Machine Learning in Production

Versioning, Provenance, and Reproducability

Foundational Technology for Responsible Engineering

Fundamentals of Engineering AI-Enabled Systems

Holistic system view: AI and non-AI components, pipelines, stakeholders, environment interactions, feedback loops

Requirements:
System and model goals
User requirements
Environment assumptions
Quality beyond accuracy
Measurement
Risk analysis
Planning for mistakes

Architecture + design:
Modeling tradeoffs
Deployment architecture
Data science pipelines
Telemetry, monitoring
Anticipating evolution
Big data processing
Human-AI design

Quality assurance:
Model testing
Data quality
QA automation
Testing in production
Infrastructure quality
Debugging

Operations:
Continuous deployment
Contin. experimentation
Configuration mgmt.
Monitoring
Versioning
Big data
DevOps, MLOps

Teams and process: Data science vs software eng. workflows, interdisciplinary teams, collaboration points, technical debt

Responsible AI Engineering

Provenance, versioning, reproducibility

Safety

Security and privacy

Fairness

Interpretability and explainability

Transparency and trust

Ethics, governance, regulation, compliance, organizational culture

Readings

Required readings

- Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.F. and Dennison, D., 2015. Hidden Technical Debt in Machine Learning Systems. In Advances in neural information processing systems (pp. 2503-2511).

Learning Goals

- Judge the importance of data provenance, reproducibility and explainability for a given system
- Create documentation for data dependencies and provenance in a given system
- Propose versioning strategies for data and models
- Design and test systems for reproducibility

Case Study: Credit Scoring

DHH
@dhh · Follow

The @AppleCard is such a fucking sexist program. My wife and I filed joint tax returns, live in a community-property state, and have been married for a long time. Yet Apple's black box algorithm thinks I deserve 20x the credit limit she does. No appeals work.

8:34 PM · Nov 7, 2019

22.7K   Reply   Copy link

Read 1.1K replies


DHH · Nov 8, 2019
@dhh · Follow
Replying to @dhh
I wasn’t even pessimistic to expect this outcome, but here we are: @AppleCard just gave my wife the VIP bump to match my credit limit, but continued to be an utter fucking failure of a customer service experience. Let me explain...

DHH
@dhh · Follow

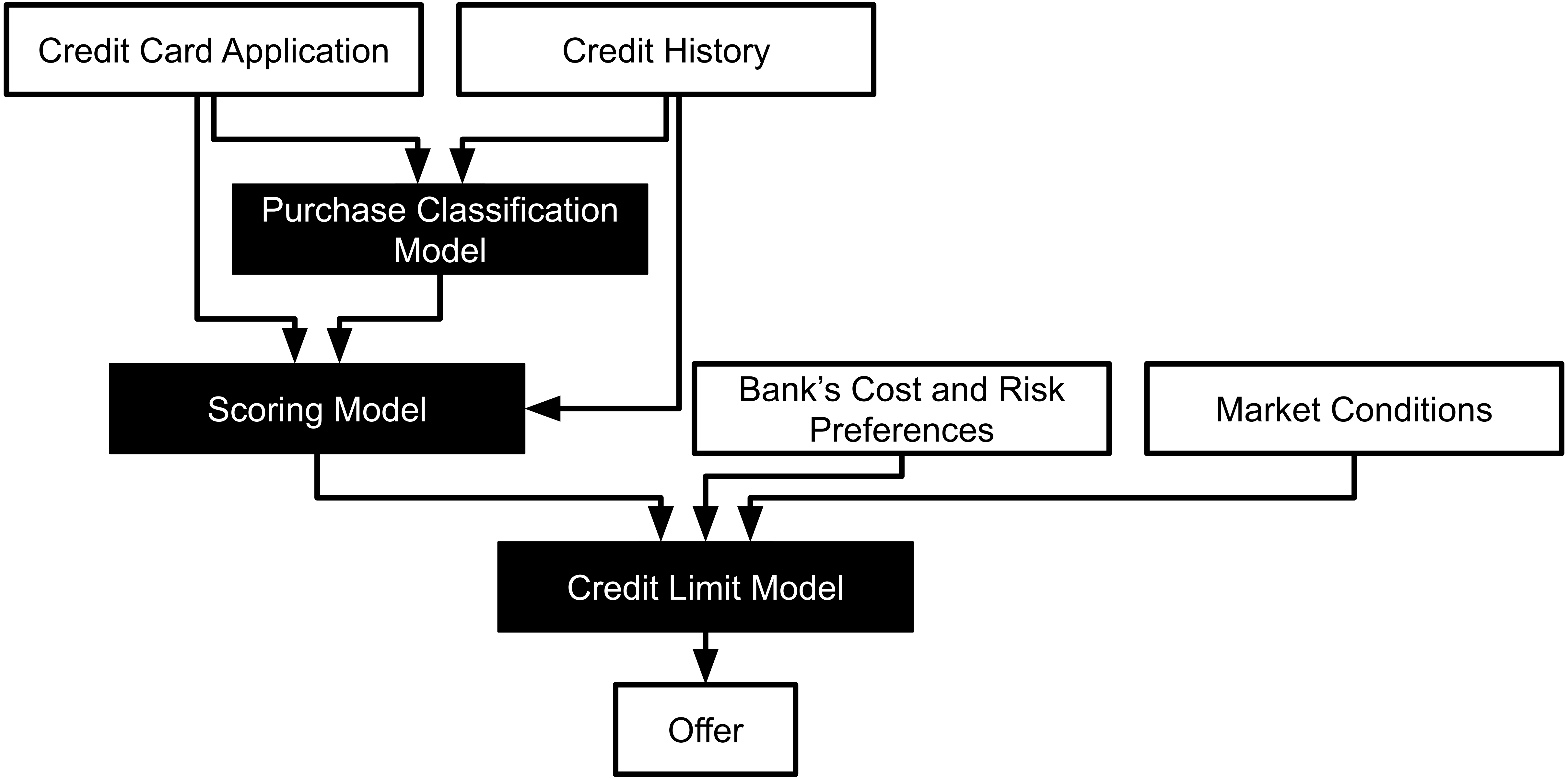
She spoke to two Apple reps. Both very nice, courteous people representing an utterly broken and reprehensible system. The first person was like “I don’t know why, but I swear we’re not discriminating, IT’S JUST THE ALGORITHM”. I shit you not. “IT’S JUST THE ALGORITHM!”.

11:20 PM · Nov 8, 2019

3.6K Reply Copy link to post

Read 52 replies

- What model was used? Can we reproduce?
- What caused the issue? What data was used?


Credit Card Application
Credit History
Purchase Classification Model
Scoring Model
Bank's Cost and Risk Preferences
Market Conditions
Credit Limit Model
Offer

Debugging?

What went wrong? Where? How to fix?

Debugging Questions beyond Interpretability

- Can we reproduce the problem?
- What were the inputs to the model?
- Which exact model version was used?
- What data was the model trained with?
- What pipeline code was the model trained with?
- Where does the data come from? How was it processed/extracted?
- Were other models involved? Which version? Based on which data?
- What parts of the input are responsible for the (wrong) answer? How can we fix the model?

Breakout Discussion: Movie Predictions

> *Assume you are receiving complains that a child gets many recommendations about R-rated movies*

In a group, discuss how you could address this in your own system and post to `#lecture`, tagging team members:

- How could you identify the problematic recommendation(s)?
- How could you identify the model that caused the prediction?
- How could you identify the training code and data that learned the model?
- How could you identify what training data or infrastructure code "caused" the recommendations?

K.G Orphanides. Children's YouTube is still churning out blood, suicide and cannibalism. Wired UK, 2018; Kristie Bertucci. 16 NSFW Movies Streaming on Netflix. Gadget Reviews, 2020

Practical Data and Model Versioning

- Versioning the data
- Versioning the model

Versioning Large Datasets

- Track the history of data
- Trace back the data used to train a model, even after we make some changes to the data

How to Version Large Datasets?

```
InquiryID,CustomerID,InquiryDate,LoanType,LoanAmount,AccountSt
1001,001,2020-01-15,Mortgage,250000,Open,Current
1002,002,2020-02-20,Auto Loan,20000,Closed,Paid Off
1003,003,2020-03-05,Credit Card,5000,Open,Late (30 days)
1004,004,2020-04-10,Personal Loan,10000,Open,Current
1005,005,2020-05-15,Student Loan,30000,Closed,Paid Off
1006,001,2020-06-20,Mortgage,200000,Open,Current
1007,002,2020-07-25,Credit Card,7000,Open,Late (60 days)
1008,003,2020-08-30,Auto Loan,15000,Closed,Paid Off
1009,004,2020-09-10,Personal Loan,8000,Open,Current
1010,005,2020-10-15,Credit Card,10000,Open,Late (90 days)
```

(example customer data from the credit scenario, can be TBs!)

How to Version Large Datasets?

Recall: Event Sourcing

- **Append only databases + offsets**
- Record edit events, never mutate data
- Compute current state from all past events, can reconstruct old state
- For efficiency, take state snapshots
- Similar to traditional database logs

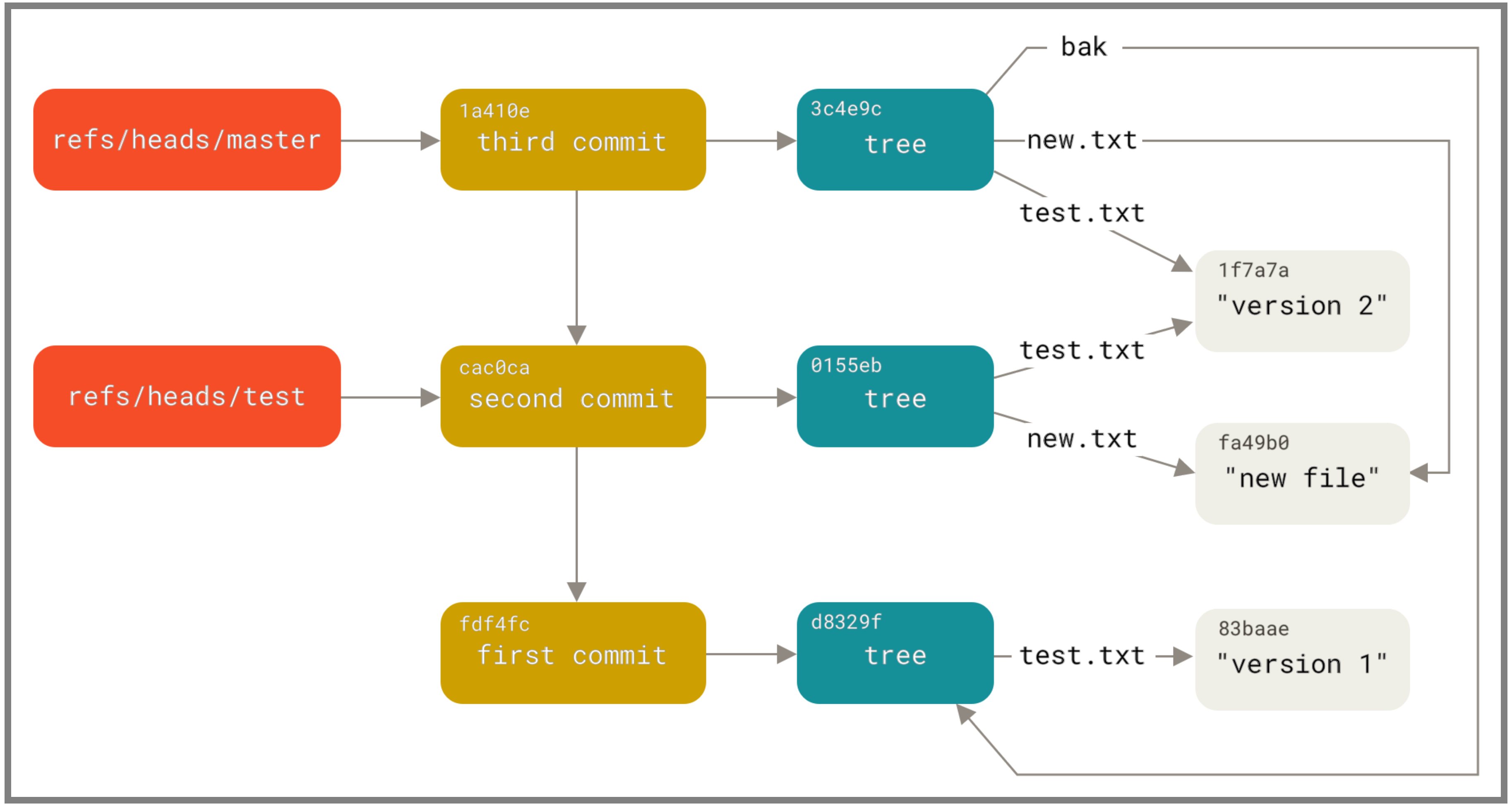
```
createUser(id=5, name="Christian", dpt="SCS")
updateUser(id=5, dpt="ISR")
deleteUser(id=5)
```

Versioning Strategies for Datasets

1. Store copies of entire datasets (like Git), identify by checksum
2. Store deltas between datasets (like Mercurial)
3. History of individual database records (e.g. S3 bucket versions)
   - some databases specifically track provenance (who has changed what entry when and how)
   - specialized data science tools eg Hangar for tensor data
4. Offsets in append-only database (like Kafka), identify by offset
5. Version pipeline to recreate derived datasets ("views", different formats)
   - e.g. version data before or after cleaning?

≡

Aside: Git Internals



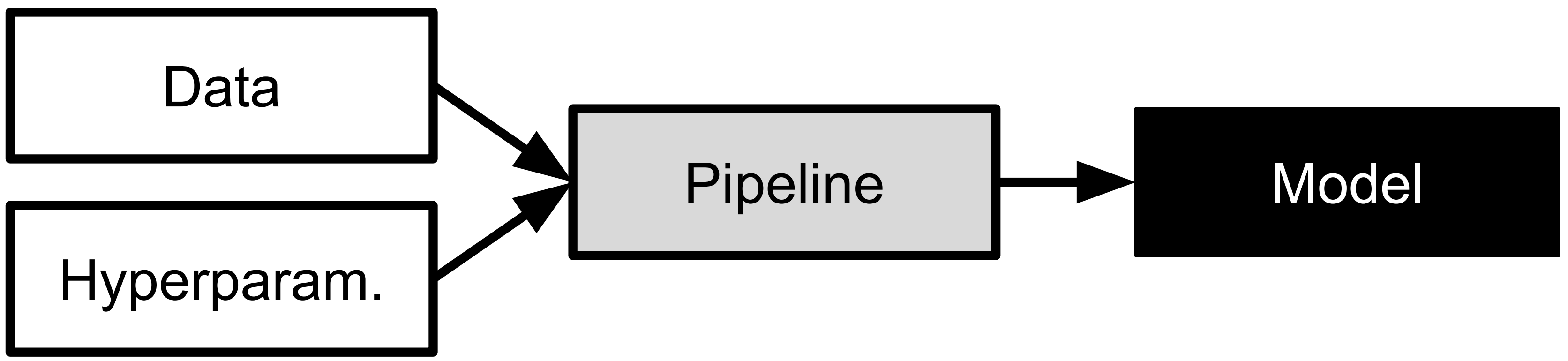
Versioning Models

Versioning Models

Usually no meaningful delta/compression, version as binary objects

Any system to track versions of blobs

Versioning Models: Multiple Parts!



Associate model version with:

- pipeline code version
- data version
- hyperparameters

ML Versioning Tools (MLOps)

Tracking **data**, **pipeline**, and **model versions**

Modeling pipelines: inputs and outputs and their versions

- automatically tracks how data is used and transformed

Often tracking also metadata about versions

- Accuracy
- Training time
- ...

Example: ModelDB

- Low complexity
- Your responsibility to design what to log and when!

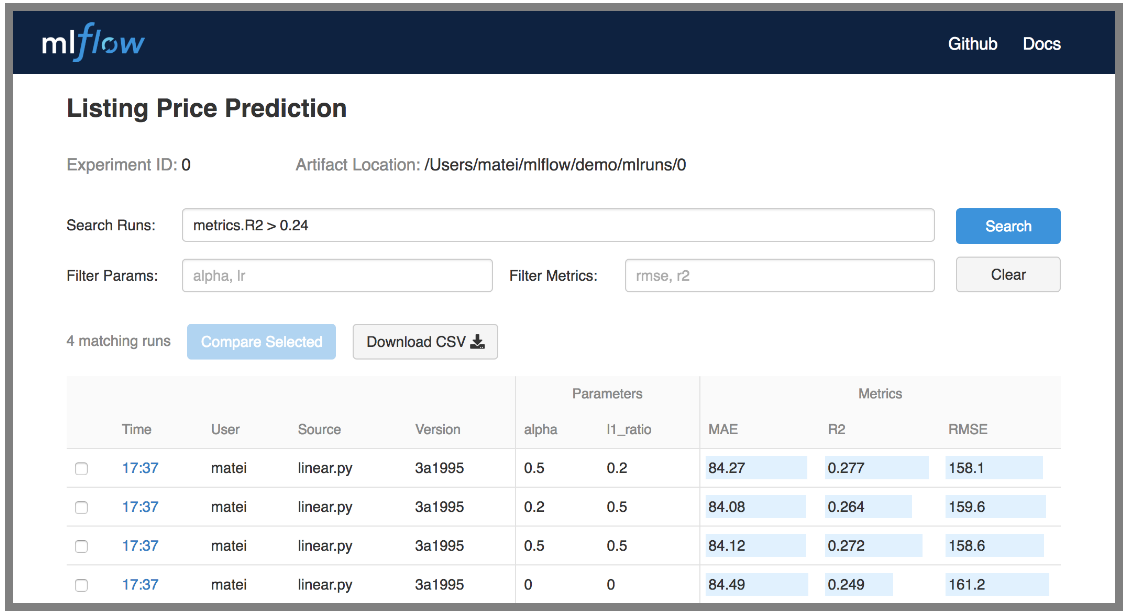
```
from verta import Client
client = Client("http://localhost:3000")

proj = client.set_project("My first ModelDB project")
expt = client.set_experiment("Default Experiment")

# log the first run
run = client.set_experiment_run("First Run")
run.log_hyperparameters({"regularization" : 0.5})
run.log_dataset_version("training_and_testing_data", dataset_v
model1 = # ... model training code goes here
```

Example w/ Metadata: Experiment Tracking

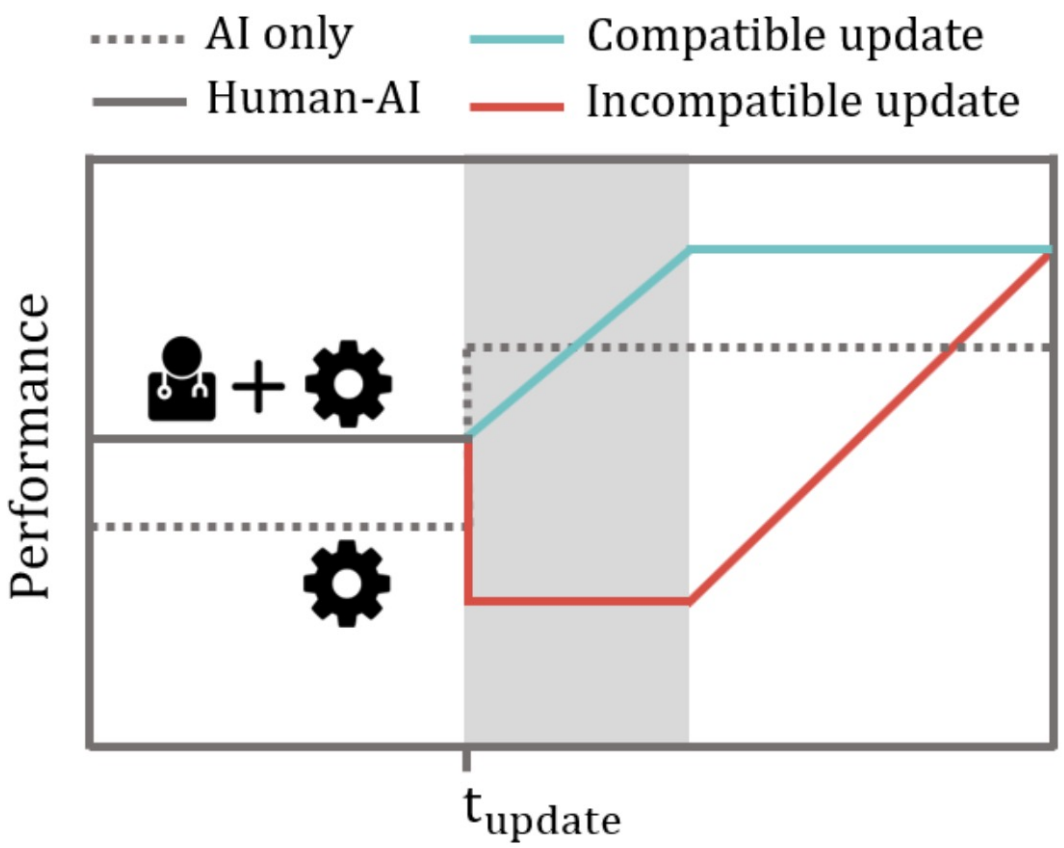
Log information within pipelines: **hyperparameters used**, **evaluation results**, and **model files**



Speaker notes

Image from Matei Zaharia. Introducing MLflow: an Open Source Machine Learning Platform, 2018

Compatibility in Versioning



Bansal, Gagan, et al. "Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff." AAAI 2019

Example: DVC (Data Version Control)

```
dvc add images
dvc run -d images -o model.p cnn.py
dvc remote add myrepo s3://mybucket
dvc push
```

- Tracks models and datasets, built on Git
- Define a sequence of steps (e.g. data processing, training, evaluation) in a reproducible pipeline, incrementalization
- Orchestrates learning in cloud resources: Detect changes in any step and re-run only the necessary parts when there's an update

https://dvc.org/

DVC Example

- Higher complexity and constraints
- More automation and less manual work

```yaml
stages:
  features:
    cmd: jupyter nbconvert --execute featurize.ipynb
    deps:
      - data/clean
    params:
      - levels.no
    outs:
      - features
    metrics:
      - performance.json
```

Google's Goods

Automatically derive data dependencies from system log files

Track metadata for each table

No manual tracking/dependency declarations needed

Requires homogeneous infrastructure

Similar systems for tracking inside databases, MapReduce, Sparks, etc.

Versioning Dependencies

Pipelines depend on many frameworks and libraries

Ensure reproducible builds

- Declare versioned dependencies from stable repository (e.g. requirements.txt + pip)
- Avoid floating versions
- Optionally: commit all dependencies to repository ("vendoring")

Optionally: Version entire environment (e.g. Docker container)

Test build/pipeline on independent machine (container, CI server, ...)

From Model Versioning to Deployment

Decide which model version to run where

- automated deployment and rollback (cf. canary releases)
- Kubernetis, Cortex, BentoML, ...

Track which prediction has been performed with which model version (logging)

Logging and Audit Traces

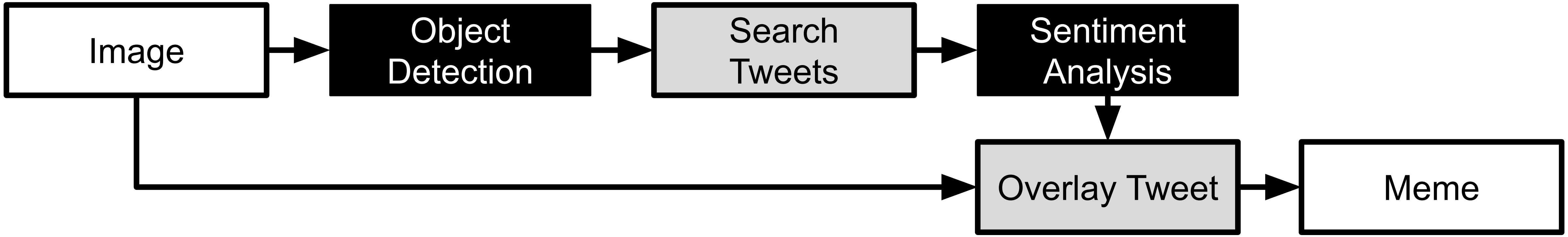
Key goal: If a customer complains about an interaction, can we reproduce the prediction with the right model? Can we debug the model's pipeline and data? Can we reproduce the model?

- Version everything
- Record every model evaluation with model version
- Append only, backed up

```
<date>,<model>,<model version>,<feature inputs>,<output>
<date>,<model>,<model version>,<feature inputs>,<output>
<date>,<model>,<model version>,<feature inputs>,<output>
```

Logging for Composed Models

Ensure all predictions are logged

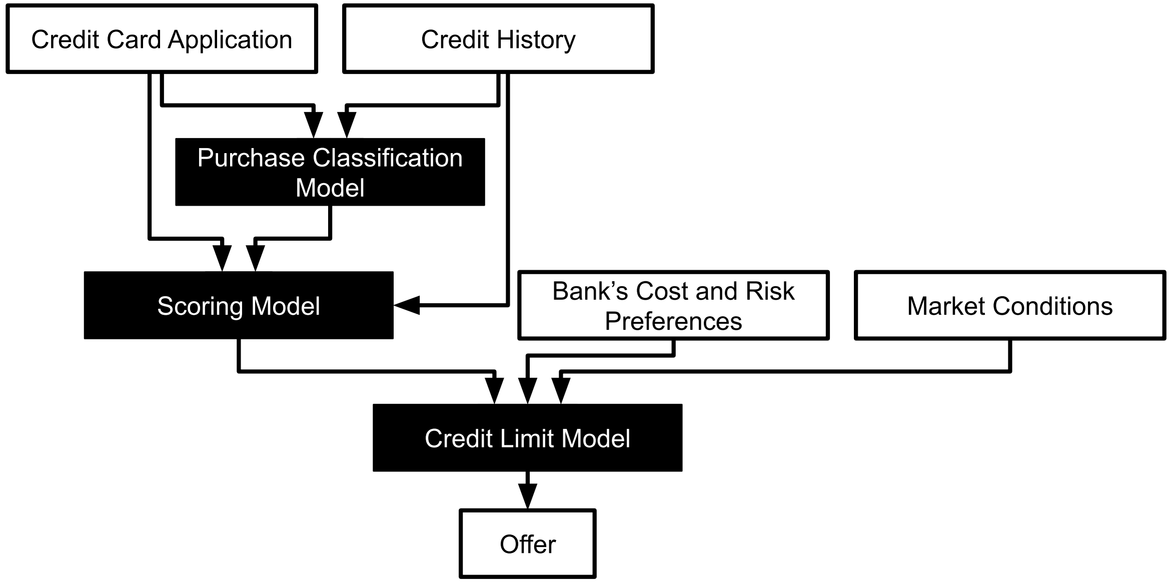


Provenance Tracking

Historical record of data and its origin

Data Provenance

- Track origin of all data
  - Collected where?
  - Modified by whom, when, why?
  - Extracted from what other data or model or algorithm?
- ML models often based on data drived from many sources through many steps, including other models



Versioning vs Provenance

Feature	Data Versioning	Data Provenance
Purpose	Tracks changes in data over time	Documents data's origin, lifecycle, usage
Primary Focus	Data states / changes between versions	Data lineage, sources, and transformations
Granularity	Focuses on entire datasets or files	Tracks individual data entries and transformations
Application	Rollbacks, collaborative editing	Compliance, auditing, trust, and traceability
Example	Version 1.0, 1.1, 2.0 of a dataset	Source, timestamp, consent for each entry

Excursion: Provenance Tracking in Databases

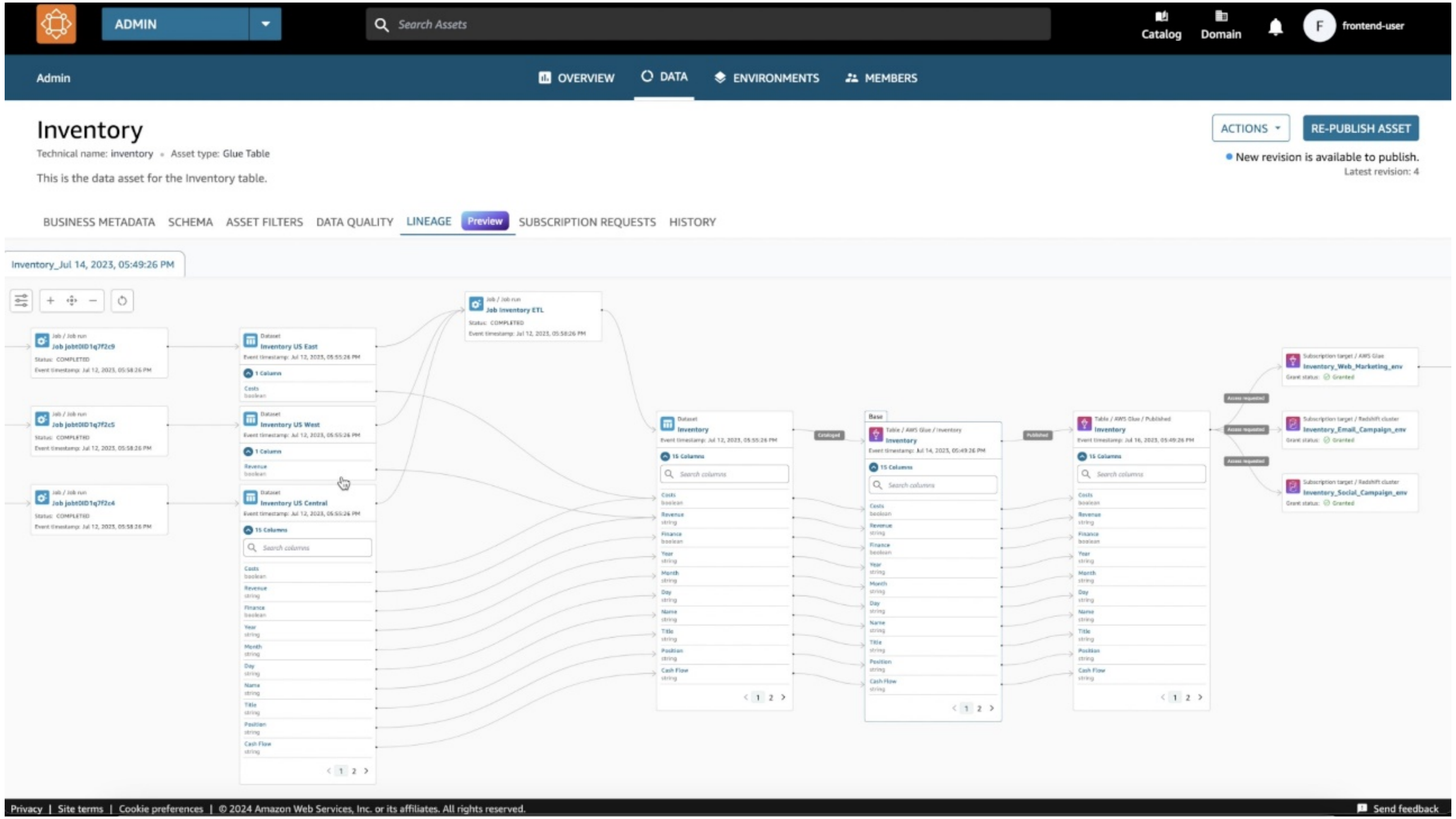
Whenever value is changed, record:

- who changed it
- time of change
- history of previous values
- possibly also justifcation of why

Embedded as feature in some databases or implemented in business logic

Possibly signing with cryptographic methods

Tracking Data Lineage



Tracking Data Lineage

Document all data sources

Identify all model dependencies and flows

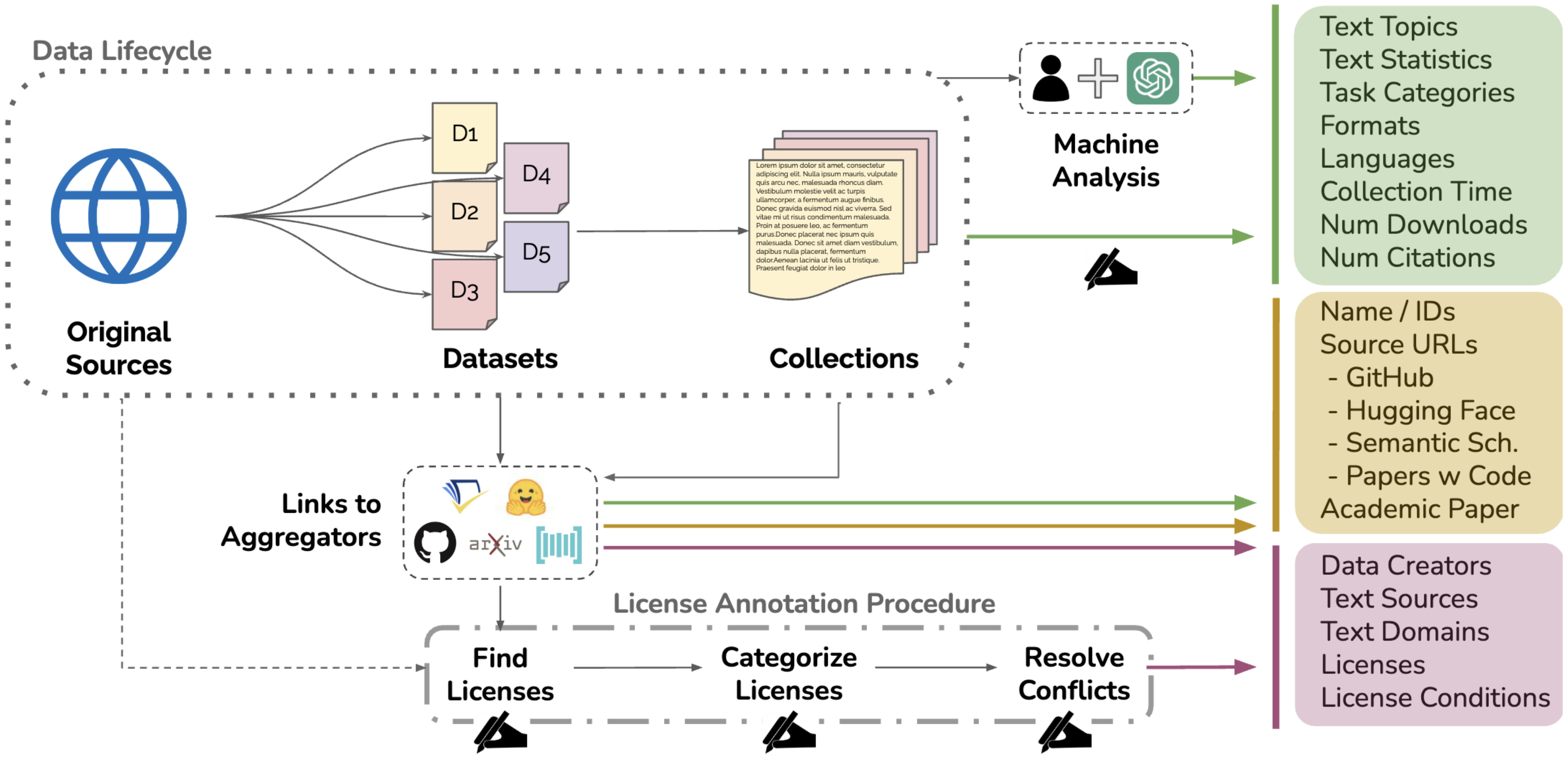
Ideally model all data and processing code

Avoid "visibility debt"

(Advanced: Use infrastructure to automatically capture/infer dependencies and flows as in Goods)

Example: Data Provenance Initiative

Key idea: Need to recover data provenance for existing datasets!



Example: Data Provenance Initiative

Can make many interesting observations…

Collection	Property Counts							Text Lens		Dataset Types							
	Datasets	Dialogs	Tasks	Langs	Topics	Domains	Downs	Inpt	Tgt	Source	Z	F	C	R	M	Use	O
Airoboros	1	17k	5	2	10	1	1k	347	1k	🤖	✔					🔴	✔
Alpaca	1	52k	8	1	10	1	100k	505	270	🤖	✔					🔴	✔
Anthropic HH	1	161k	3	1	10	1	82k	69	311	🤖				✔		🔵	
BaizeChat	4	210k	12	2	37	3	<1k	74	234	🤖	✔					🔴	✔
BookSum	1	7k	4	1	10	1	<1k	14k	2k	🌐	✔					🔴	
CamelAI Sci.	3	60k	2	1	29	1	<1k	190	2k	🤖	✔					🔴	✔
CoT Coll.	6	2,183k	12	7	29	1	<1k	728	265	🤖			✔			🔴	✔
Code Alpaca	1	20k	3	2	10	1	5k	97	196	🤖	✔					🟡	✔
CommitPackFT	277	702k	1	278	751	1	4k	645	784	🌐	✔					🔵🟡	
Dolly 15k	7	15k	5	1	38	1	10,116k	423	357	🌐	✔					🔵	
Evol-Instr.	2	213k	11	2	17	1	2k	570	2k	🤖	✔					🔴	✔
Flan Collection	450	9,813k	19	39	1k	23	19k	2k	128	🌐🤖	✔	✔	✔			🔵🟡🔴	✔
GPT-4-Alpaca	1	55k	7	1	10	1	1k	130	543	🤖	✔					🔴	✔
GPT4AllJ	7	809k	10	1	56	1	<1k	883	1k	🤖	✔					🟡🔴	✔
GPTeacher	4	103k	8	2	33	1	<1k	227	360	🤖	✔					🔵	✔
Gorilla	1	15k	4	2	10	2	<1k	119	76	🤖	✔					🔵	✔
HC3	12	37k	6	2	102	6	2k	119	652	🤖				✔		🔵🟡🔴	✔
Joke Expl.	1	<1k	2	1	10	1	<1k	96	547	🌐	✔					🔵	
LAION OIG	26	9,211k	12	1	171	11	<1k	343	595	🌐🤖					✔	🔵🟡	✔
LIMA	5	1k	10	2	43	6	3k	228	3k	🌐	✔	✔			✔	🔴	
Longform	7	23k	11	1	63	4	3k	810	2k	🤖	✔					🔵🟡	✔
OpAsst OctoPack	1	10k	3	20	10	1	<1k	118	884	🌐					✔	🔵	
OpenAI Summ.	1	93k	5	1	10	1	14k	1k	134	🤖				✔		🔵	✔
OpenAssistant	19	10k	4	20	99	1	14k	118	711	🌐					✔	🔵	
OpenOrca	4	4,234k	11	1	30	23	28k	1k	492	🤖	✔					🔵🔴	✔
SHP	18	349k	6	2	151	1	4k	824	496	🌐				✔		🟡	
Self-Instruct	1	83k	6	2	10	1	3k	134	104	🤖	✔					🔵	✔
ShareGPT	1	77k	9	1	10	2	<1k	303	1k	🤖					✔	🟡	✔
StackExchange	1	10,607k	1	2	10	1	<1k	1k	901	🌐	✔					🟡	
StarCoder	1	<1k	1	2	10	1	<1k	195	504	🤖	✔					🔵	
Tasksource Ins.	288	3,397k	13	1	582	20	<1k	518	18	🌐🤖	✔					🔵🟡🔴	✔
Tasksource ST	229	338k	15	1	477	18	<1k	3k	6	🌐🤖	✔					🔵🟡🔴	✔
TinyStories	1	14k	4	1	10	1	12k	517	194k	🤖	✔					🔵	✔
Tool-Llama	1	37k	2	2	10	1	-	7k	1k	🤖					✔	🔴	✔
UltraChat	1	1,468k	7	1	11	2	2k	282	1k	🤖	✔				✔	🔴	✔

Example: Data Provenance Initiative

Can make many interesting observations, on creators...

- Most data come from academic, then industry labs
- Also from Wikimedia, then social media

Name	Pct
Academic	**68.7%**
University of Washington	8.9%
Stanford University	6.8%
New York University	5.4%
University of Southern...	3.5%
Carnegie Mellon Univer...	3.5%
Saarland University	2.6%
Cardiff University	2.3%
Industry Lab	**21.4%**
Facebook AI Research	8.4%
Microsoft Research	4.1%
Google Research	2.9%
DeepMind	1.9%
Microsoft Semantic Mac...	0.9%
NAVER AI Lab	0.8%
Salesforce Research	0.7%
Research Group	**17.1%**
AI2	12.3%
CLUE team	0.5%
Alan Turing Institute	0.5%
CodeX	0.4%
Qatar Computing Resear...	0.4%
Barcelona Supercomputi...	0.4%
BigCode	0.2%
Corporation	**15.8%**
Google	2.1%
IBM	2.0%
Microsoft	1.4%
Wind Information Co.	1.4%
Snap Inc.	1.3%
Meta	1.1%

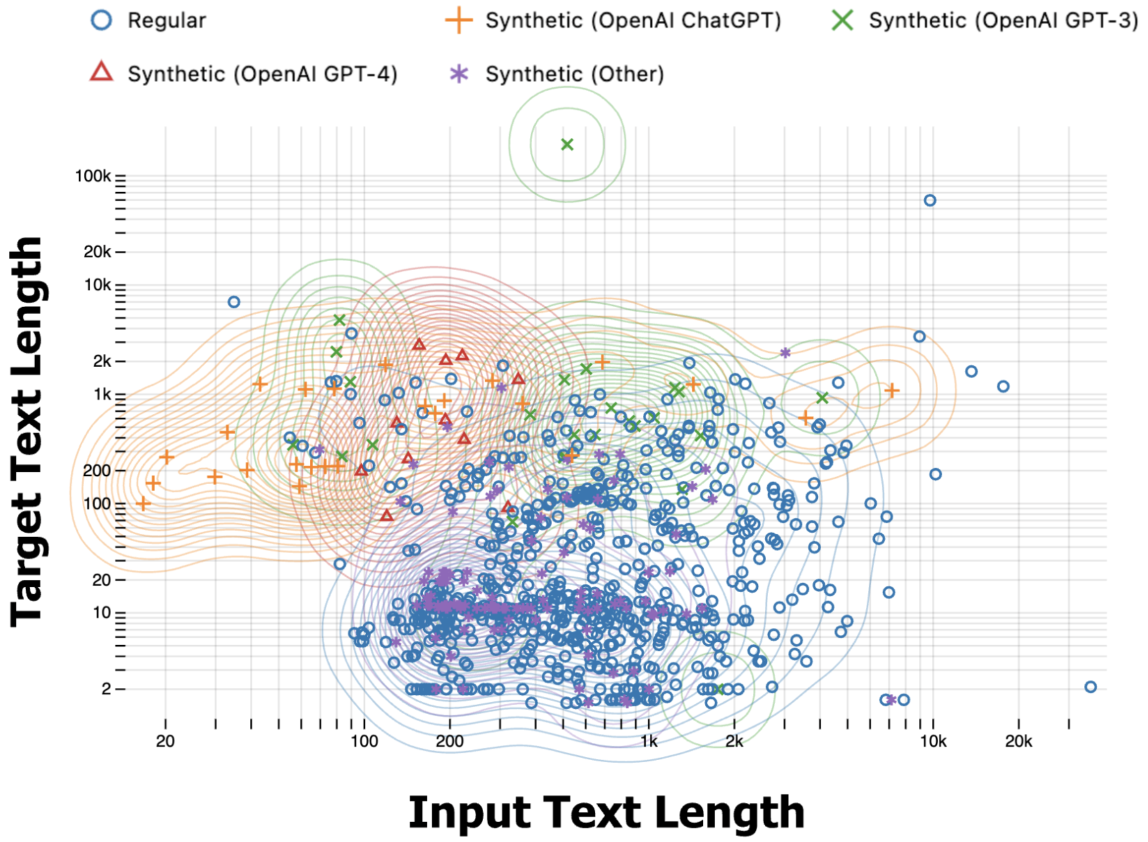
Name	Pct
Question Answering	**36.0%**
Question Answering	27.7%
Multiple Choice Questi...	3.9%
Information Extraction	1.8%
Text Classification	**29.9%**
Text Classification	16.1%
Sentiment Analysis	9.8%
Named Entity Recognition	4.3%
Natural Language Inf...	**21.1%**
Textual Entailment	14.6%
Natural Language Infer...	5.3%
Fact Verification	1.3%
Open-form Text Gener...	**11.3%**
Open-form Text Generation	2.2%
Title Generation	1.5%
Inverted Summarization	1.2%
Short Text Generation	**10.9%**
Question Generation	4.0%
Fill in The Blank	1.4%
Inverted Multiple-Choi...	0.9%
Dialog Generation	**9.0%**
Dialogue Generation	4.2%
Dialog Generation	3.7%
Dialogue Act Recognition	0.4%
Summarization	**6.3%**
Summarization	5.7%
Simplification	0.5%
Summarization of US Co...	0.1%

Name	Pct
Encyclopedias	**21.5%**
wikipedia.org	14.6%
wikihow.com	2.7%
dbpedia	1.4%
Social Media	**15.9%**
reddit	6.2%
twitter	4.0%
quora	1.6%
General Web	**11.2%**
undisclosed web	7.0%
commoncrawl.org	2.5%
data.world/samayo/coun...	0.6%
News	**11.1%**
cnn.com	1.6%
financial news	1.5%
press releases	1.4%
Entertainment	**8.5%**
opensubtitles.org	2.5%
imdb.com	1.6%
travel guides	1.3%
Code	**5.7%**
stackexchange.com	2.0%
github	1.2%
opus software projects	0.9%
Exams	**5.6%**
web exams	2.9%
gmat	1.1%
gre exams	0.9%

Example: Data Provenance Initiative

Can make many interesting observations, on data source…

- More synthesized datasets
- Very different distribution



Longpre et al. "A large-scale audit of dataset licensing and attribution in AI." Nature Machine Intelligence 2024

Example: Data Provenance Initiative

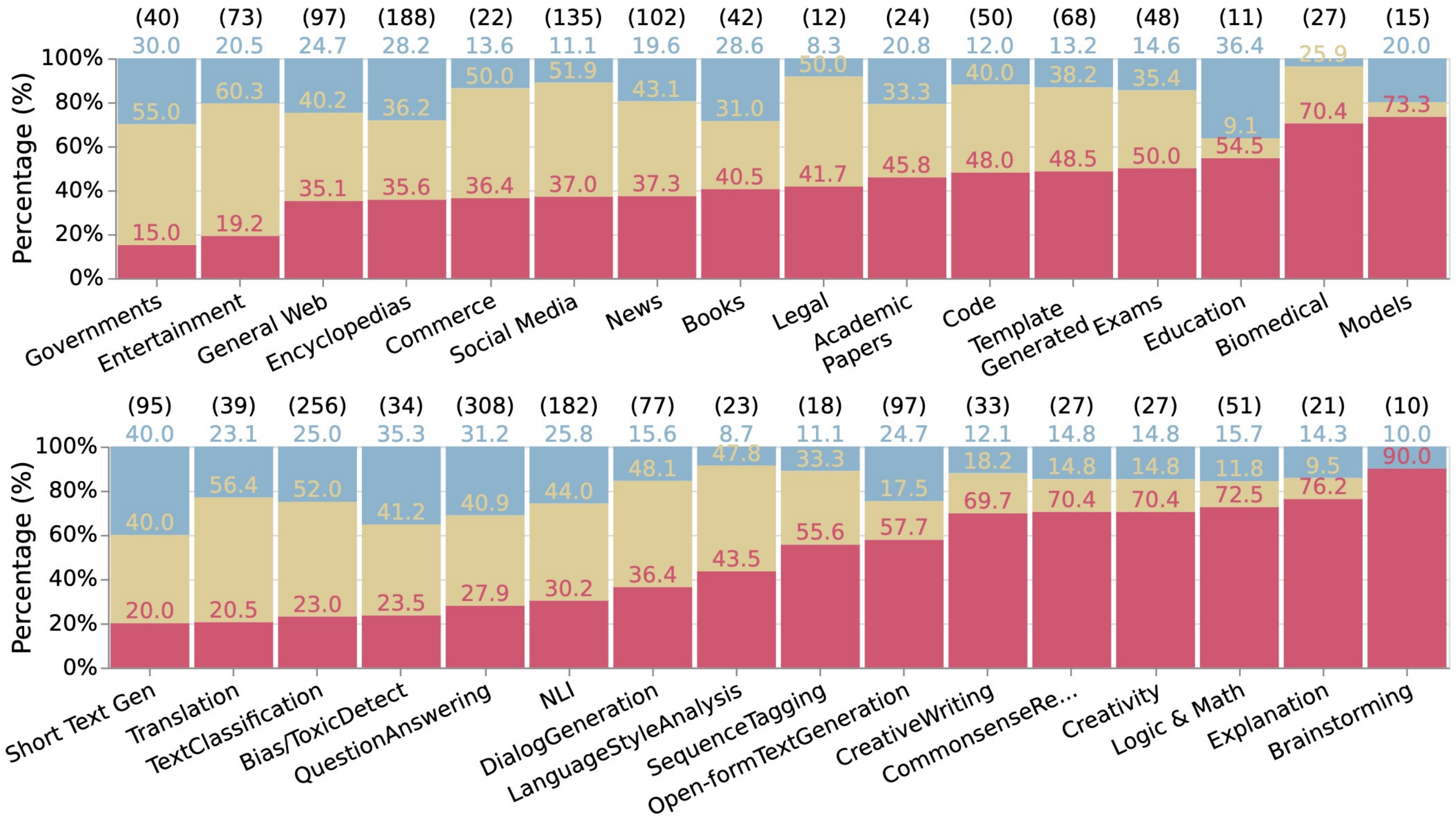
Observe issues in data provenance: Many datasets are annotated to have more permissive licenses than they actually have!

Correct license		License according to aggregators (Agg.)				
License	Count	Agg.	Comm.	Unspec.	Non-Comm.	Acad.-Only
Commercial	856 (46.1%)		349	507	0	0
			176	677	1	2
			313	520	1	22
Unspecified	570 (30.7%)		112	458	0	0
			164	395	6	5
			31	523	1	15
Non-Commercial	352 (19.0%)		49	303	0	0
			113	152	80	7
			2	191	157	2
Academic-Only	80 (4.3%)		9	71	0	0
			9	65	2	4
			5	65	2	8
Total	1858 (100%)		519 (28%)	1339 (72%)	0 (0%)	0 (0%)
			462 (25%)	1289 (69%)	89 (5%)	18 (1%)
			351 (19%)	1299 (70%)	161 (9%)	47 (3%)

Longpre et al. "A large-scale audit of dataset licensing and attribution in AI." Nature Machine Intelligence 2024

Example: Data Provenance Initiative

Make interesting observations on what you can and cannot train models on (blue is commerially usable, red is not)



Longpre et al. "A large-scale audit of dataset licensing and attribution in AI." Nature Machine Intelligence 2024

Feature Provenance

How are features extracted from raw data?

- during training
- during inference

Has feature extraction changed since the model was trained?

Recommendation: Modularize and version feature extraction code

Example?

DVC Example

```yaml
stages:
  features:
    cmd: jupyter nbconvert --execute featurize.ipynb
    deps:
      - data/clean
    params:
      - levels.no
    outs:
      - features
    metrics:
      - performance.json
```

Advanced Practice: Feature Store

Stores feature extraction code as functions, versioned

Catalog features to encourage reuse

Compute and cache features centrally

Use same feature used in training and inference code

Advanced: Immutable features -- never change existing features, just add new ones (e.g., creditscore, creditscore2, creditscore3)

Model Provenance

How was the model trained?

What data? What library? What hyperparameter? What code?

Ensemble of multiple models?

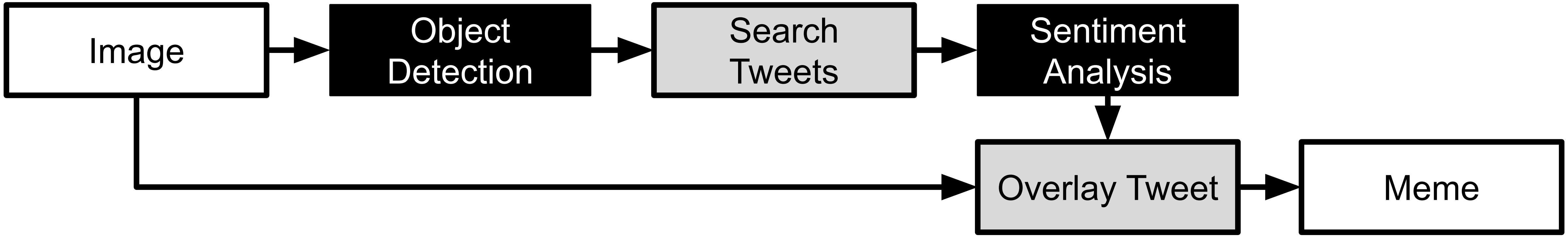
Unified Model Record (UMR)

UMR tracks the entire lifecycle of a model, including design, development, testing, and deployment phases.

Feature	Model Card	Unified Model Record (UMR)
Purpose	Summarizes model's purpose, limitations, and usage	Tracks full model lifecycle and operations
Content Focus	Usage guidelines, ethical considerations, limitations	Technical specs, lifecycle history, compliance records
Compliance	Limited focus on regulatory compliance	Aligned with governance and regulatory standards
Maintenance	Updated occasionally, typically per major update	Continuous updates with lifecycle events and audits

https://modelrecord.com/

In Real Systems: Tracking Provenance Across Multiple Models



Version all models involved!

Example adapted from Jon Peck. [Chaining machine learning models in production with Algorithmia](). Algorithmia blog, 2019

Complex Model Composition: ML Models for Feature Extraction

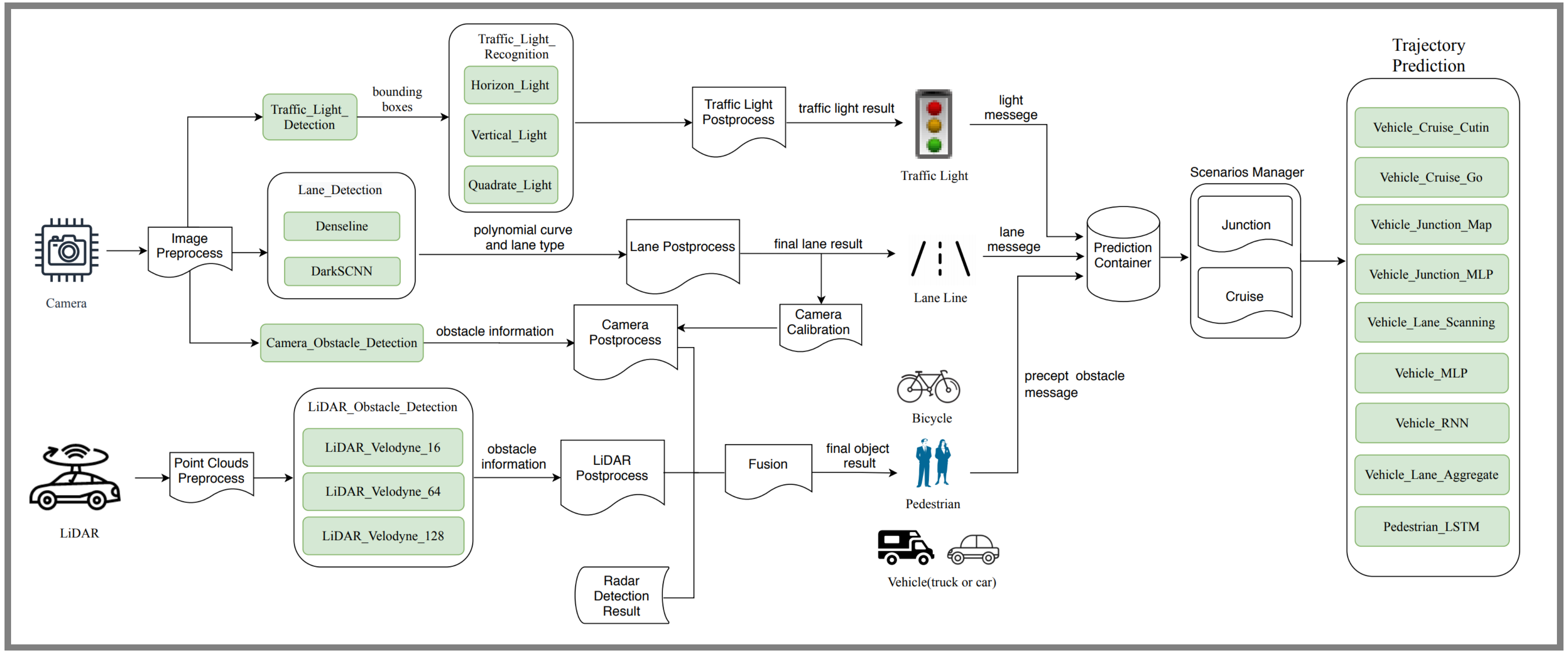
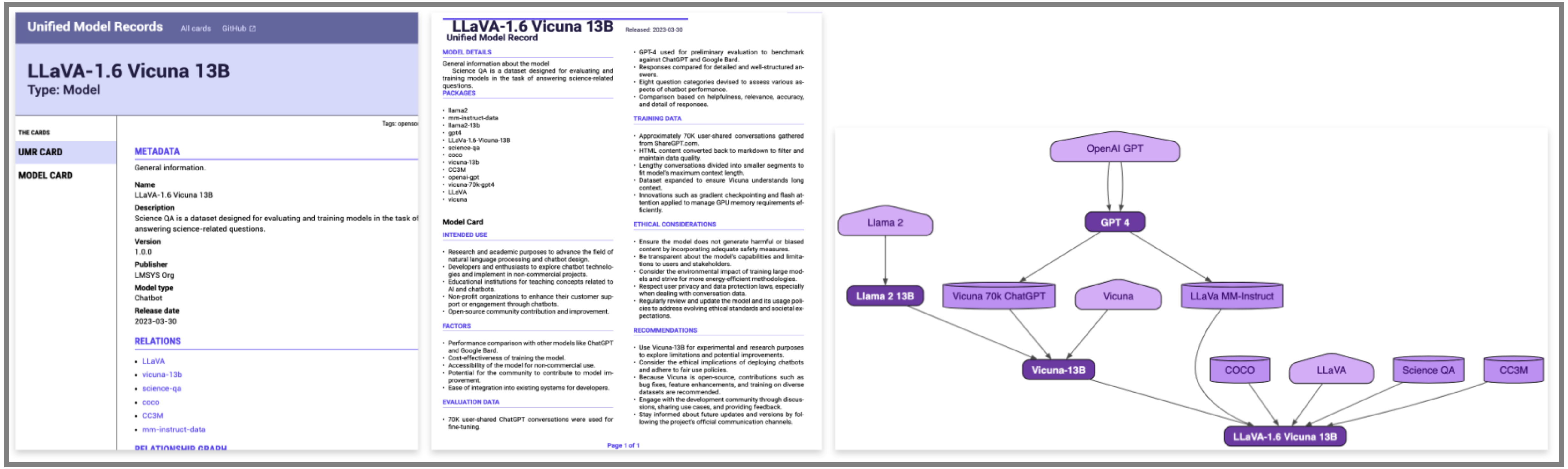


Image: Peng, Zi, Jinqiu Yang, Tse-Hsun Chen, and Lei Ma. "A first look at the integration of machine learning models in complex autonomous driving systems: a case study on Apollo." In Proc. FSE. 2020.

How about Provenance for Foundation Models?

Complicated Foundation Models



Wang, Keyu, et al. "Mitigating Downstream Model Risks via Model Provenance." arXiv 2024

Summary: Provenance

Data provenance

Feature provenance

Model provenance

Breakout Discussion: Movie Predictions (Revisited)

> *Assume you are receiving complains that a child gets mostly recommendations about R-rated movies*

Discuss again, updating the previous post in `#lecture`:

- How would you identify the model that caused the prediction?
- How would you identify the code and dependencies that trained the model?
- How would you identify the training data used for that model?

Reproducability

On Terminology

Replicability: ability to reproduce results exactly

- Ensures everything is clear and documented
- All data, infrastructure shared; requires determinism

Reproducibility: the ability of an experiment to be repeated with minor differences, achieving a consistent expected result

- In science, reproducing important to gain confidence
- many different forms distinguished: conceptual, close, direct, exact, independent, literal, nonexperiemental, partial, retest, ...

Juristo, Natalia, and Omar S. Gómez. "Replication of software engineering experiments." In Empirical software engineering and verification, pp. 60-88. Springer, Berlin, Heidelberg, 2010.

"Reproducibility" of Notebooks

2019 Study of 1.4M notebooks on GitHub:

- 21% had unexecuted cells
- 36% executed cells out of order
- 14% declare dependencies
- success rate for installing dependencies <40% (version issues, missing files)
- notebook execution failed with exception in >40% (often ImportError, NameError, FileNotFoundError)
- only 24% finished execution without problem, of those 75% produced different results

2020 Study of 936 executable notebooks:

- 40% produce different results due to nondeterminism (randomness without seed)
- 12% due to time and date
- 51% due to plots (different library version, API misuse)
- 2% external inputs (e.g. Weather API)
- 27% execution environment (e.g., Python package versions)

Pimentel, João Felipe, et al. "A large-scale study about quality and reproducibility of jupyter notebooks." In Proc. MSR, 2019. and Wang, Jiawei, K. U. O. Tzu-Yang, Li Li, and Andreas Zeller.

Practical Reproducibility

Ability to generate the same research results or predictions

Recreate model from data

Requires versioning of data and pipeline (incl. hyperparameters and dependencies)

Nondeterminism

- Model inference almost always deterministic for a given model
- Many machine learning algorithms are nondeterministic
  - Nondeterminism in neural networks initialized from random initial weights
  - Nondeterminism from distributed computing, random forests
  - Determinism in linear regression and decision trees
- Many notebooks and pipelines contain nondeterminism
  - Depend on time or snapshot of online data (e.g., stream)
  - Initialize random seed
  - Different memory addresses for figures
- Different library versions installed on the machine

Recommendations for Reproducibility

- Version pipeline and data (see above)
- Document each step
  - document intention and assumptions of the process (not just results)
  - e.g., document why data is cleaned a certain way
  - e.g., document why certain parameters chosen
- Ensure determinism of pipeline steps (-> test)
- Modularize and test the pipeline
- Containerize infrastructure -- see MLOps

Summary

Provenance is important for debugging and accountability

Data provenance, feature provenance, model provenance

Reproducibility vs replicability

Version everything!

- Strategies for data versioning at scale
- Version the entire pipeline and dependencies
- Adopt a pipeline view, modularize, automate
- Containers and MLOps, many tools

Further Readings

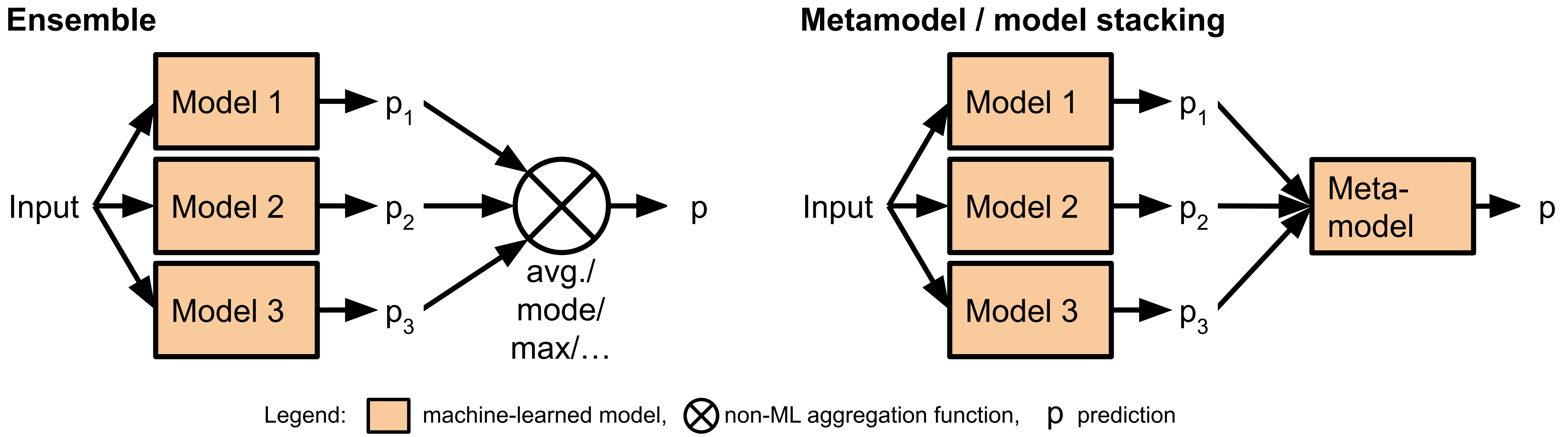
- Sugimura, Peter, and Florian Hartl. "Building a Reproducible Machine Learning Pipeline." *arXiv preprint arXiv:1810.04570* (2018).
- Chattopadhyay, Souti, Ishita Prasad, Austin Z. Henley, Anita Sarma, and Titus Barik. "What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities." In Proceedings of the CHI Conference on Human Factors in Computing Systems, 2020.
- Sculley, D, et al. "Hidden technical debt in machine learning systems." In Advances in neural information processing systems, pp. 2503–2511. 2015.

Bonus: Debugging and Fixing Models

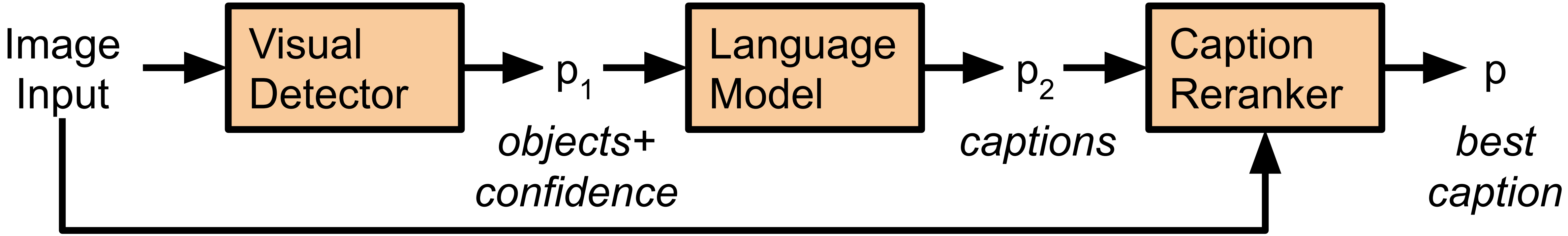
See also Hulten. Building Intelligent Systems. Chapter 21

See also Nushi, Besmira, Ece Kamar, Eric Horvitz, and Donald Kossmann. "On human intellect and machine failures: troubleshooting integrative machine learning systems." In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 1017-1025. 2017.

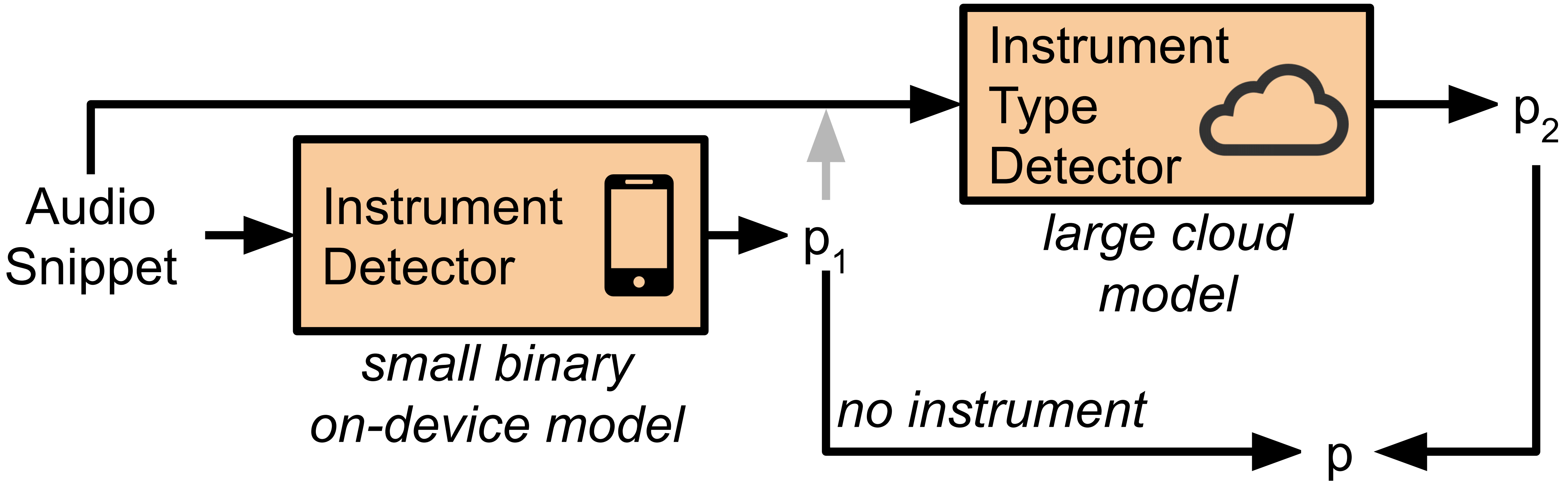
Recall: Composing Models: Ensemble and metamodels



Recall: Composing Models: Decomposing the problem, sequential



Recall: Composing Models: Cascade/two-phase prediction

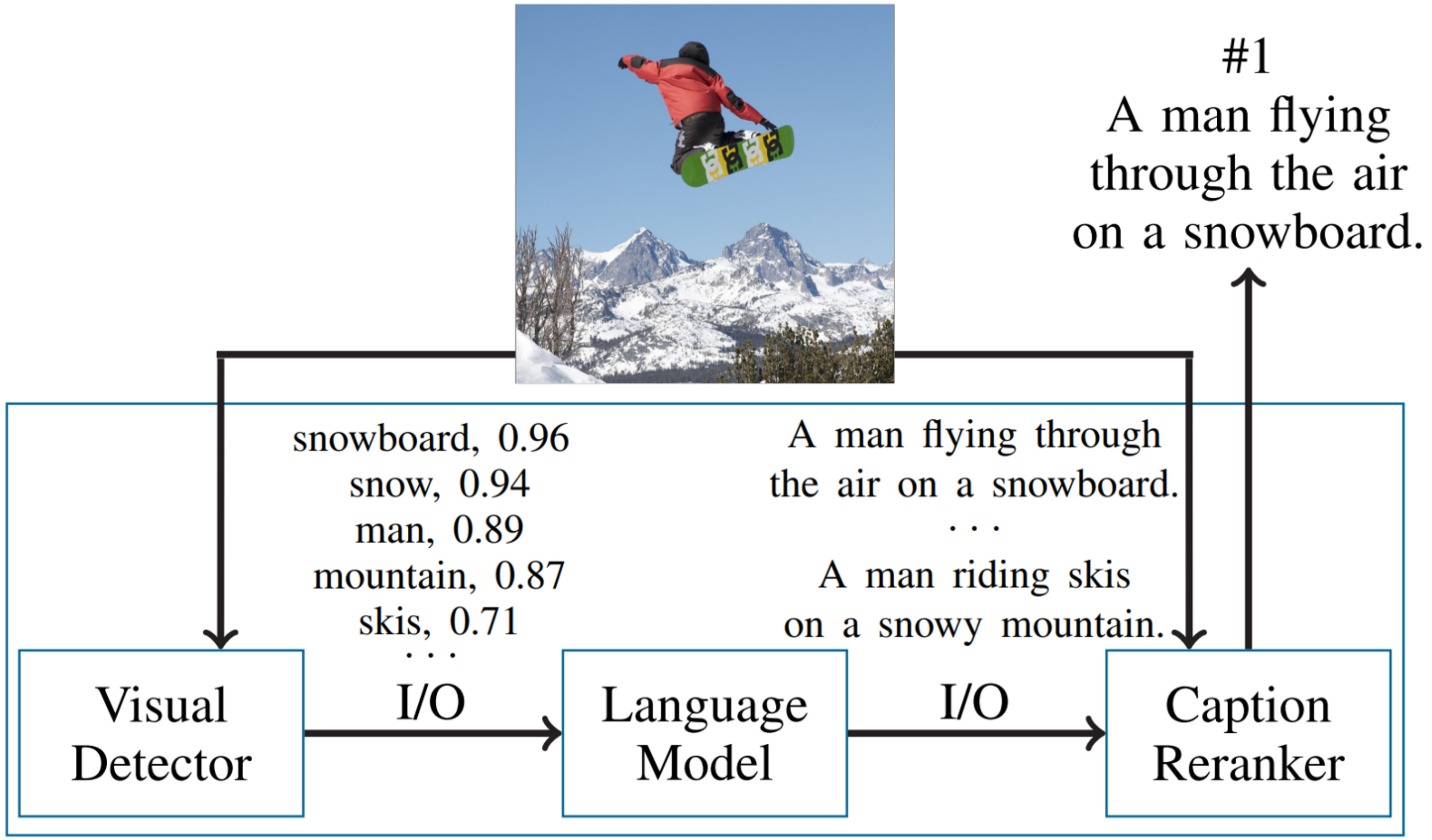


Decomposing the Image Captioning Problem?

Speaker notes

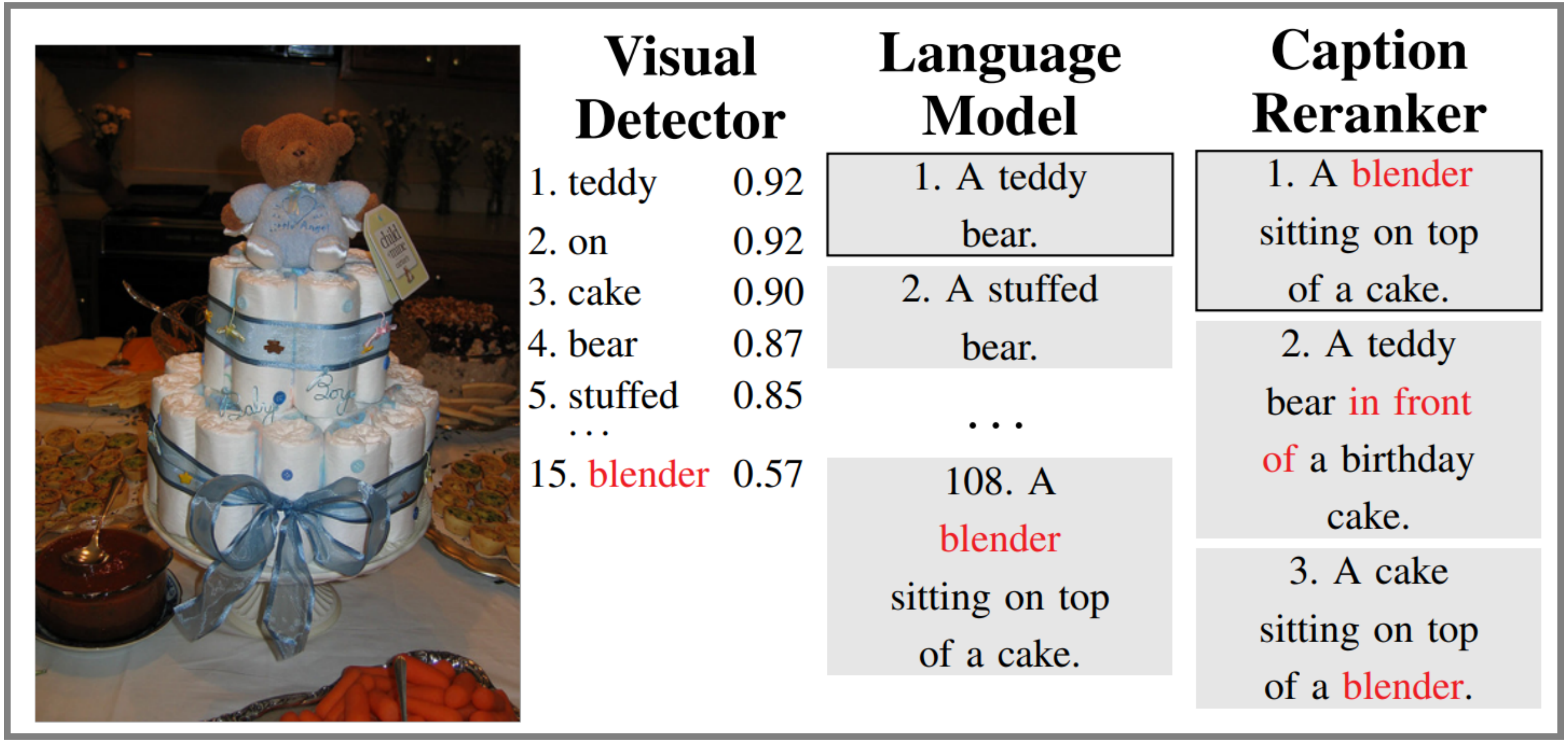
Using insights of how humans reason: Captions contain important objects in the image and their relations. Captions follow typical language/grammatical structure

State of the Art Decomposition (in 2015)



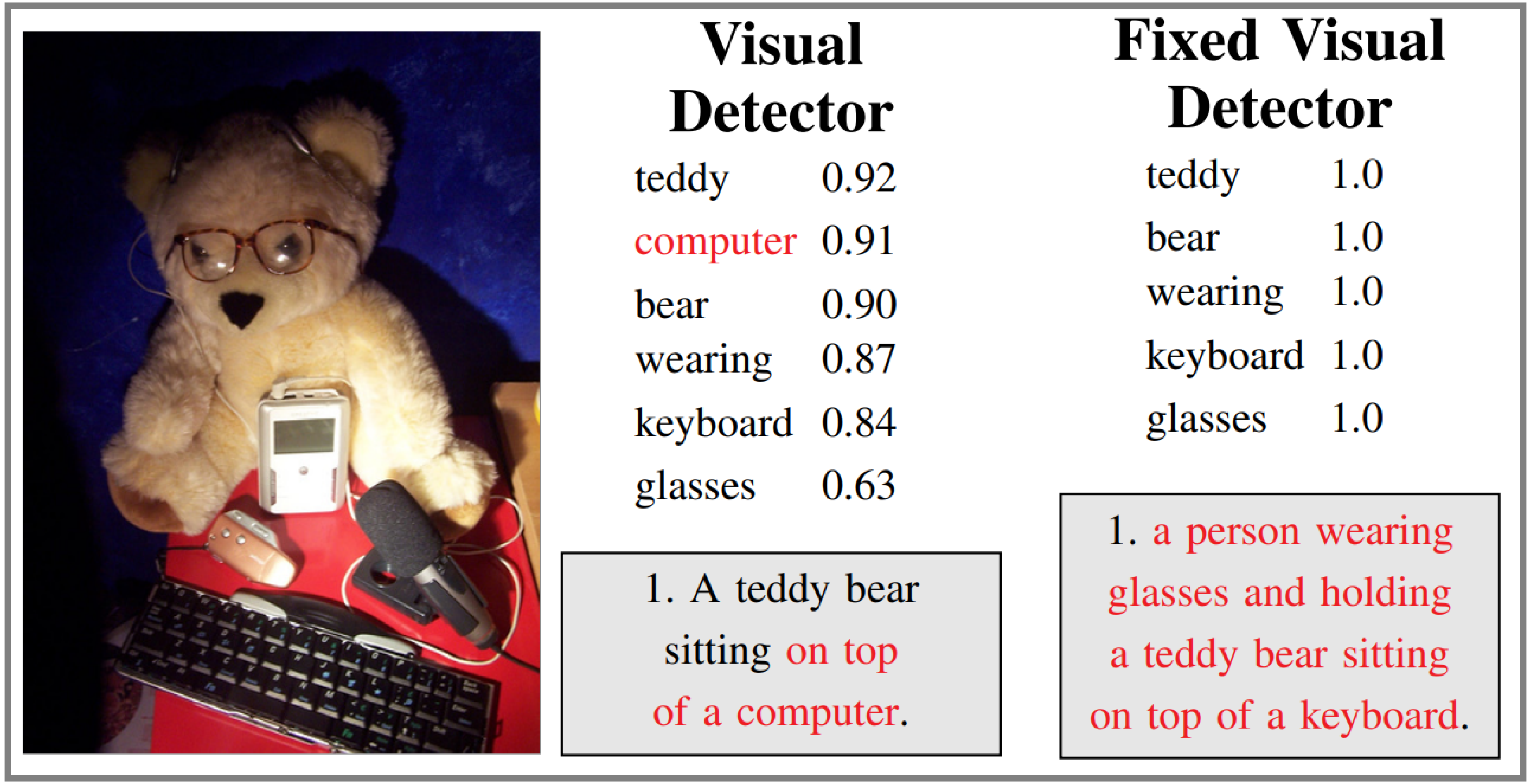
Example and image from: Nushi, Besmira, Ece Kamar, Eric Horvitz, and Donald Kossmann. "On human intellect and machine failures: troubleshooting integrative machine learning systems." In Proc. AAAI. 2017.

Blame assignment?



Example and image from: Nushi, Besmira, Ece Kamar, Eric Horvitz, and Donald Kossmann. "On human intellect and machine failures: troubleshooting integrative machine learning systems." In Proc. AAAI. 2017.

Nonmonotonic errors



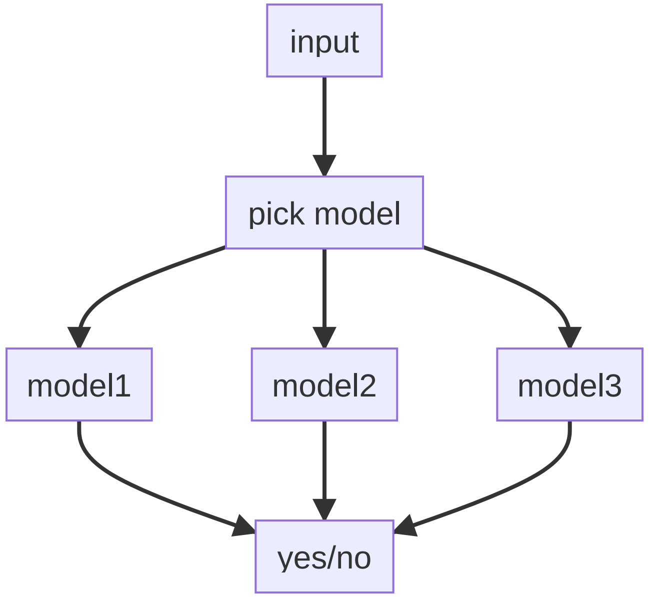
Example and image from: Nushi, Besmira, Ece Kamar, Eric Horvitz, and Donald Kossmann. "On human intellect and machine failures: troubleshooting integrative machine learning systems." In Proc. AAAI. 2017.

Chasing Bugs

- Update, clean, add, remove data
- Change modeling parameters
- Add regression tests
- Fixing one problem may lead to others, recognizable only later

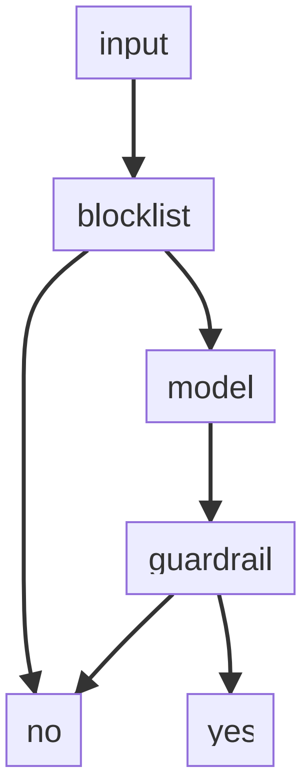
Partitioning Contexts

- Separate models for different subpopulations
- Potentially used to address fairness issues
- ML approaches typically partition internally already



Overrides

- Hardcoded heuristics (usually created and maintained by humans) for special cases
- Blocklists, guardrails
- Potential neverending attempt to fix special cases



Ideas?