised

- Use structural assumptions to leverage a large amount of unlabeled data together with a small amount of labeled data
    - Hashtags in the same profile/tweet are probably of similar topics

Weakly-supervised

- Leverage noisy, imprecise sources to create labels
  - e.g. if “money” is in an email it’s probably spam



Harnessing Organizational Knowledge for Machine Learning (Ratner et al., Google AI Blog 2019)

Active learning

- Label only samples that are estimated to be most valuable to the model
  - e.g. label only CT scans close to the decision boundary

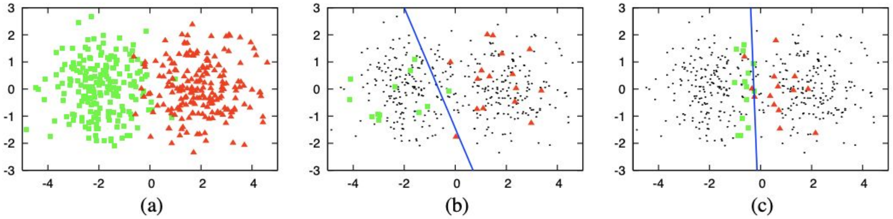
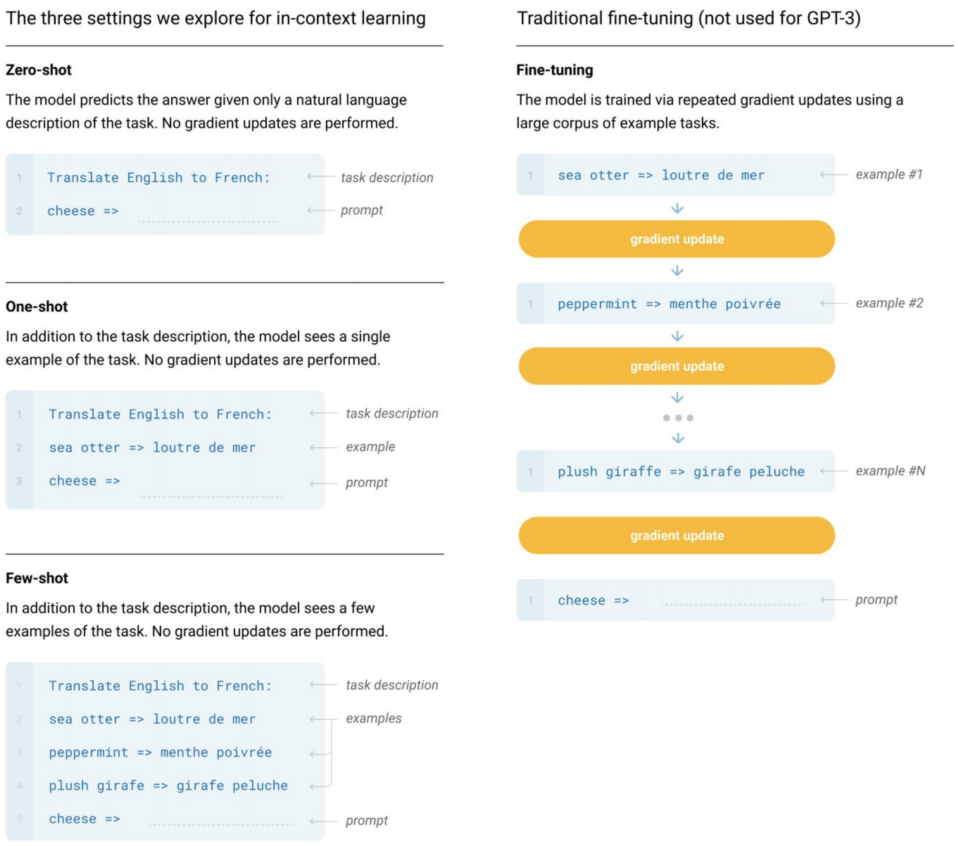


Figure 2: An illustrative example of pool-based active learning. (a) A toy data set of 400 instances, evenly sampled from two class Gaussians. The instances are represented as points in a 2D feature space. (b) A logistic regression model trained with 30 labeled instances randomly drawn from the problem domain. The line represents the decision boundary of the classifier (70% accuracy). (c) A logistic regression model trained with 30 actively queried instances using uncertainty sampling (90%).

Active Learning Literature Survey (Burr Settles, 2010)

Transfer learning

- Apply model trained for one task to another task
  - might or might not require fine-tuning



Language Models are Few-Shot Learners (OpenAI 2020)

Machine Learning Systems Design

Next class: Model development & training