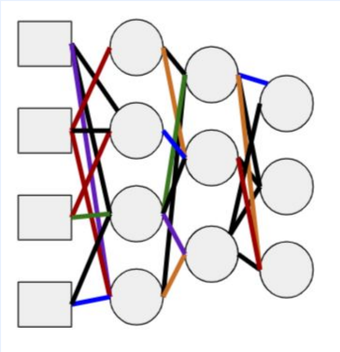
Google

Reliable Data for Large ML Models

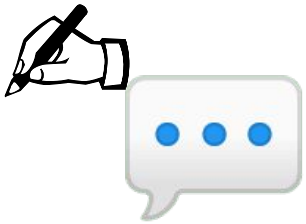
Mary McGlohon
marymc@google.com



Thanks to: Alex Lince, Dylan Curley, Todd Underwood, Herve Quiroz, Igor Karpov

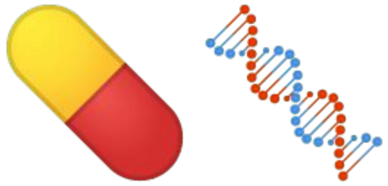
Why this matters

ML is doing more and more sophisticated things, *and* becoming more accessible.



language summarization and generation

code generation



drug discovery

Why this matters

The **emergent** behavior of ML systems means that the **inputs** matter more.

It’s not just that your outputs depend on the inputs– it’s that the outputs also become a part of what you are executing

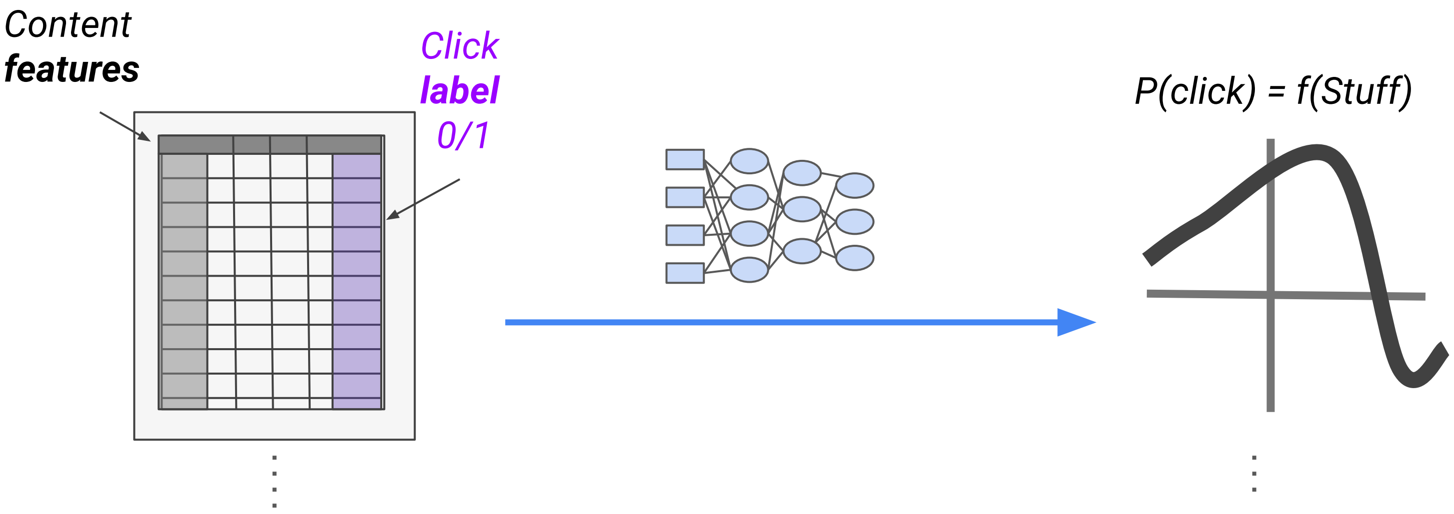
…and this can impact things in ways you don’t expect.

What we will talk about here

- “Classic” (supervised) ML models and their data problems
- LLMs and their data problems
- How to help prevent / mitigate bad data issues

Supervised ML and risks

Supervised ML

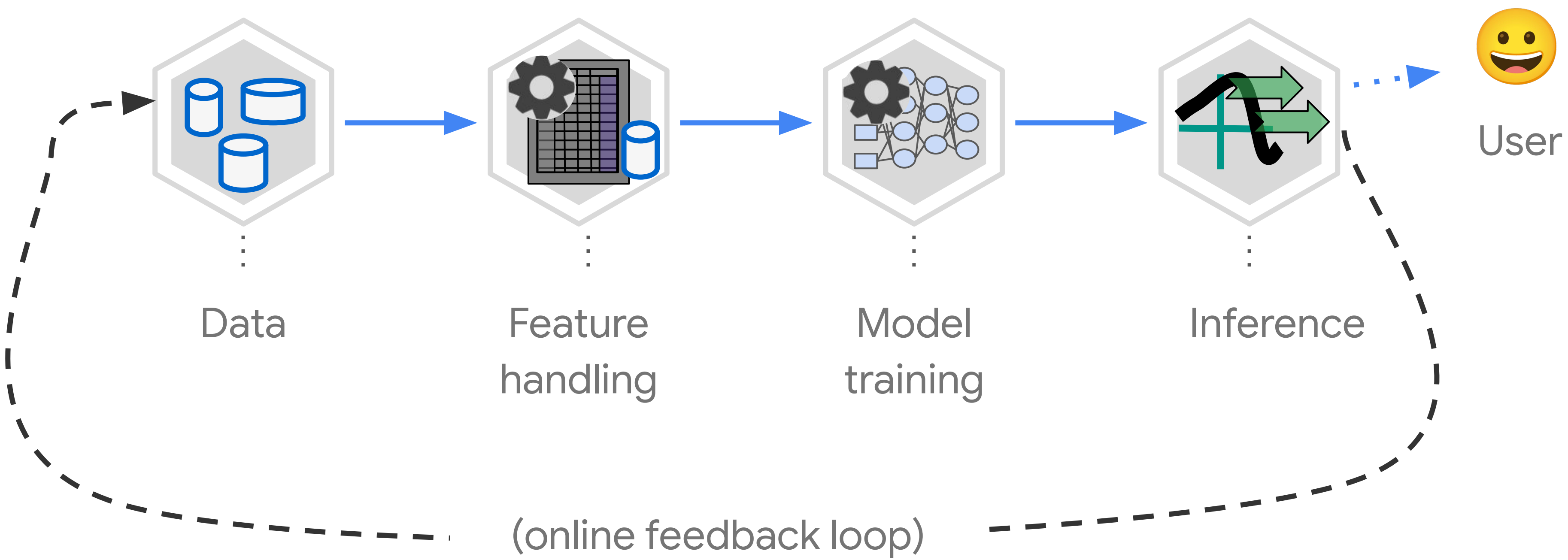


Labeled data
(Features)

Neural network
(or other linear
algebra)

Model

Supervised ML in production



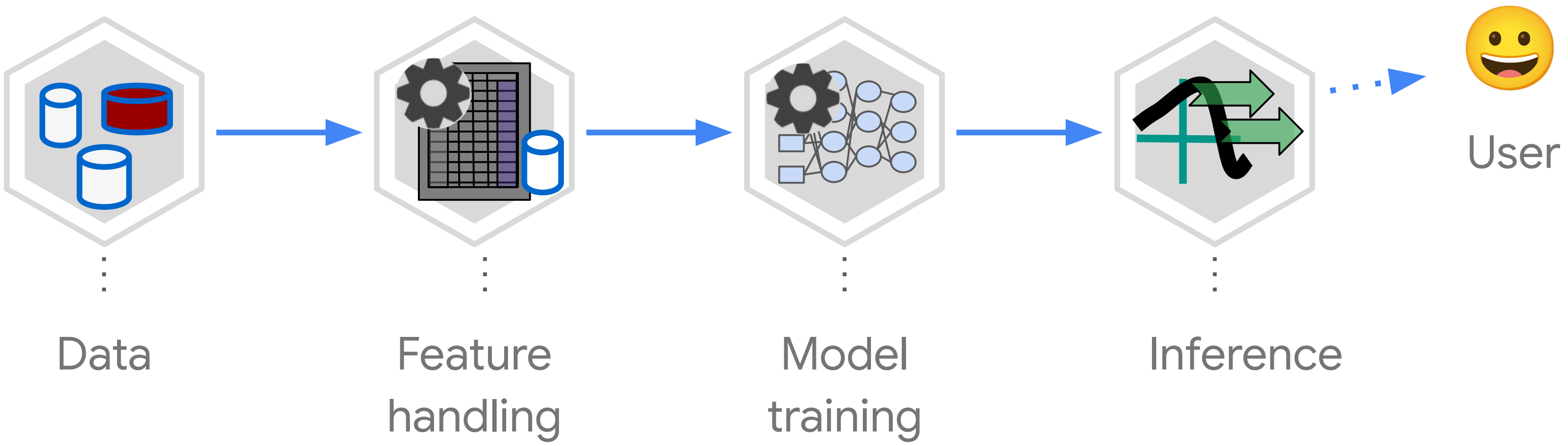
Supervised ML data risks

Your input data may be “bad” for a number of reasons. For example:

- Incomplete or missing
- Mis-labeled
- Biased
- Corrupted
- Later deemed unacceptable to use

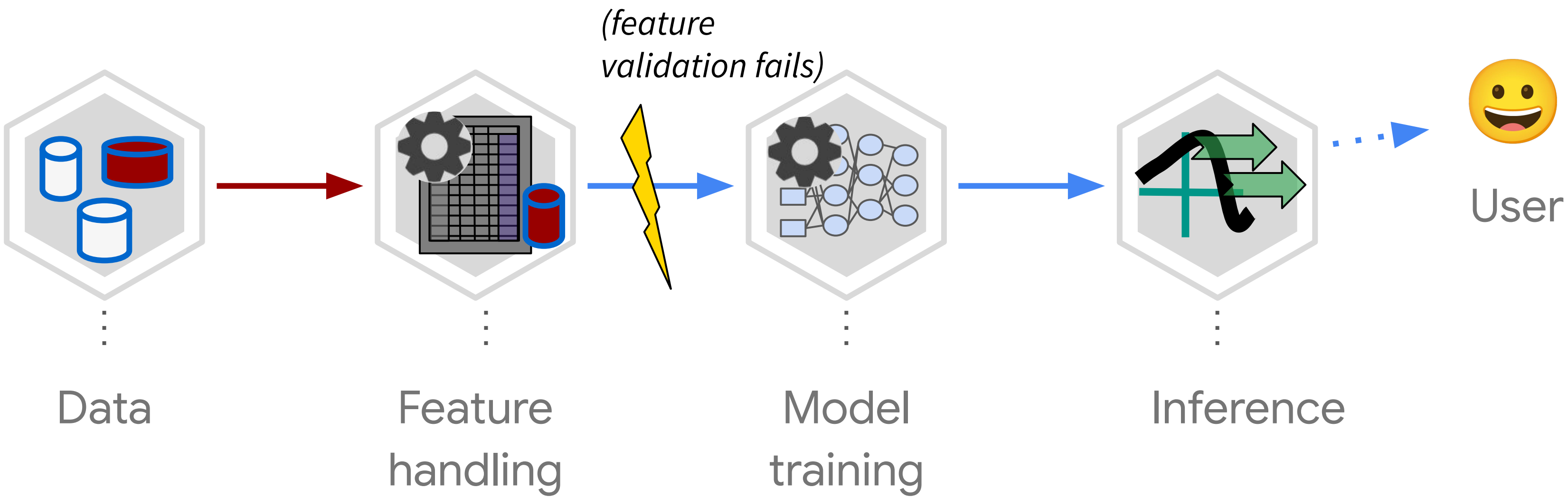
ML data outage impact

What happens if you have “bad” data?



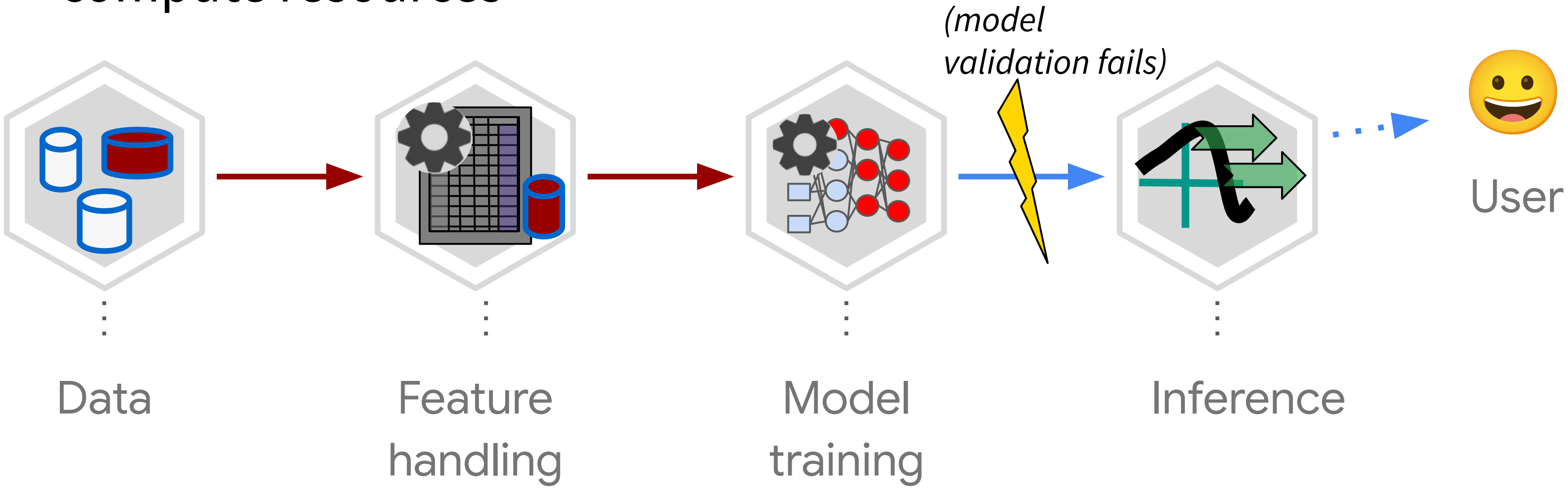
ML data outage impact

Bad outcome: Training is delayed



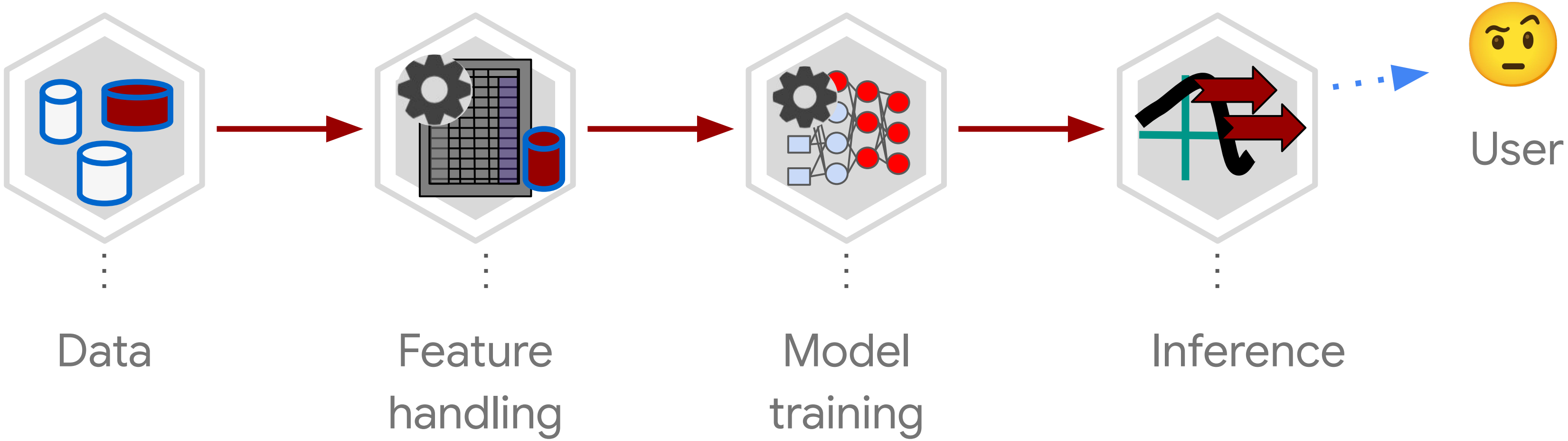
ML data outage impact

Worse outcome: Training is bad, and you wasted a **lot** of compute resources



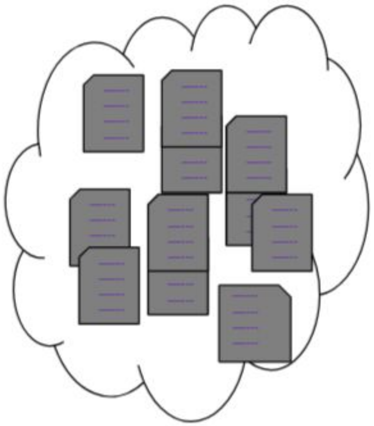
ML data outage impact

Worst outcome: Badly-trained models pushed to prod -> your predictions are bad, and everyone knows it

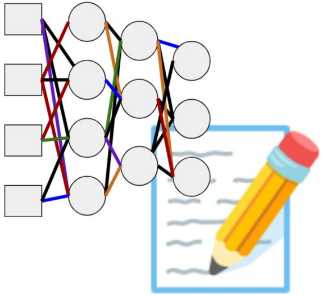


Large language models and risks

Language model (pre-training)

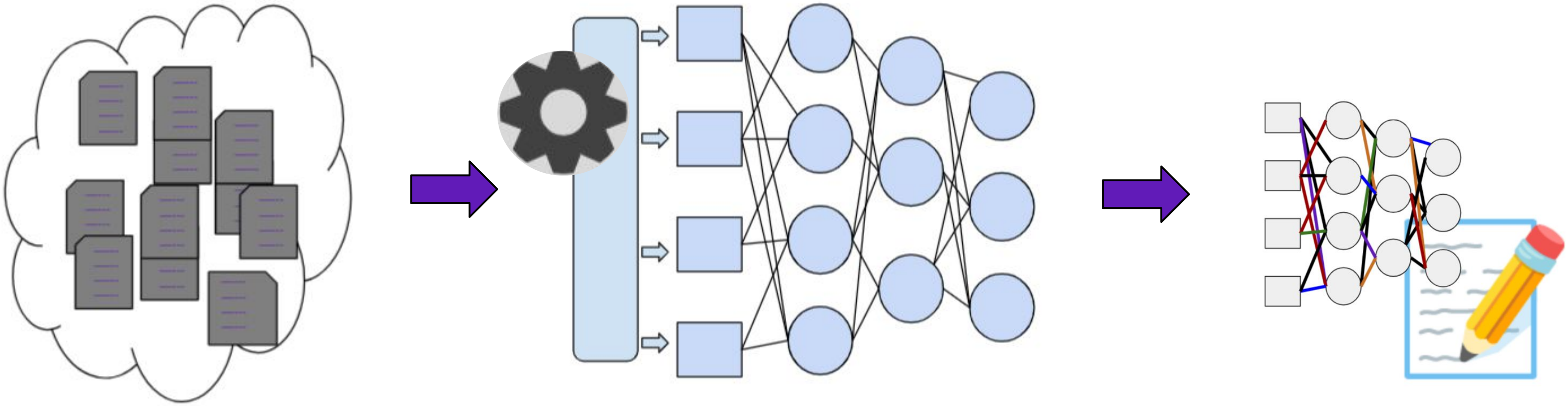


Text corpus
(Unlabeled)



Inference
(predict
next word)

Language model (pre-training)

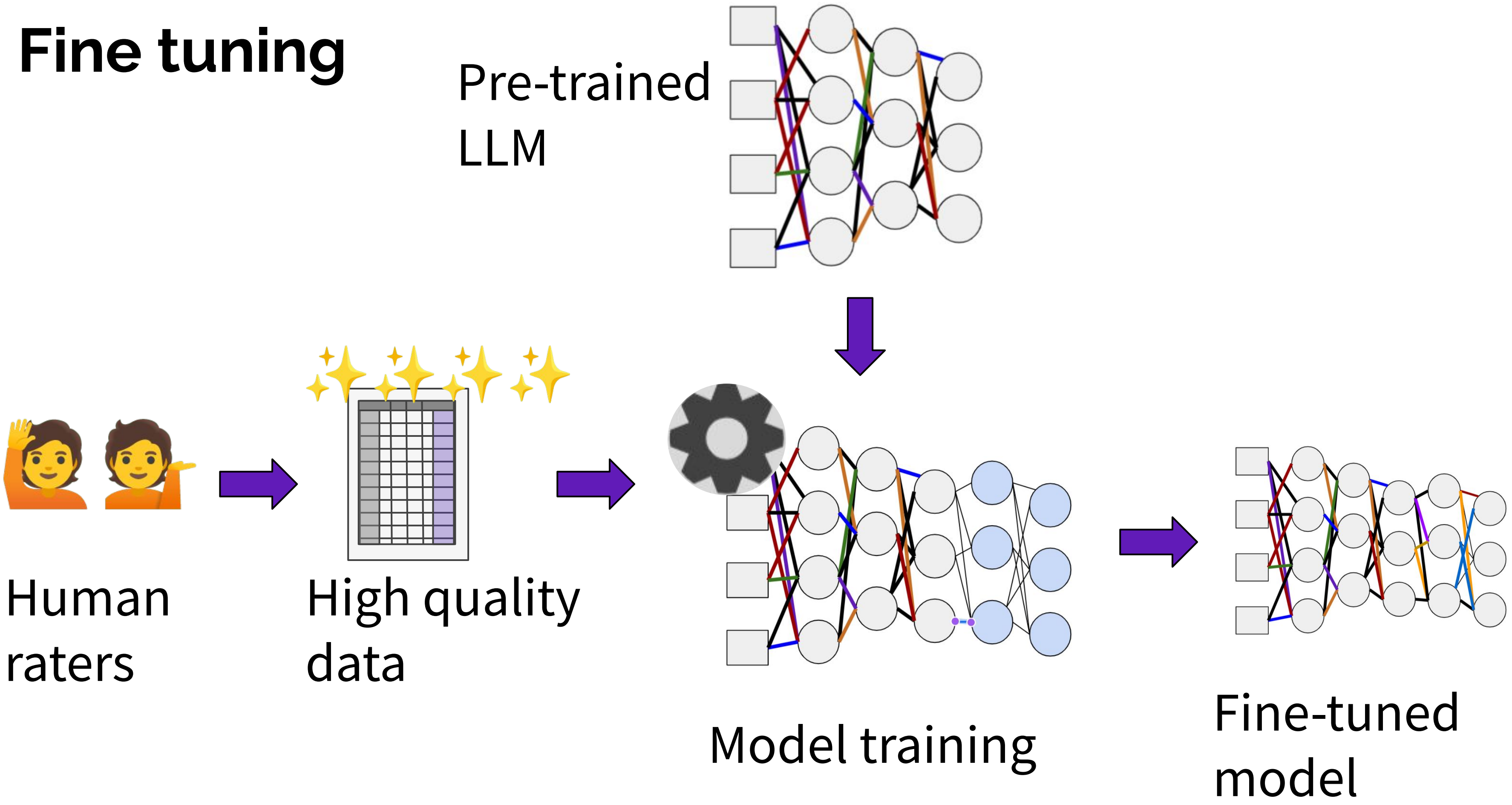


Text corpus
(Unlabeled)

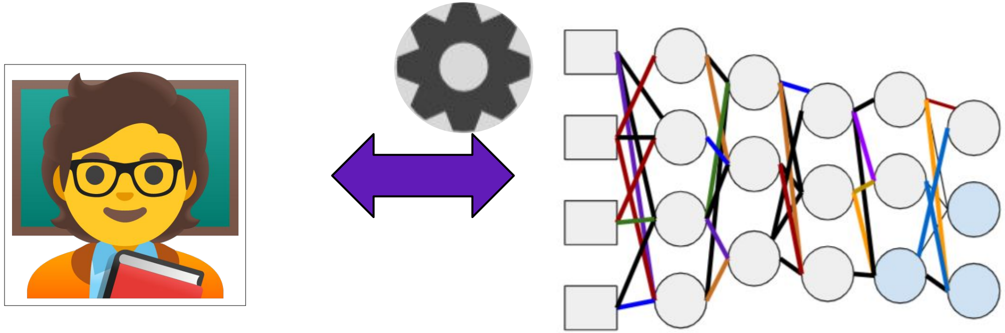
Model training
(high compute
and I/O cost)

Inference
(predict
next word)

Fine tuning

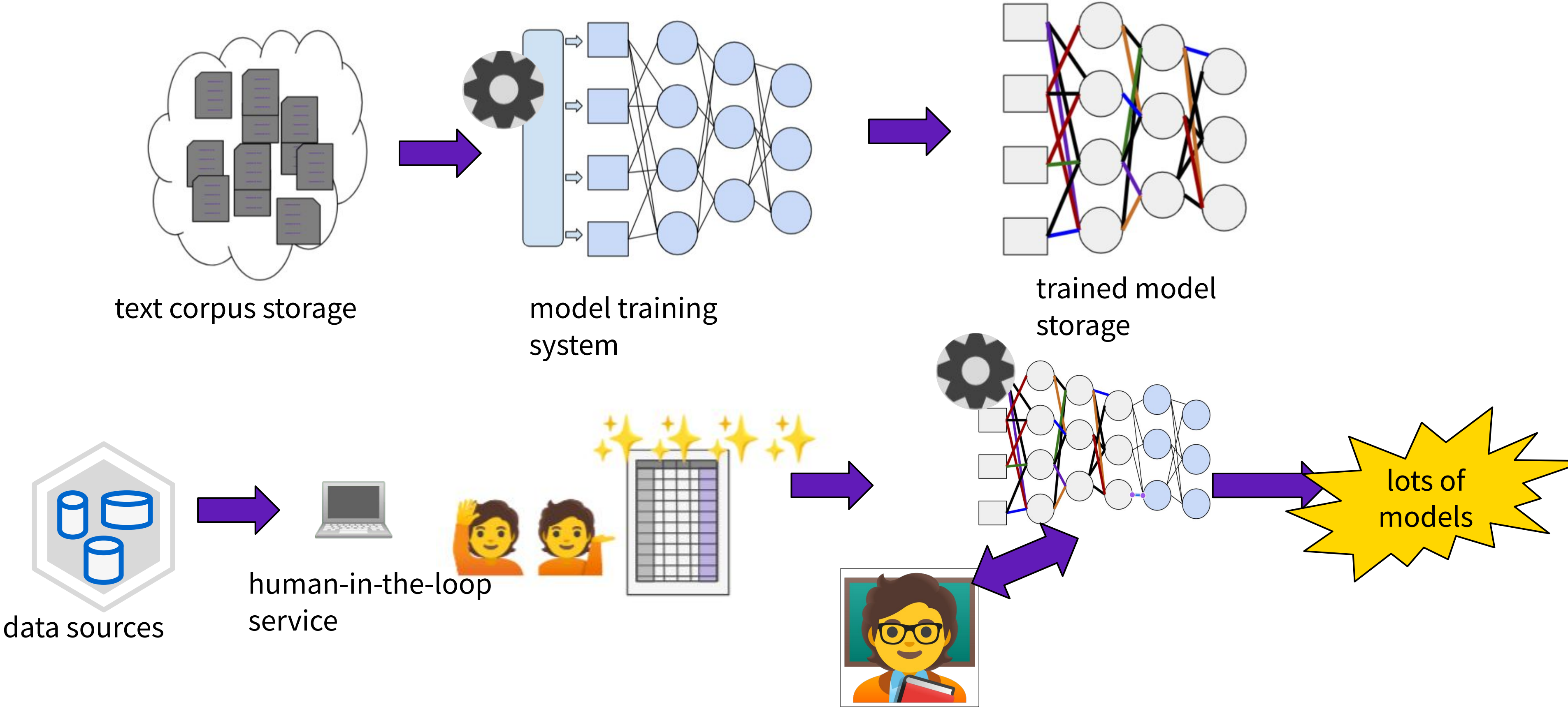

Pre-trained LLM
Human raters
High quality data
Model training
Fine-tuned model

Fine tuning: Reinforcement Learning from Human Feedback (RLHF)

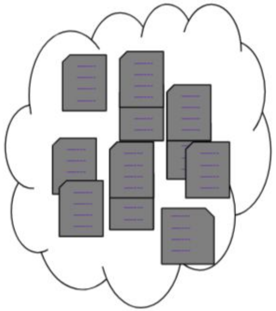


(...and there are other ways for refining the pre-trained model)

LLMs in production



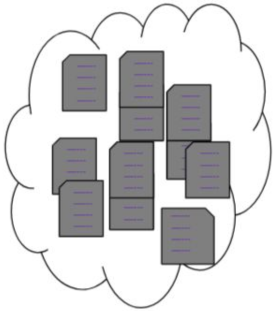
Data risks: pre-training



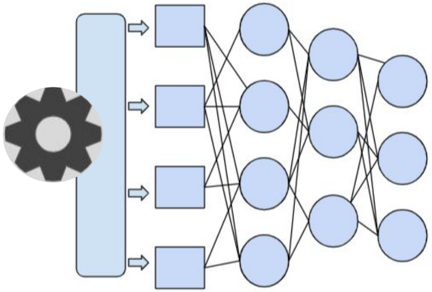
text corpus storage

- Unstructured, dynamic data
- Inherent bias
- Unknown origin (human vs AI generated)
- Too large to multi-home
- Usage requirements (privacy, governance, etc)

Data risks: pre-training



text corpus storage

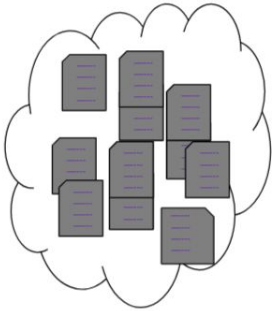


model training system

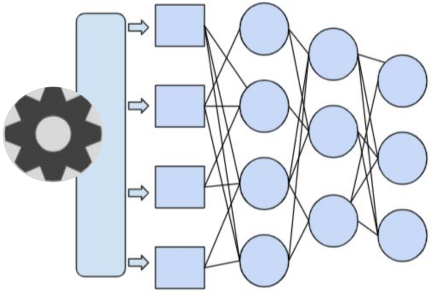
- Unstructured, dynamic data
- Inherent bias
- Unknown origin (human vs AI generated)
- Too large to multi-home
- Usage requirements (privacy, governance, etc)

- High bandwidth required to train
- Very high resource wastage if training is wrong
- Data accidentally dropped

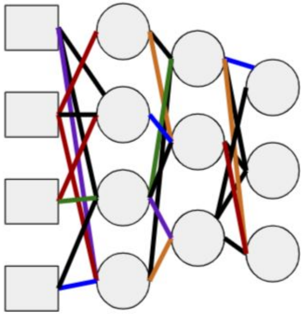
Data risks: pre-training



text corpus storage



model training system



trained model storage

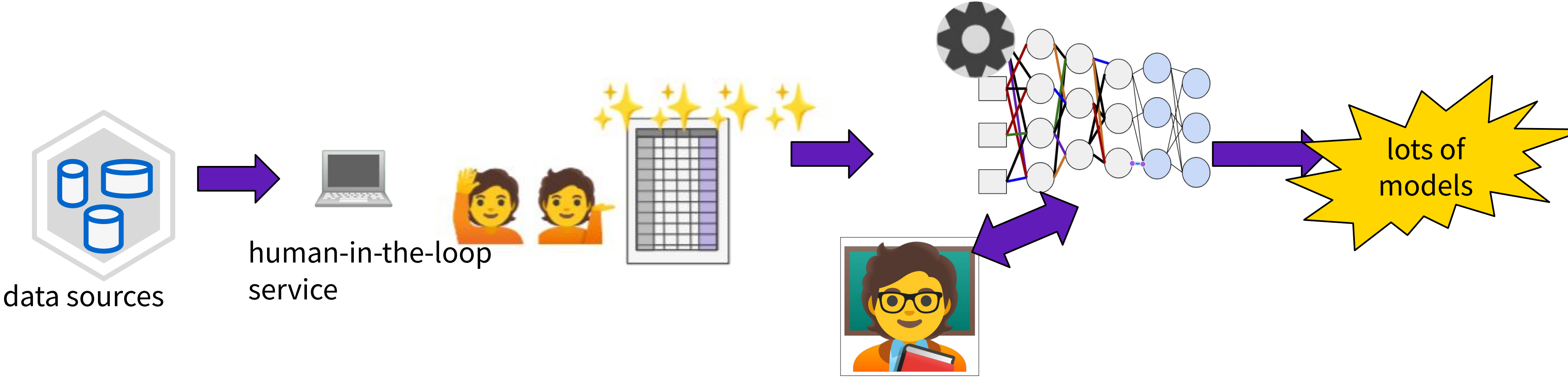
- Unstructured, dynamic data
- Inherent bias
- Unknown origin (human vs AI generated)
- Too large to multi-home
- Usage requirements (privacy, governance, etc)

- High bandwidth required to train
- Very high resource wastage if training is wrong
- Data accidentally dropped

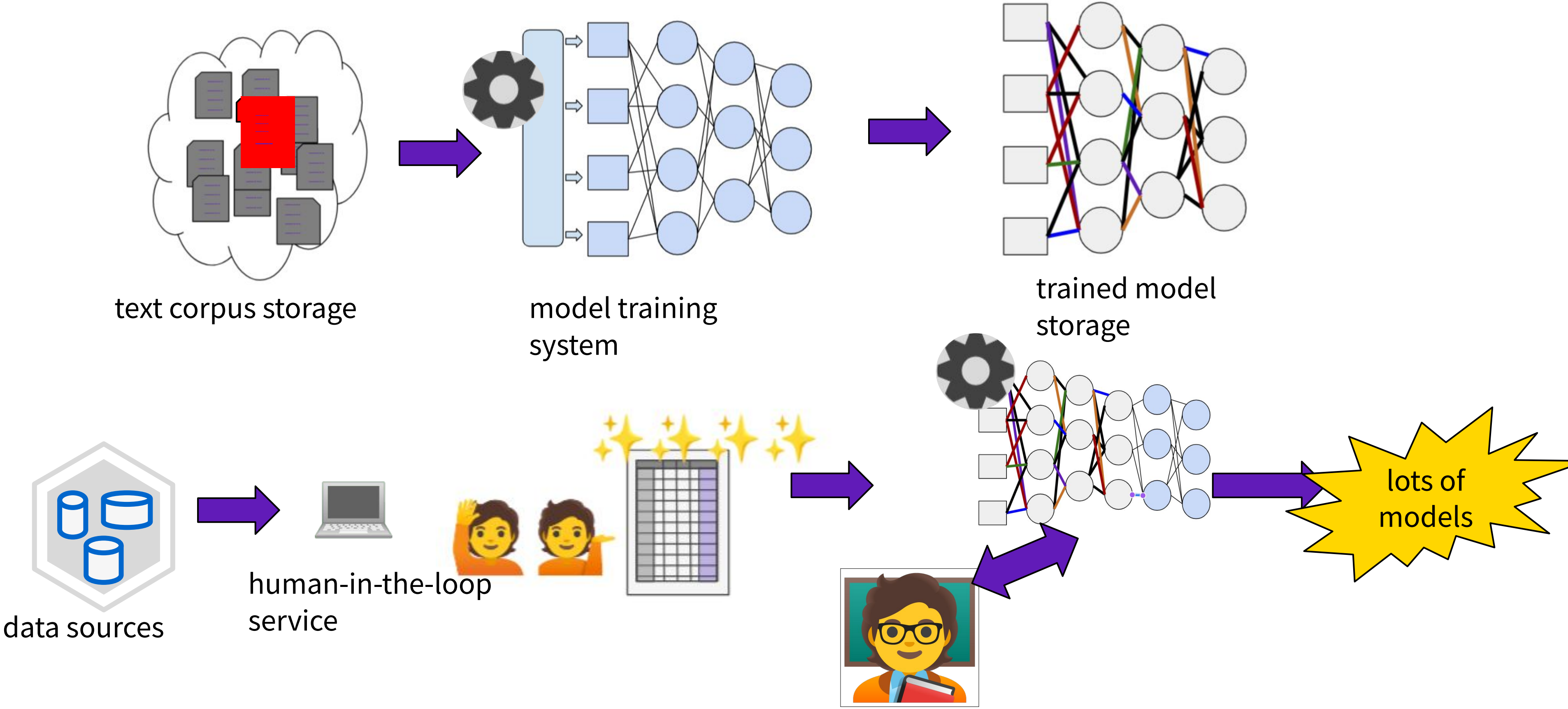
- Bad pre-trained model could lead to all the dependent (fine-tuned) models being wrong

Data risks: fine-tuning

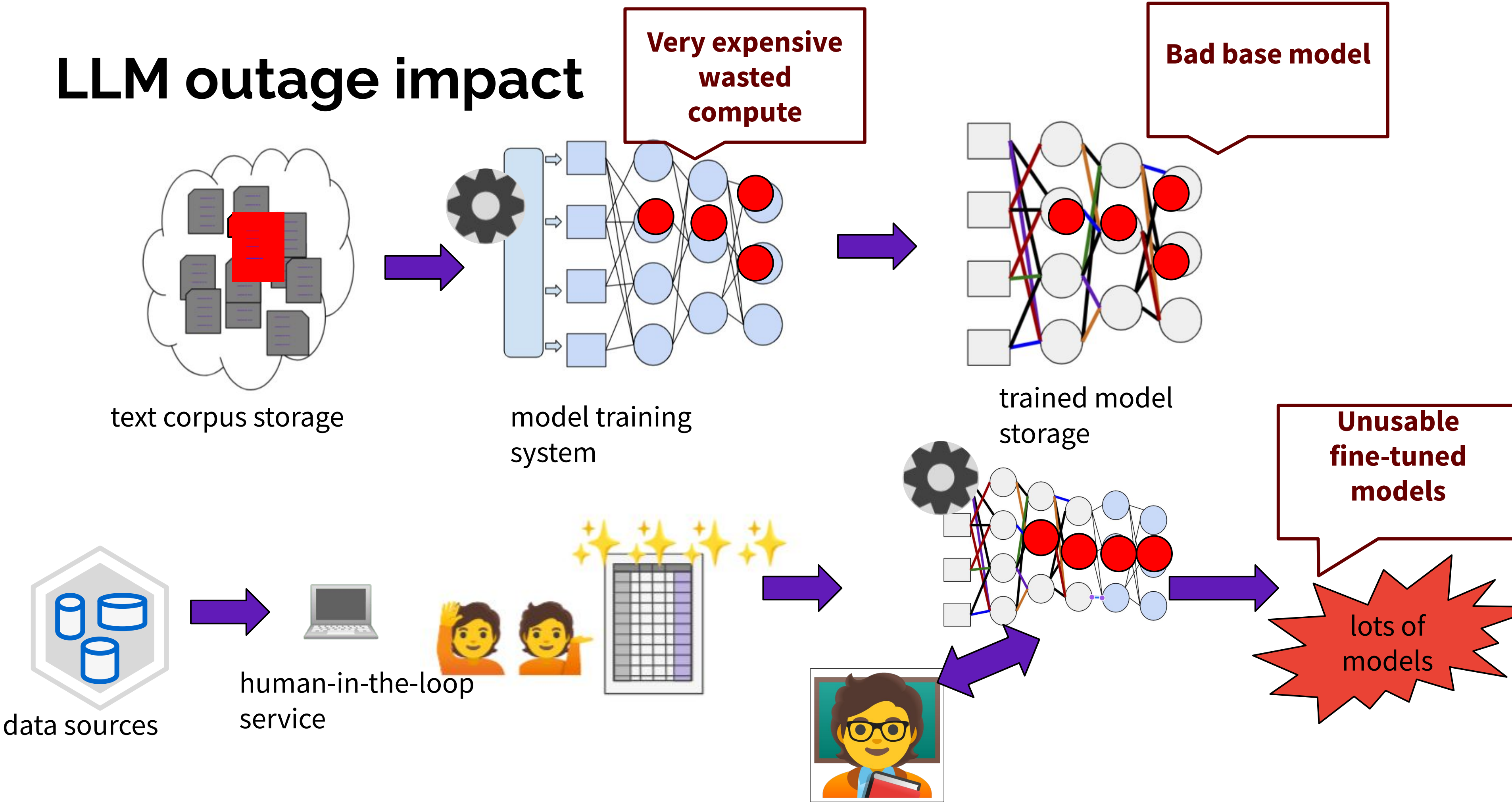
- Human labeled datasets are smaller -> criticality of high-quality data
- Tracking / consistency challenges for human labels
- Rating tasks need to work on a dynamic system, which is harder to keep stable
- High cost to validating model (ambiguous objective functions)
- Hard to monitor all these systems in sequence



LLM outage impact



LLM outage impact



Recommendations

Trace lineage of data *and* models throughout the journey

If you discover an issue with data or a model, you need to be able to trace back to debug a source, or trace forward to assess impact.

This is necessary for any ML model, but it is especially important for LLMs because:

- Higher complexity - more likely something will go wrong along the way
- Wider impact (especially for pre-training)

Trace lineage of data *and* models throughout the journey: How

- Datasets metadata store
- Model metadata store

Trace lineage of data *and* models throughout the journey: Why this is hard

- Lots of different kinds of data (including human operations)
- Many systems need to instrument tracking
- Need to get consensus on what metadata needs to be tracked

Ensure agreement between data producers and consumers for data integrity requirements

Many parties are consuming data (and models as data), and have expectations and requirements that the producers may not be aware of.

If data is misinterpreted between steps of the model journey, it can produce inaccurate results.

Ensuring agreement: How

For example:

- Where data is stored (and which versions are valid)
- What availability / recovery objectives are needed
- What data and model metadata should be tracked
- What validations exist (and how to tell if they were used)
- Data usage requirements
- What input data is writeable - and what processes can run against it
- Requirements for data labeling, and how to interpret labeled data
- How to handle shared data (incl models as data)

Ensuring agreement: why this is hard

- You have lots of roles involved. (Model researcher, system engineer, prompt writers, data labelers, model evaluators, human ops managers).
- They have different requirements, which may not be compatible with one single storage system
- There might be differences between the data that went in, vs how the model interpreted it

Make rollouts safe and rollback easy

Detecting model quality outages before they are widely visible is always best.

Even better if you can detect data quality issues before training.

But it if something does go wrong, you want to be able to detect it and recover quickly.

(Safe rollouts and easy rollbacks are not specific to ML.)

Make rollouts safe and rollback easy: How

- Gradual rollouts with canarying
- Reliable storage for snapshots and versions
- Test rollback procedures
- Retain old model versions, and know what is needed to recover
- Have ability to quickly exclude bad data before retraining

Make rollouts safe and rollback easy: Why this is hard

- Challenges in scale (time to load, ability to quickly swap versions for canarying)
- Backward compatibility for requests can be challenging
- Knowing when to roll back depends on how good your lineage tracking is

Conclusion

LLM data reliability is very similar to reliability for any other service, but **much more expensive** to get wrong. We have just highlighted:

- **Lineage tracking**: Instrumenting monitoring end-to-end, to surface difficult-to-detect failures
- **Agreements in data integrity:** The role of human communication paths in data integrity
- **Safe data and model rollouts/rollbacks:** The importance of limiting blast radius and knowing how to recover

Thanks!