CSE 291D/234
Data Systems for Machine Learning

Arun Kumar

Topic 4: Data Sourcing and Organization for ML

Chapters 8.1 and 8.3 of MLSys book
Data Sourcing in the Lifecycle

Data Scientist/ML Engineer

Source → Build → Deploy

ML/AI + Data Systems Infrastructure

Data acquisition
Data preparation

Feature Engineering
Training & Inference
Model Selection

Serving
Monitoring
Data Sourcing in the Big Picture

Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.
Overview

- Data Acquisition
- Data Reorganization and Preparation
- Data Cleaning and Validation
- Data Labeling
- Data Governance
Bias-Variance-Noise Decomposition

ML (Test) Error = Bias + Variance + Bayes Noise

Complexity of model/hypothesis space
Discriminability of examples

\[ x = (a, b, c); \ y = +1 \]
\[ vs \]
\[ x = (a, b, c); \ y = -1 \]
Q: How do real-world data scientists spend their time?

What data scientists spend the most time doing

- Building training sets: 3%
- Cleaning and organizing data: 60%
- Collecting data sets: 19%
- Mining data for patterns: 9%
- Refining algorithms: 4%
- Other: 5%

Q: How do real-world data scientists spend their time?

What’s the least enjoyable part of data science?

- Building training sets: 10%
- Cleaning and organizing data: 57%
- Collecting data sets: 21%
- Mining data for patterns: 3%
- Refining algorithms: 4%
- Other: 5%

# Data Science in the Real World

Q: How do real-world data scientists spend their time?

During a typical data science project at work or school, approximately what proportion of your time is devoted to the following?

<table>
<thead>
<tr>
<th>Activity</th>
<th>All</th>
<th>Data Scientist (N = 3310)</th>
<th>Software Engineer (N = 2067)</th>
<th>Research Scientist (N = 915)</th>
<th>Data Engineer (N = 539)</th>
<th>Data Analyst (N = 1385)</th>
<th>Business Analyst (N = 545)</th>
<th>Student (N = 3094)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gathering data</td>
<td>16.8%</td>
<td>15.9%</td>
<td>17.7%</td>
<td>18.0%</td>
<td>17.9%</td>
<td>17.3%</td>
<td>17.9%</td>
<td>15.3%</td>
</tr>
<tr>
<td>Cleaning data</td>
<td>22.9%</td>
<td>25.2%</td>
<td>19.9%</td>
<td>19.5%</td>
<td>25.6%</td>
<td>27.1%</td>
<td>26.5%</td>
<td>20.8%</td>
</tr>
<tr>
<td>Visualizing data</td>
<td>13.6%</td>
<td>12.9%</td>
<td>12.7%</td>
<td>13.2%</td>
<td>12.5%</td>
<td>15.6%</td>
<td>14.8%</td>
<td>14.2%</td>
</tr>
<tr>
<td>Model building / model selection</td>
<td>20.9%</td>
<td>20.3%</td>
<td>21.9%</td>
<td>25.5%</td>
<td>19.0%</td>
<td>16.0%</td>
<td>15.7%</td>
<td>24.3%</td>
</tr>
<tr>
<td>Putting model into production</td>
<td>9.0%</td>
<td>10.3%</td>
<td>9.7%</td>
<td>8.0%</td>
<td>10.8%</td>
<td>7.5%</td>
<td>6.6%</td>
<td>9.7%</td>
</tr>
<tr>
<td>Finding insights &amp; communicating with stakeholders</td>
<td>11.4%</td>
<td>12.6%</td>
<td>9.4%</td>
<td>10.7%</td>
<td>9.7%</td>
<td>13.5%</td>
<td>14.1%</td>
<td>9.8%</td>
</tr>
<tr>
<td>Other</td>
<td>5.3%</td>
<td>2.8%</td>
<td>8.6%</td>
<td>5.1%</td>
<td>4.5%</td>
<td>3.1%</td>
<td>4.3%</td>
<td>5.8%</td>
</tr>
</tbody>
</table>

Kaggle State of ML and Data Science Survey 2018
Q: How do real-world data scientists spend their time?

Data workers spend 90% of their work week on data related activities.

Activities Performed by Data Workers

- Preparation: 96%
- Searching: 92%
- Analytics: 88%
- Data Science: 76%
- Data App Dev: 59%

Searching for and preparing data are the most common activities regardless of role of data worker.
ML applications do not exist in a vacuum. They work with the **data-generating process** and prediction application.

**Sourcing:**
- The stage of where you go from raw datasets to “analytics/ML-ready” datasets
- Rough end point: Feature engineering/extraction
Q: What makes Sourcing challenging?

- Data access/availability constraints
- **Heterogeneity** of data sources/formats/types
- **Bespoke**/diverse kinds of prediction applications
- **Messy**, incomplete, ambiguous, and/or erroneous data
- Large **scale** of data
- Poor data **governance** in organization
Sourcing involves 4 high-level groups of activities:

1. Acquiring
2. Organizing
3. Cleaning
   - Feature Engineering (aka Feature Extraction)
4. Labeling (Sometimes)
Outline

❖ Overview
❖ Data Acquisition
❖ Data Reorganization and Preparation
❖ Data Cleaning and Validation
❖ Data Labeling
❖ Data Governance
Acquiring Data

1. Acquiring
2. Organizing
3. Cleaning

Feature Engineering (aka Feature Extraction)

1. Acquiring Data
2. Organizing
3. Cleaning
4. Labeling (Sometimes)
Acquiring Data: Data Sources

- Modern data-driven applications tend to have multitudes of data storage repositories and sources

- Structured data: Exported from RDBMSs (e.g., Redshift), often with SQL
- Semistructured data: Exported from “NoSQL” stores (e.g., MongoDB)
- Log files, text files, docs, multimedia, etc.: typically stored on HDFS, S3, etc.
- Graph/network data: Typically managed by systems such as Neo4j
Acquiring Data: Examples

**Example:** Recommendation System (e.g., Netflix)
**Prediction App:** Identify top movies to display for user

**Data Sources:**
- **mongoDB**: User data and past click logs
- **Amazon REDSHIFT**: Movie data
- **Amazon S3**: Movie images

**Example:** Social media analytics for social science
**Prediction App:** Predicts which tweets will go viral

**Data Sources:**
- **PostgreSQL**: Tweets as JSON, Structured metadata
- **SQLite**: Entity Dictionaries
- **neo4j**: Graph data
Modern data-driven applications tend to have multitudes of data storage repositories and sources

Potential challenges and mitigation:
- Access control: Learn organization’s data security and authentication policies
- Heterogeneity: Do you really need *all* data sources/types?
- Volume: Do you really need *all* data?
- Scale: Avoid copying files one by one
- Manual errors: Use automated workflow tools such as AirFlow
Acquiring Data: Data Discovery

- Some orgs have built “data discovery” tools to help ML users
- **Goal:** Make it easier to find relevant datasets
- **Approach:** Relevance ranking over schemas/metadata

**Example:**

![Google Dataset Search](https://storage.googleapis.com/pub-tools-public-publication-data/pdf/afd0602172f297bccdb4ee720bc3832e90e62042.pdf)

<table>
<thead>
<tr>
<th>Category</th>
<th>Number of datasets</th>
<th>% of total</th>
<th>Sample formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables</td>
<td>7,822K</td>
<td>37%</td>
<td>CSV, XLS</td>
</tr>
<tr>
<td>Structured</td>
<td>6,312K</td>
<td>30%</td>
<td>JSON, XML, OWL, RDF</td>
</tr>
<tr>
<td>Documents</td>
<td>2,277K</td>
<td>11%</td>
<td>PDF, DOC, HTML</td>
</tr>
<tr>
<td>Images</td>
<td>1,027K</td>
<td>5%</td>
<td>JPEG, PNG, TIFF</td>
</tr>
<tr>
<td>Archives</td>
<td>659K</td>
<td>3%</td>
<td>ZIP, TAR, RAR</td>
</tr>
<tr>
<td>Text</td>
<td>623K</td>
<td>3%</td>
<td>TXT, ASCII</td>
</tr>
<tr>
<td>Geospatial</td>
<td>376K</td>
<td>2%</td>
<td>SHP, GEOJSON, KML</td>
</tr>
<tr>
<td>Computational biology</td>
<td>110K</td>
<td>&lt;1%</td>
<td>SBML, BIOPAX2, SBGN</td>
</tr>
<tr>
<td>Audio</td>
<td>27K</td>
<td>&lt;1%</td>
<td>WAV, MP3, OGG</td>
</tr>
<tr>
<td>Video</td>
<td>9K</td>
<td>&lt;1%</td>
<td>AVI, MPG</td>
</tr>
<tr>
<td>Presentations</td>
<td>7K</td>
<td>&lt;1%</td>
<td>PPTX</td>
</tr>
<tr>
<td>Medical imaging</td>
<td>4K</td>
<td>&lt;1%</td>
<td>NII, DCM</td>
</tr>
<tr>
<td>Other categories</td>
<td>2,245K</td>
<td>11%</td>
<td></td>
</tr>
</tbody>
</table>
Acquiring Data: Tabular Datasets

- Tabular datasets especially amenable for augmentation
- Foreign keys (FK) implicitly suggest possible joins

Example:

- GOODS catalogs billions of tables within Google
- Extracts schema from file
- Assigns versions, owners
- Search and dashboards

Sometimes, tables joined in with primary key-FK joins may not help ML accuracy!

Hamlet showed avoiding FK join table does not alter noise; variance may rise; bias stays same or reduces

Decision rule to predict if a given FK join may hurt accuracy—before running ML

Intuition: If # training examples per FK value is high, “safe” to avoid the join

Tuple ratio rule quantifies how “high”

Outline

❖ Overview
❖ Data Acquisition
❖ Data Reorganization and Preparation
❖ Data Cleaning and Validation
❖ Data Labeling
❖ Data Governance
Organizing Data

1. Acquiring
2. Organizing
3. Cleaning
4. Labeling
(Sometimes)

Feature Engineering
(aka Feature Extraction)

Data Lake Repository
- Web
- Sensor
- Log
- Social
- Images
- Data extracted from relational DB

Raw data sources/repos

Build ML models
Reorganizing Data for ML

- Raw datasets sit in source platforms in their own formats
- Need to unify and *reorganize* them for ML tool

- How to reorganize depends on data types and analytics/ML task at hand
- May need SQL, MapReduce, and file I/O
- **Common steps:**
  - Change file formats (e.g., export table -> CSV -> TFRecords)
  - Decompression (e.g., multimedia)
  - Key-FK joins on tabular data
  - Key-key Joins for multimodal data
Reorganizing Data for ML: Examples

Prediction App: Fraud detection in banking

- Joins to denormalize
- Flatten JSON records
- Large single-table CSV file, say, on HDFS

Prediction App: Image captioning on social media

- Fuse JSON records
- Extract image tensors
- Large binary file with 1 image tensor and 1 string per line
Data preparation (“prep”) is often a synonym for data reorg.

Sometimes viewed as after major reorg. steps

Prep steps impact downstream bias-variance-noise

Figure 1: Illustrating major data prep tasks. The user loads a customers table to train, say, a churn predictor. BC stands for binary classification. MC stands for multi-class classification. Seq2Seq stands for sequence-to-sequence learning. Seq2SetSeq stands for sequence-to-set-of-sequence learning.
Typically, need coding (SQL, Python) and scripting (bash)

Some best practices:

- **Automation**: Use scripts for reorg. workflows
- **Documentation**: Maintain notes/READMEs for code
- **Provenance**: Manage metadata on source/rationale for each data source and feature
- **Versioning**: Reorg. is never one-and-done! Maintain logs of what version has what and when
Data Reorg./Prep for ML

“Feature stores” in industry help catalogue ML data (topic 6)
Data Reorg./Prep: Schematization

- "ML platforms" help streamline reorganization (topic 6)
  - Lightweight and flexible schemas now common
  - Makes it easier to automate data validation

https://www.tensorflow.org/tfx/guide
On ML platforms, ML itself can help automate many data prep/reorg. steps

Example: SortingHat’s ML-based feature type inference
Outline

❖ Overview
❖ Data Acquisition
❖ Data Reorganization and Preparation
❖ Data Cleaning and Validation
❖ Data Labeling
❖ Data Governance
Data Cleaning

1. Acquiring
2. Organizing
3. Cleaning
4. Labeling (Sometimes)

Feature Engineering (aka Feature Extraction)
Data Cleaning

❖ Real-world datasets often have errors, ambiguity, incompleteness, inconsistency, and other **quality** issues

❖ **Data cleaning**: Process of fixing data quality issues to ensure errors do not cascade/corrupt ML results

❖ 2 main stages: Error *detection/verification* -> *Repair*

What data scientists spend the most time doing

- Building training sets: 3%
- Cleaning and organizing data: 60%
- Collecting data sets: 19%
- Mining data for patterns: 9%
- Refining algorithms: 4%
- Other: 5%
Data Cleaning

Q: What causes data quality issues?

- Human-generated data: Mistakes, misunderstandings
- Hardware-generated data: Noise, failures
- Software-generated data: Bugs, errors, semantic issues
- Attribute encoding/formatting conventions (e.g., dates)
- Attribute unit/semantics conventions (e.g., km vs mi)
- Data integration: Duplicate entities, value differences
- Evolution of data schemas in application
Data Cleaning Task: Missing Values

❖ Long studied in statistics
❖ Various “missingness” assumptions based on relationship of missing vs observed values:
  ❖ Missing Completely at Random (MCAR): No (causal) relationships
  ❖ Missing at Random (MAR): Systematic relationships
  ❖ Missing Not at Random (MNAR): Missingness itself depends on the value missing
❖ Many ways to handle these:
  ❖ Add 0/1 missingness variable; impute missing values: statistical or ML/DL-based
  ❖ Many tools scale these computations (e.g., DaskML)
Data Cleaning Task: Entity Matching

- Often in multi-source datasets, real-world entities may have duplicate records
- W/o deduplication, query/ML accuracy can be hurt
- Aka entity deduplication/record linkage/entity linkage

<table>
<thead>
<tr>
<th>Customers1</th>
<th>Customers2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FullName</strong></td>
<td><strong>LastName</strong></td>
</tr>
<tr>
<td>Aisha Williams</td>
<td>Williams</td>
</tr>
<tr>
<td>Age</td>
<td>Age</td>
</tr>
<tr>
<td>San Diego</td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td></td>
</tr>
</tbody>
</table>

Q: Are these the same person (“entity”)?
General Workflow of Entity Matching

❖ 3 main stages: Blocking -> Pairwise check -> Clustering
❖ Pairwise check:
  ❖ Given 2 records, how likely is it that they are the same entity? SOTA: Entity embeddings + DL
❖ Blocking:
  ❖ Pairwise check cost for a whole table is too high: O(n^2)
  ❖ Create “blocks”/subsets of records; pairwise only within
  ❖ Domain-specific heuristics for obvious non-matches using similarity/distance metrics (e.g., edit dist. on Name)
❖ Clustering:
  ❖ Given pairwise scores, consolidate records into entities
Q: Is it even possible to automate data cleaning?

- Many approaches studied in DB and AI:
  - Integrity constraints, e.g., if ZipCode is same across customer records, State must be same too
  - Business logic/rules: domain knowledge programs
  - Supervised ML, e.g., predict missing values
- Alas, errors are often too peculiar and specific to dataset/application that manual cleaning (esp. repair) is the norm
  - “Death by a thousand cuts”
  - Crowdsourcing / expertsourcing another alternative
Automating Quality Checks: Deequ

- Some tools/libraries now help automate quality verification but workflow still hand-defined by humans; repair is manual

**Example:** Deequ from Amazon:
- *Verification* stage, not *repair*
- “Declarative” constraints
- API with many functions
- “Unit tests” analogy for data
- Scalable execution on Spark
## Automating Quality Checks: Deequ

<table>
<thead>
<tr>
<th>constraint</th>
<th>arguments</th>
<th>semantic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dimension completeness</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>isComplete</td>
<td>column</td>
<td>check that there are no missing values in a column</td>
</tr>
<tr>
<td>hasCompleteness</td>
<td>column, udf</td>
<td>custom validation of the fraction of missing values in a column</td>
</tr>
<tr>
<td><strong>dimension consistency</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>isUnique</td>
<td>column</td>
<td>check that there are no duplicates in a column</td>
</tr>
<tr>
<td>hasUniqueness</td>
<td>column, udf</td>
<td>custom validation of the unique value ratio in a column</td>
</tr>
<tr>
<td>hasDistinctness</td>
<td>column, udf</td>
<td>custom validation of the unique row ratio in a column</td>
</tr>
<tr>
<td>isInRange</td>
<td>column, value range</td>
<td>validation of the fraction of values that are in a valid range</td>
</tr>
<tr>
<td>hasConsistentType</td>
<td>column</td>
<td>validation of the largest fraction of values that have the same type</td>
</tr>
<tr>
<td>isNonNegative</td>
<td>column</td>
<td>validation whether all values in a numeric column are non-negative</td>
</tr>
<tr>
<td>isLessThan</td>
<td>column pair</td>
<td>validation whether values in the 1st column are always less than in the 2nd column</td>
</tr>
<tr>
<td>satisfies</td>
<td>predicate</td>
<td>validation whether all rows match predicate</td>
</tr>
<tr>
<td>satisfiesIf</td>
<td>predicate pair</td>
<td>validation whether all rows matching 1st predicate also match 2nd predicate</td>
</tr>
<tr>
<td>hasPredictability</td>
<td>column, column(s), udf</td>
<td>user-defined validation of the predictability of a column</td>
</tr>
</tbody>
</table>

**statistics (can be used to verify dimension consistency)**

<table>
<thead>
<tr>
<th>constraint</th>
<th>arguments</th>
<th>semantic</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasSize</td>
<td>udf</td>
<td>custom validation of the number of records</td>
</tr>
<tr>
<td>hasTypeConsistency</td>
<td>column, udf</td>
<td>custom validation of the maximum fraction of values of the same data type</td>
</tr>
<tr>
<td>hasCountDistinct</td>
<td>column</td>
<td>custom validation of the number of distinct non-null values in a column</td>
</tr>
<tr>
<td>hasApproxCountDistinct</td>
<td>column, udf</td>
<td>custom validation of the approx. number of distinct non-null values</td>
</tr>
<tr>
<td>hasMin</td>
<td>column, udf</td>
<td>custom validation of a column’s minimum value</td>
</tr>
<tr>
<td>hasMax</td>
<td>column, udf</td>
<td>custom validation of a column’s maximum value</td>
</tr>
<tr>
<td>hasMean</td>
<td>column, udf</td>
<td>custom validation of a column’s mean value</td>
</tr>
<tr>
<td>hasStandardDeviation</td>
<td>column, udf</td>
<td>custom validation of a column’s standard deviation</td>
</tr>
<tr>
<td>hasApproxQuantile</td>
<td>column, quantile, udf</td>
<td>custom validation of a particular quantile of a column (approx.)</td>
</tr>
<tr>
<td>hasEntropy</td>
<td>column, udf</td>
<td>custom validation of a column’s entropy</td>
</tr>
<tr>
<td>hasMutualInformation</td>
<td>column pair, udf</td>
<td>custom validation of a column pair’s mutual information</td>
</tr>
<tr>
<td>hasHistogramValues</td>
<td>column, udf</td>
<td>custom validation of column histogram</td>
</tr>
<tr>
<td>hasCorrelation</td>
<td>column pair, udf</td>
<td>custom validation of a column pair’s correlation</td>
</tr>
</tbody>
</table>

**time**

<table>
<thead>
<tr>
<th>constraint</th>
<th>arguments</th>
<th>semantic</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasNoAnomalies</td>
<td>metric, detector</td>
<td>validation of anomalies in time series of metric values</td>
</tr>
</tbody>
</table>
Data Validation in TFDV

- **Validation** is the process of enforcing expectations on data:
  - Is schema as expected?
  - Are features values from valid domains?
  - Catch anomalous features/values
  - Detection is automatic; repair is still manual

![Data Validation System Diagram](image)
Data Validation in TFDV

Key ideas in TFDV:

- Loosely coupled source schemas with constraints
- Catching training-serving skews (feature vs distribution)
- Unit tests to check model outputs

Figure 4: Schema-driven validation

<table>
<thead>
<tr>
<th>Anomaly Category</th>
<th>Used</th>
<th>Fired</th>
<th>Fixed given Fired</th>
</tr>
</thead>
<tbody>
<tr>
<td>New feature column (in data but not in schema)</td>
<td>100%</td>
<td>10%</td>
<td>65%</td>
</tr>
<tr>
<td>Out of domain values for categorical features</td>
<td>45%</td>
<td>6%</td>
<td>66%</td>
</tr>
<tr>
<td>Missing feature column (in schema but not in data)</td>
<td>97%</td>
<td>6%</td>
<td>53%</td>
</tr>
<tr>
<td>The fraction of examples containing a feature is too small</td>
<td>97%</td>
<td>3%</td>
<td>82%</td>
</tr>
<tr>
<td>Too small feature value vector for example</td>
<td>98%</td>
<td>2%</td>
<td>56%</td>
</tr>
<tr>
<td>Too large feature value vector for example</td>
<td>98%</td>
<td>&lt;1%</td>
<td>28%</td>
</tr>
<tr>
<td>Data completely missing</td>
<td>100%</td>
<td>3%</td>
<td>65%</td>
</tr>
<tr>
<td>Incorrect data type for feature values</td>
<td>98%</td>
<td>&lt;1%</td>
<td>100%</td>
</tr>
<tr>
<td>Non-boolean value for boolean feature type</td>
<td>14%</td>
<td>&lt;1%</td>
<td>100%</td>
</tr>
<tr>
<td>Out of domain values for numeric features</td>
<td>67%</td>
<td>1%</td>
<td>77%</td>
</tr>
</tbody>
</table>
Discussion on TFDV paper
Outline

❖ Overview
❖ Data Acquisition
❖ Data Reorganization and Preparation
❖ Data Cleaning and Validation
❖ Data Labeling
❖ Data Governance
Data Labeling

1. Acquiring
2. Organizing
3. Cleaning
4. Labeling (Sometimes)

Feature Engineering (aka Feature Extraction)

Raw data sources/repos
Build ML models

Data Lake Repository
- Web
- Sensor
- Log
- Social
- Images
- Data extracted from relational DB

44
Data Labeling

- Most recent AI successes due to *supervised* ML
- Large dataset is not enough—need *labeled* datasets, i.e., pairs of (input, output) examples

Object detection performance when pre-trained on different subsets of JFT-300M from scratch. x-axis is the dataset size in log-scale, y-axis is the detection performance in mAP@[.5,.95] on COCO-minival subset.

Data Labeling

Labeling: Process of annotating an example (raw or featurized) with ground truth label for a given prediction task. Notion of “label” is prediction task-specific and data type-specific; can be almost any data structure!

Q: What is a label for this image?

Dog (object recognition)
Couch (object recognition)
Shiba Inu (dog breed classifier)
Yes (meme classifier!)
Dog w/ bounding box (obj. detection)
Highlight dog (segmentation)
WRT sources of labels, 3 kinds of prediction applications:

1. Data-generating *process offers labels naturally*
   E.g.: Customer churn prediction, forecasting

2. Product/service *users offer labels (in)directly*
   E.g.: Email spam filters, online advertising, product recommendations, photo tagging, web search

3. Need *application-specific extra effort* for labels
   E.g.: Radiology, self-driving cars, species classification, video surveillance, machine translation, knowledge base construction, document summarization
Data Labeling Approaches

How to get more labeled training data?

- **Traditional Supervision:** Have subject matter experts (SMEs) hand-label more training data

- **Semi-supervised Learning:** Use structural assumptions to automatically leverage unlabeled data

- **Weak Supervision:** Get lower-quality labels more efficiently and/or at a higher abstraction level

- **Transfer Learning:** Use models already trained on a different task

**Active Learning:** Estimate which points are most valuable to solicit labels for

- **Get cheaper, lower-quality labels from non-experts**

- **Get higher-level supervision over unlabeled data from SMEs**

**Heuristics**
- **Distant Supervision**
- **Expected distributions**
- **Invariances**

[https://www.snorkel.org/blog/weak-supervision](https://www.snorkel.org/blog/weak-supervision)
Data Labeling Approaches

5 most common approaches to acquiring labels:

1. **Manual supervision** by subject matter experts (SMEs)
   
   Traditional approach; slow and expensive but common

2. **Active learning** with SMEs (less common)
   
   Prioritize which unlabeled examples SME must label based on benefit; possible for some kinds of ML; pay-as-you-go

3. **Crowdsourcing**; expertsourcing
   
   For tasks where lay people intelligence suffices; o/w if task is more technical, get workers with domain expertise

4. **Programmatic supervision**

5. **Transfer learning-based** supervision
Programmatic Supervision

❖ **Basic Idea:** Instead of manually labeling each example, write programs/rules/heuristics that encode some domain intuition to label examples en masse.

❖ **Pros:** Improved labeling productivity; likely lower costs.

❖ **Cons:** Need to write code; less reliable accuracy; unclear if complex prediction outputs supportable.

[Diagram of Programming Stack and Supervision Stack]

Programmatic Supervision: Snorkel

- **Snorkel**: A prog. framework/tool for weak supervision
  - Users can give various forms of supervision
  - Snorkel “denoises” the labels using statistical techniques
  - Output is a probability distribution over class labels
Snorkel now allows users to input 3 kinds of functions:

- **Labeling Training Data**
  - Higher level rules/sources for labeling example
    \[{x_i} \rightarrow {y_i}\]

- **Data Augmentation**
  - Semi-synthetically create more labeled examples
    \[{(x_i,y_i)} \rightarrow {({x_j′,y_j′})}\]

- **Monitoring Critical Data Subsets**
  - Monitor accuracy on specific data subsets; more focused augmentation

[https://www.snorkel.org/](https://www.snorkel.org/)
Transfer Learning

❖ **Basic Idea:** Use a model pre-trained on a different but related task (maybe it had large labeled dataset) to reduce labeled data needs of your task

❖ Works well for image/vision and text/NLP

❖ If target task is a *subset* of source task: just use its outputs as pseudo-labels!

[Source Task](#) [Target Task](#)

https://medium.com/the-official-integrate-ai-blog/transfer-learning-explained-7d275c1e34e2
Review Zoom Poll
Outline

❖ Overview
❖ Data Acquisition
❖ Data Reorganization and Preparation
❖ Data Cleaning and Validation
❖ Data Labeling
❖ Data Governance
Data Governance

❖ Data are “entities” with “value”—kinda like people? :)
  ❖ Born/created, live/used, die/deleted, stewarded, protected, managed, etc.
  ❖ Just as people must be governed, so must data

❖ Key aspects of governing data:
  ❖ Cataloging: What is it, where, how to access?
  ❖ Defining: Data dictionaries, business knowledge.
  ❖ Quality: Follow conventions, reduce errors.
  ❖ Provenance: Track usage, changes, evolution. Audit.
Legal Regulations on Data Handling

❖ Just as laws exist to govern people, laws exist to govern data
❖ No laws (yet) on ML “algorithms”, but yes for ML data
❖ Long history of laws surrounding data:

FERPA
1974
Broadly applies to all “education records” of students

FERPA is a federal law that protects the privacy of student records and applies to all schools that receive funds from the USDOE.

Types of Student Records:
- Financial information
- Disciplinary files
- Student transcripts
- Immunization & health records

To be compliant, schools can utilize a paperless system for storing student records. School’s funding is based on compliance.

https://www.recordnations.com/2019/07/ferpa-how-to-manage-student-records
Legal Regulations on Data Handling

HIPAA; 1996
Broadly applies to all healthcare data, especially PII

5 Steps Toward HIPAA Compliance
FOR SECURE COMMUNICATION AND COLLABORATION

1. Determine security requirements.
2. Identify types of data sent and set protocols.
3. Think about how data could be leaked or lost.
4. Implement secure communication practices.
5. Educate staff on security policies.

Produced in collaboration with ADA Business Resources and PBHS.com

Access tips at Success.ADA.org
Legal Regulations on Data Handling

HIPAA FACTS

1 in 7 healthcare organizations have still not appointed a HIPAA compliance officer

80% of healthcare organizations fail meaningful use audits

1 in 4 HIPAA breaches still not reported

Unauthorized access to records accounts for 20% of HIPAA breaches

50% of healthcare organizations believe they would fail a HIPAA Audit

Phishing and ransomware: the top hacker tactic

Average cost per record is $363

Ransomware for medical records accelerating rapidly

About 600 HIPAA Violations referred to DoJ

Nearly 200 million patient records compromised since introduction of HIPAA

Copyright © 2018 The HIPAA Guide
Legal Regulations on Data Handling

❖ Broadly applies to any data collected from individuals in the EU and EEA
❖ Offers many new rights on “personal data”: right to access, right to forget/erasure, right to object, etc.
❖ Many Web companies scrambled; some “exited” EU area
Legal Regulations on Data Handling

- New technical challenges on making data/ML infra. GDPR-compliant: metadata handling, efficiency, etc.
- Open legal+technical questions for ML applications:
  - Are ML models under purview?
  - Any form of derived / aggregated data?
Benchmarking Impact of GDPR

- GDPR compliance may make data systems slower
- Prior benchmarks TPC and YCSB not enough
- GDPRBench: New benchmark to study GDPR impact:
  - Formalizes workloads of GDPR-mandated agents
  - Redis faces 5x overhead; PostgreSQL 2x

https://www.gdprbench.org/
## Legal Regulations on Data Handling

### CCPA

**January 1, 2020**
- Enforcement begins July 1, 2020

**For-Profit Companies That:**
- Collect personal data on 50K+ California residents
- Have annual revenues of over $25 million
- Earn 50%+ of annual revenue from California residents’ data

- Business, service providers, third parties, and California consumers

- Personal data that is sold for monetary or other value considerations (releasing, disclosing, transferring, or even renting of the data)

- Up to $7,500 per violation with no ceiling on the number of violations
- $100-$750 per consumer per incident for statutory damages related to breaches

**When Does the Law Go Into Effect?**

**GDPR**

**May 25, 2018**
- Enforcement in effect

**Any Organization That:**
- Operates inside or outside the European Union (EU) and offers goods or services to customers or businesses in the union

- EU citizens, businesses, controller, processor, and data subjects

- Personal data of any type

**Who is Affected?**

**What Data is Within Scope?**

**What Are the Fines of Noncompliance?**

- Up to 20 million euros or 4% of total global turnover from the prior fiscal year for the most severe violations
- Up to 10 million euros or 2% of the worldwide annual revenue of the prior fiscal year for less severe violations

Provenance Management

❖ All data objects must be tracked throughout lifecycle
  ❖ Compliance with data regulations; auditing
  ❖ Makes data easier to find and consume
❖ Provenance: “Chronology of the ownership, custody or location of a historical object”
❖ Key aspects of provenance:
  ❖ Context of data creation/deletion, access/use, etc.
  ❖ Evolution of metadata
  ❖ Versioning of data and all derived objects
❖ For ML: track derived data (e.g., feature extraction), ML artifacts (models, code/scripts, etc.), & configuration
Provenance Management

❖ **Challenge**: Heterogeneity of data/ML platforms makes it notoriously messy/tedious
  ❖ Metadata? Usage logs? Versioning?
  ❖ SOTA: ad hoc or organization-specific practices

❖ **Ground**: A new unified methodology/tool to raise level of abstraction for metadata, provenance, etc.

- **Application Context**: Views, models, code
- **Behavioral Context**: Data lineage & usage
- **Change Over Time**: Version histories

https://speakerdeck.com/jhellerstein/ground-a-data-context-service
http://www.ground-context.org/
Managing Data Context in Ground

❖ **Ground**: A new unified methodology/tool to raise level of abstraction for metadata, provenance, etc.
  ❖ “Meta-model” to unify metadata and provenance, aka data context

❖ **Desiderata:**
  ❖ Agnostic to data model: variety, heterogeneity
  ❖ Immutable: consistency, quality, backwards-compatible
  ❖ Scalable: volume, versioning-friendly
  ❖ “Politically” neutral: integrate with many platforms

https://speakerdeck.com/jhellerstein/ground-a-data-context-service
http://www.ground-context.org/
Managing Data Context in Ground

- New “metamodel” to unifying metadata and provenance
- Graphs with node, edge, and sub-graph properties
- Schemas, ontologies, usage logs all cast onto this metamodel

https://speakerdeck.com/jhellerstein/ground-a-data-context-service
http://www.ground-context.org/
Managing Data Context in Ground

- New “metamodell” to unifying metadata and provenance

https://speakerdeck.com/jhellerstein/ground-a-data-context-service
http://www.ground-context.org/
Managing Data Context in Ground

Open research questions:

**Underground**
- Workloads
- Common Ground representations
- No-overwrite versioned DB
- Time travel queries: point and trend Graph queries + log analysis
- Consistency

**Aboveground**
- Content extraction
- Analytic user exhaust
- Socio-technical networks
- Collective governance
- Reproducibility
- Lifecycle of systems that learn

https://speakerdeck.com/jhellerstein/ground-a-data-context-service
http://www.ground-context.org/
Review Questions

❖ Briefly explain a major source of hints in tabular data that enables ML users to find more tables to join in.
❖ Briefly explain 2 benefits of acquiring extra tables to join in when applying ML over tabular data.
❖ What are the two main stages of data cleaning?
❖ How does the blocking stage of entity matching help?
❖ Briefly explain 2 common best practices for data reorganization discussed in class.
❖ Name 2 pros of programmatic labeling over hand labeling.
❖ Which class of functions in Snorkel is primarily meant to automatically create extra training examples?
❖ Name a data law that affects many Web companies.