Module 4

Fine-tuning and Evaluating LLMs

Learning Objectives

By the end of this module you will:

- Understand when and how to fine-tune models.
- Be familiar with common tools for training and fine-tuning, such as those from Hugging Face and DeepSpeed.
- Understand how LLMs are generally evaluated, using a variety of metrics.

A Typical LLM Release

A new generative LLM release is comprised of:

Multiple **sizes** (foundation/base model):

small | base | large

Multiple **sequence lengths:**

512 | 4096 | 62000

Flavors/fine-tuned versions (**base**, **chat**, **instruct**):

I know what word comes next.

I know how to engage in conversation.

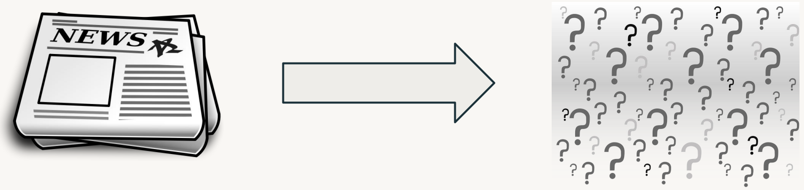
I know how to respond to instructions.

As a developer, which do you use?

For each use case, you need to balance:

- **Accuracy** (favors larger models)
- Speed (favors smaller models)
- *Task-specific performance*: (favors more narrowly fine-tuned models)

Let's look at example: **a news article summary app for riddlers**.



Applying Foundation LLMs:

Improving cost and performance with task-specific LLMs

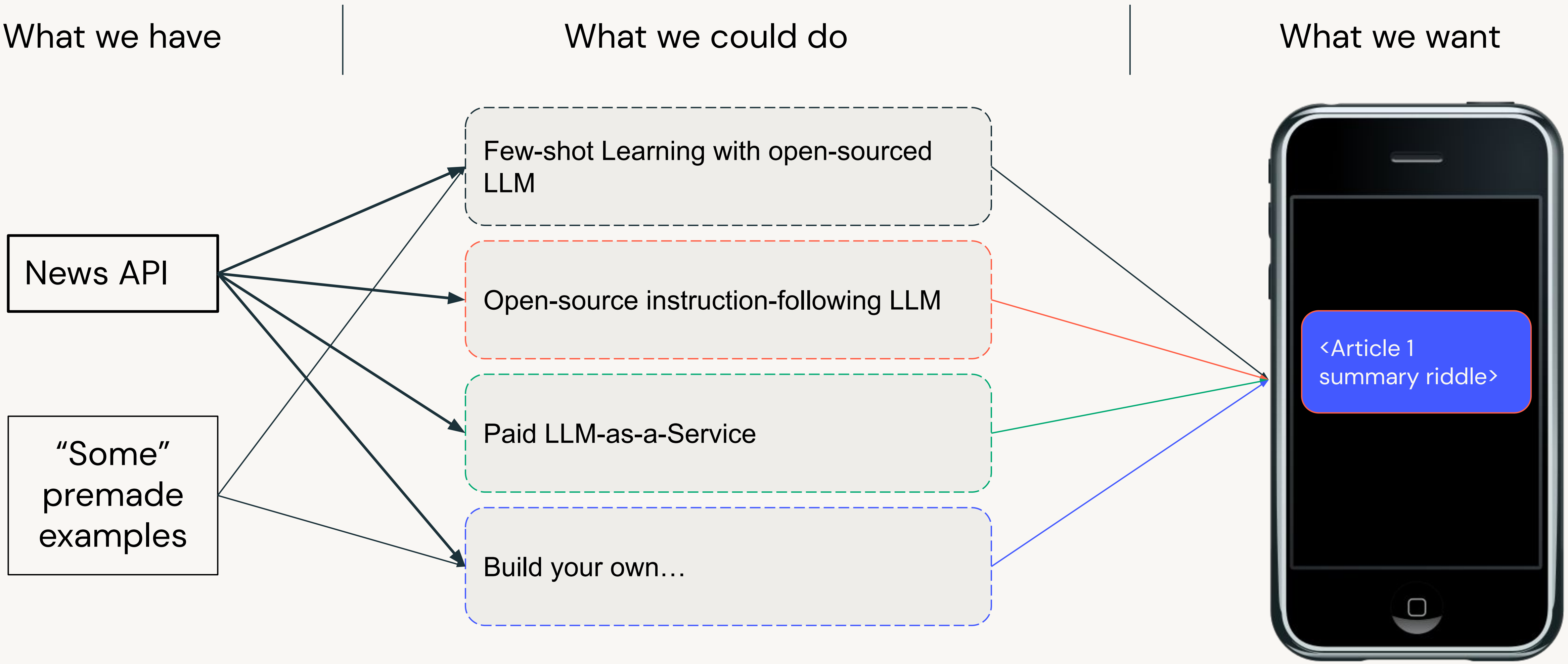
News Article Summaries App for Riddlers

My App - Riddle me this:
I want to create engaging and accurate article summaries for users in the form of *riddles*.

By the river's edge, a secret lies,
A treasure chest of a grand prize.
Buried by a pirate, a legend so old,
Whispered secrets and stories untold.
What is this enchanting mystery found?
In a riddle's realm, let your answer resound!

How do we build this?

Potential LLM Pipelines



Fine-Tuning: Few-shot learning

Potential LLM Pipelines

What we have	What we could do	What we want

News API

“Some” premade examples

Few-shot Learning with open-source LLM

Pros and cons of Few-shot Learning

Pros

- Speed of development
  - Quick to get started and working.
- Performance
  - For a larger model, the few examples often lead to good performance
- Cost
  - Since we're using a released, open LLM, we only pay for the computation

Cons

- Data
  - Requires a number of good-quality examples that cover the intent of the task.
- Size-effect
  - Depending on how the base model was trained, we may need to use the largest version which can be unwieldy on moderate hardware.

Riddle me this: Few-shot Learning version

Let's build the app with few shot learning and the new LLM

Our new articles are long, and in addition to summarization, the LLM needs to reframe the output as a riddle.

- Large version of base LLM
- Long input sequence

```
prompt = (
"""For each article, summarize and create a riddle
from the summary:
[Article 1]: "Residents were awoken to the surprise…"
[Summary Riddle 1]: "In houses they stay, the peop… "
###
[Article 2]: "Gas prices reached an all time …"
[Summary Riddle 1]: "Far you will drive, to find…"
###
…
###
[Article n]: {article}
[Summary Riddle n]:""")
```

Fine-Tuning:
Instruction-following LLMs

Potential LLM Pipelines

What we have | What we could do | What we want

News API

"Some" premade examples

Instruction-following LLM

Pros and cons of Instruction-following LLMs

Pros

- Data
  - Requires no few-shot examples. Just the instructions (aka zero-shot learning).
- Performance
  - Depending on the dataset used to train the base and fine-tune this model, may already be well suited to the task.
- Cost
  - Since we're using a released, open LLM, we only pay for the computation.

Cons

- Quality of fine-tuning
  - If this model was not fine-tuned on similar data to the task, it will potentially perform poorly.
- Size-effect
  - Depending on how the base model was trained, we may need to use the largest version which can be unwieldy on moderate hardware.

Riddle me this: Instruction-following version

Let's build the app with the Instruct version of the LLM

The new LLM was released with a number of fine-tuned flavors.

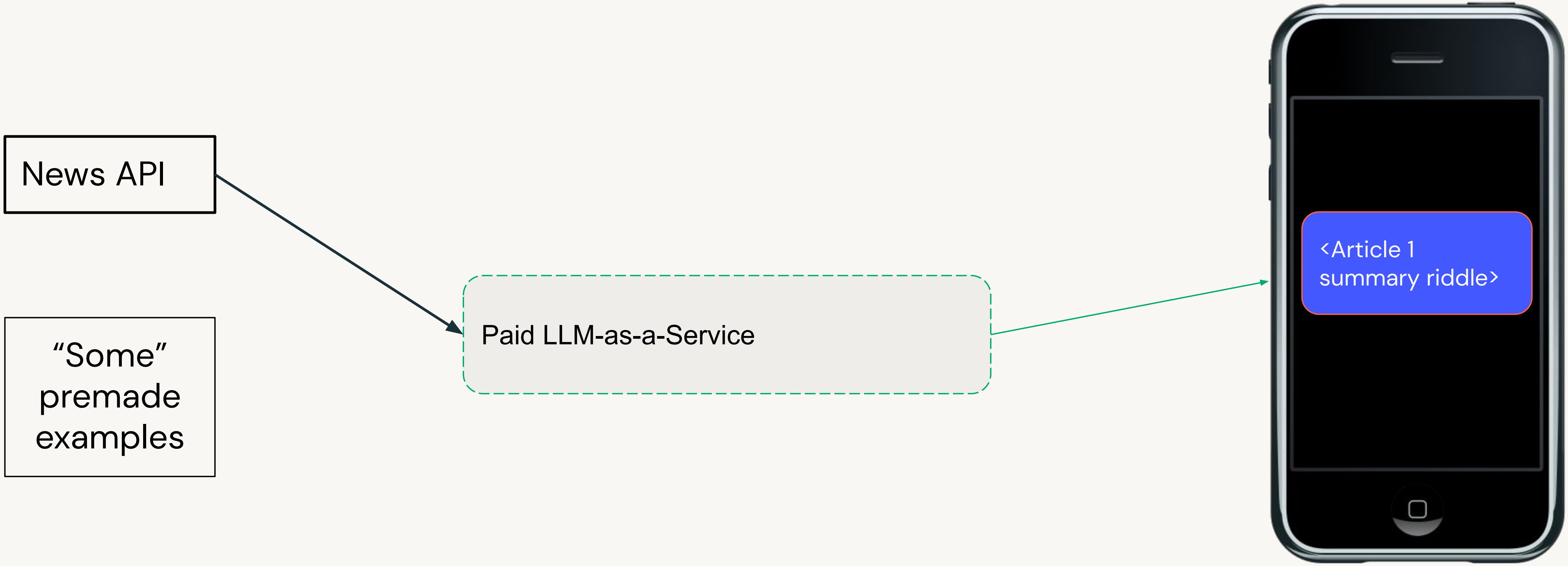
Let's use the Instruction-following LLM one as is and leverage zero-shot learning.

```
prompt = (
"""For the article below, summarize and create a
riddle from the summary:
[Article n]: {article}
[Summary Riddle n]:""")
```

Fine-Tuning: LLMs-as-a-Service

Potential LLM Pipelines

What we have | What we could do | What we want



Pros and cons of LLM-as-a-Service

Pros

- Speed of development
  - Quick to get started and working.
  - As this is another API call, it will fit very easily into existing pipelines.
- Performance
  - Since the processing is done server side, you can use larger models for best performance.

Cons

- Cost
  - Pay for each token sent/received.
- Data Privacy/Security
  - You may not know how your data is being used.
- Vendor lock-in
  - Susceptible to vendor outages, deprecated features, etc.

Riddle me this: LLM-as-a-Service version

Let's build the app using an LLM-as-a-service/API

This requires the least amount of effort on our part.

Similar to the Instruction-following LLM version, we send the article and the instruction on what we want back.

```
prompt = (
"""For the article below, summarize and create a
riddle from the summary:
[Article n]: {article}
[Summary Riddle n]:""")

response =
LLM_API(prompt(article),api_key="sk-@sjr...")
```

Fine-tuning: DIY

Potential LLM Pipelines

What we have | What we could do | What we want

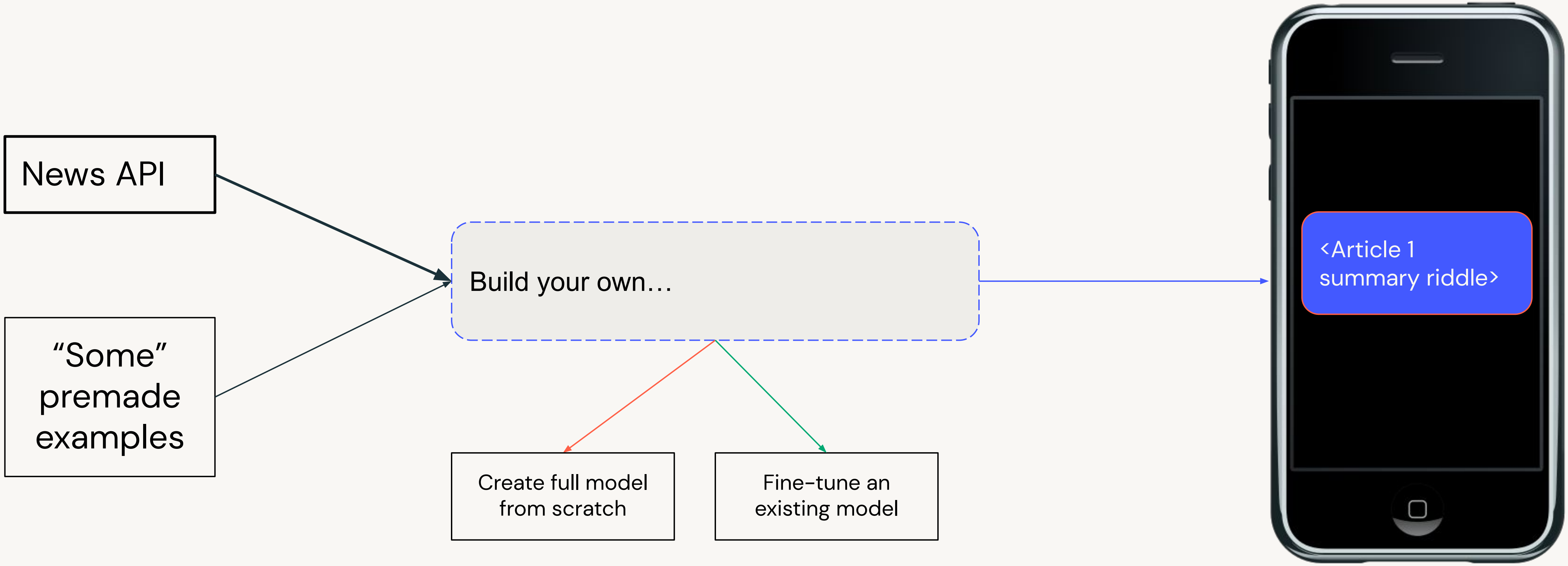
News API

"Some" premade examples

Build your own…

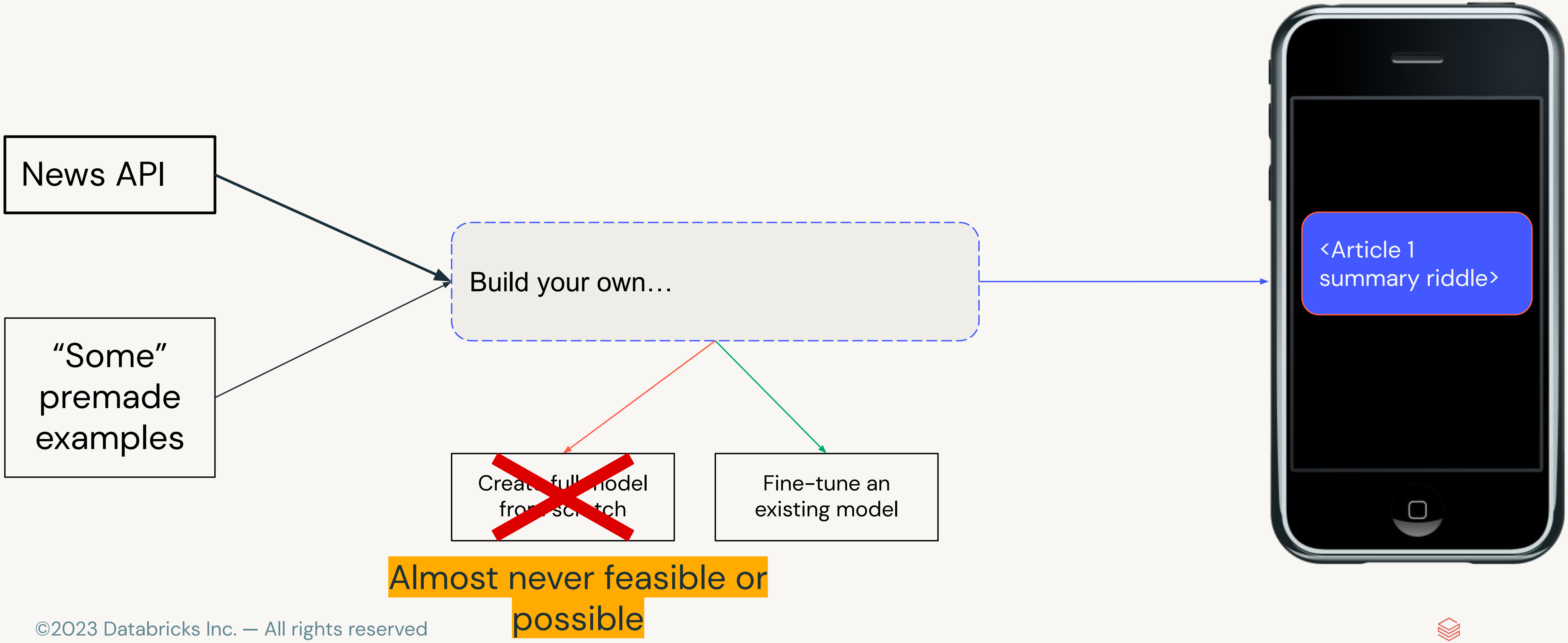
Potential LLM Pipelines

What we have | What we could do | What we want



Potential LLM Pipelines

What we have | What we could do | What we want



Pros and cons of fine-tuning an existing LLM

Pros

- Task-tailoring
  - Create a task-specific model for your use case.
- Inference Cost
  - More tailored models often smaller, making them faster at inference time.
- Control
  - All of the data and model information stays entirely within your locus of control.

Cons

- Time and Compute Cost
  - This is the most costly use of an LLM as it will require both training time and computation cost.
- Data Requirements
  - Larger models require larger datasets.
- Skill Sets
  - Require in-house expertise.

Riddle me this: fine-tuning version

Let's build the app using a fine-tuned version of the LLM

Depending on the amount and quality of data we already have, we can do one of the following:

- Self-instruct (Alpaca and Dolly v1)
  - Use another LLM to generate synthetic data samples for data augmentation.
- High-quality fine-tune (Dolly v2)
  - Go straight to fine tuning, if data size and quality is satisfactory.

Free Dolly:

Introducing the World's First Truly Open Instruction-Tuned LLM

What is Dolly?

An instruction-following LLM with a tiny parameter count less than 10% the size of ChatGPT.

EleutherAI

Pythia 12B:

```
Layers:36 Dimensions:5120
Heads:40 Seq. Len:2048
```

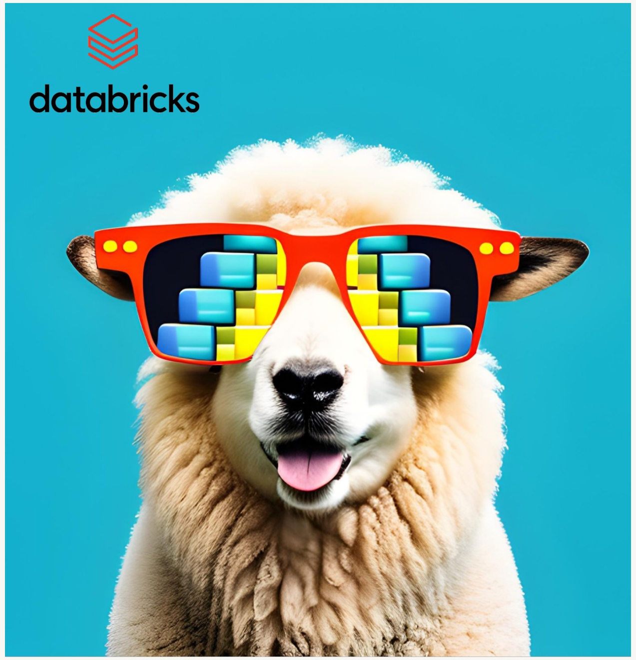
The Pile
800GB Dataset of Diverse Text for Language Modeling

+

databricks

databricks-dolly-15k

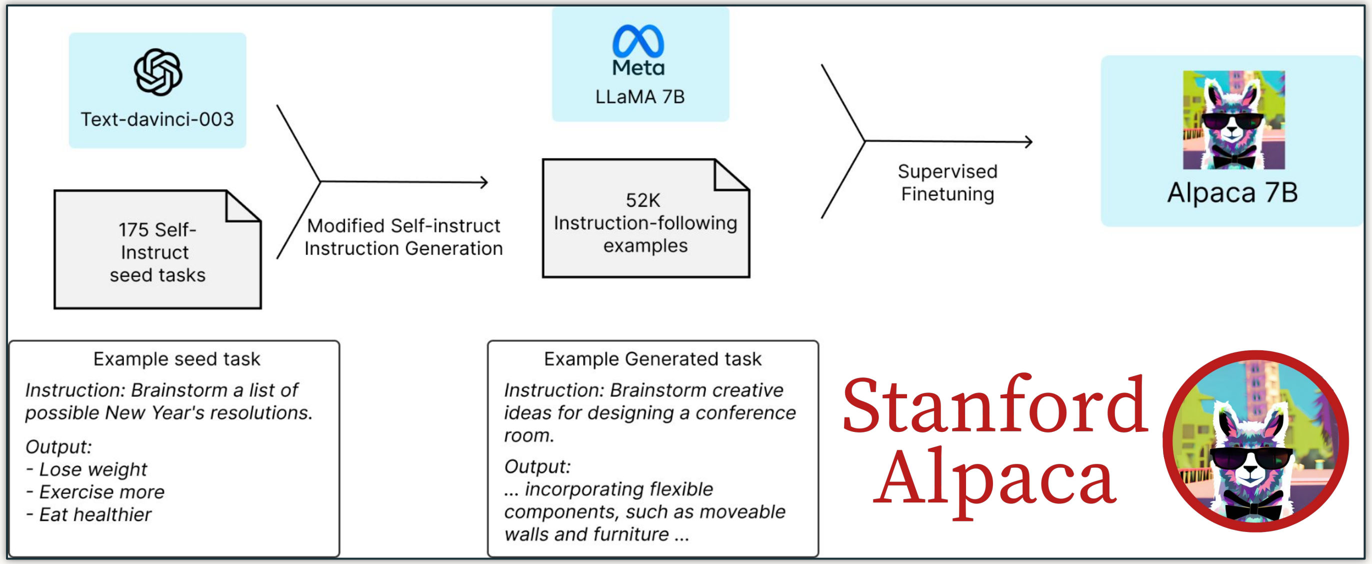
=



Entirely open source and available for commercial use.

Where did Dolly come from?

The idea behind Dolly was inspired by the Stanford Alpaca Project.



This follows on a trend in LLM research:

Smaller models >> Larger models

Training for longer on more high quality data.

However these models all lacked the open commercial licensing affordances.

The Future of Dolly

2018-2023

The foundation model era: racing to 1 trillion parameter transformer models

"I think we're at the end of the era ..[of these]... giant, giant models"

- Sam Altman, CEO OpenAI, April 2023

2023 and beyond

The Age of small LLMs and Applications

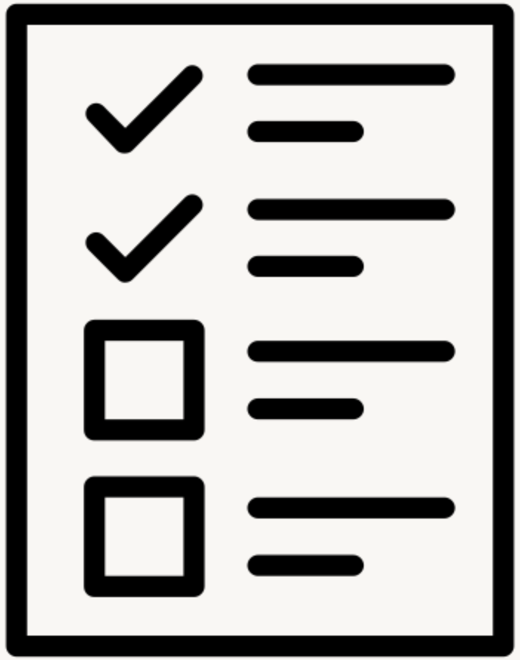
Evaluating LLMs:

“There sure are a lot of metrics out there!”

So you've decided to fine-tune…

Did it work? How can you measure LLM performance?

EVALUATION TIME!



Training Loss/Validation Scores

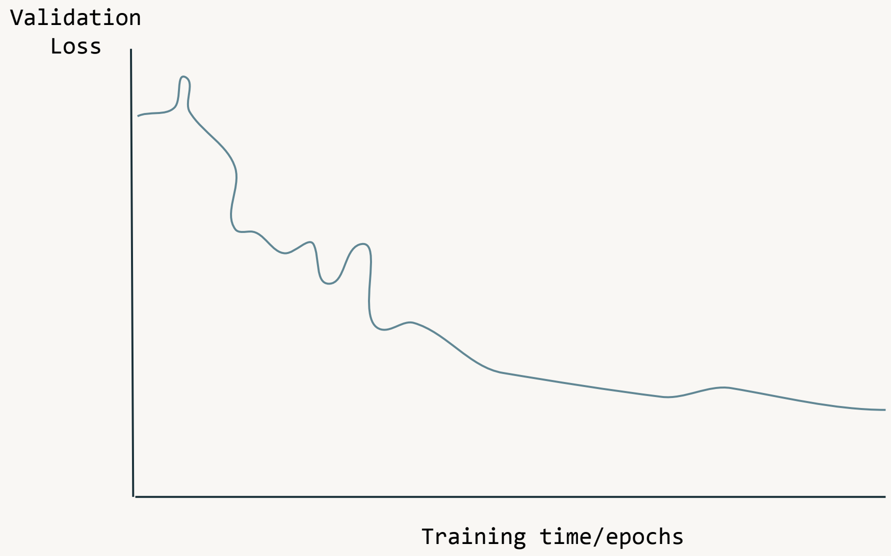
What we watch when we train

Like all deep learning models, we monitor the loss as we train LLMs.

But for a good LLM what does the loss tell us?

Nothing really. Nor do the other typical metrics

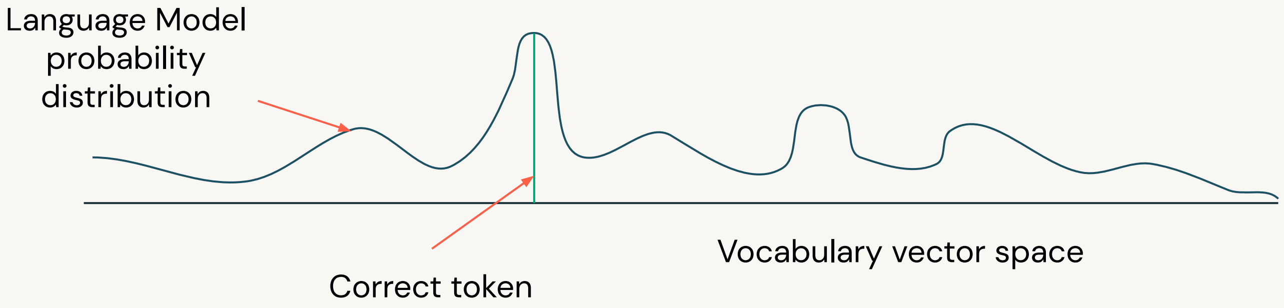
Accuracy, F1, precision, recall, etc.



Perplexity

Is the model surprised it got the answer right?

A good language will model will have **high accuracy** and **low perplexity**



Accuracy = next word is right or wrong.

Perplexity = how confident was that choice.

More than perplexity

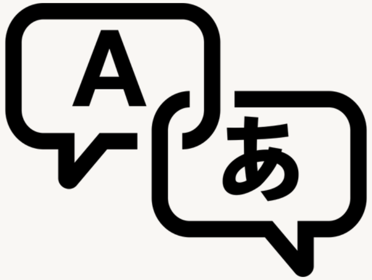
Task-specific metrics

Perplexity is better than just accuracy.

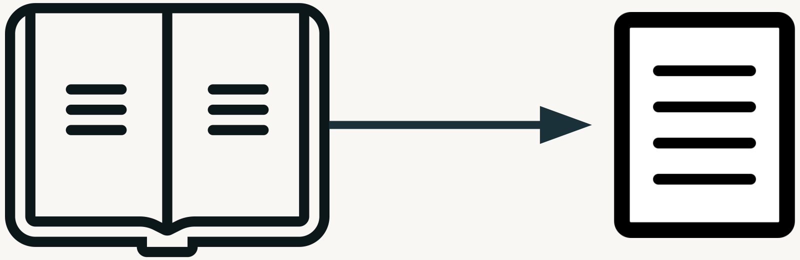
But it still lacks a measure context and meaning.

Each NLP task will have different metrics to focus on. We will discuss two:

Translation – BLEU



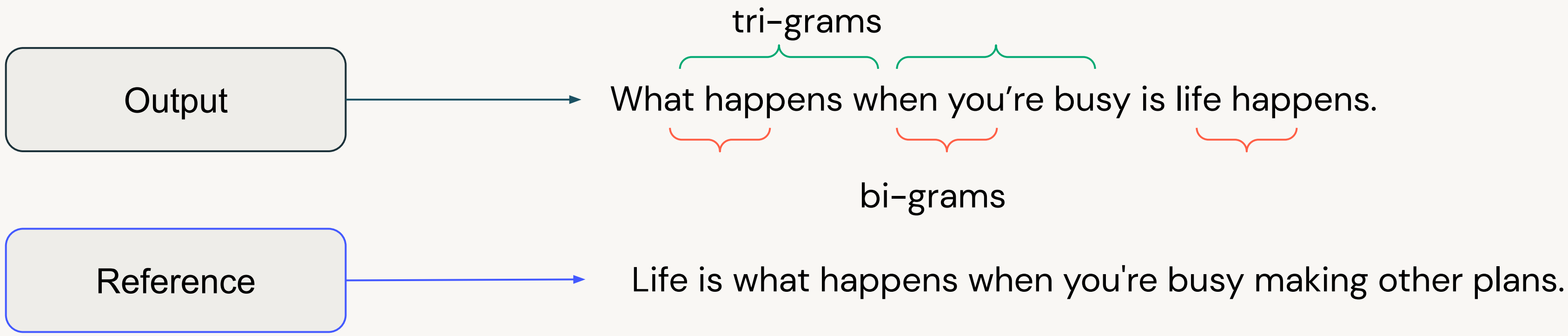
Summarization – ROUGE



Task-specific Evaluations

BLEU for translation

BiLingual Evaluation Understudy



BLEU uses reference sample of translated phrases to calculate n-gram matches: uni-gram, bi-gram, tri-gram, and quad-gram.

ROUGE for summarization

$$\text{ROUGE-N} = \frac{\sum_{S \in \{\text{Reference summaries}\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{\text{Reference summaries}\}} \sum_{gram_n \in S} Count(gram_n)}$$

Total matching N-grams

Total N-grams

N-gram recall

ROUGE score for N-grams, e.g., ROUGE-1 for words

Sum over reference summaries (test data)

Sum over N-grams in summary *S*

ROUGE-1	Words (tokens)
ROUGE-2	Bigrams
ROUGE-L	Longest common subsequence
ROUGE-Lsum	Summary-level ROUGE-L

Reference: https://aclanthology.org/W04-1013.pdf

Benchmarks on datasets: SQuAD

Stanford Question Answering Dataset – reading comprehension

- Questions about Wikipedia articles
- Answers may be text segments from the articles, or missing

Given a Wikipedia article
Steam engines are external combustion engines, where the working fluid is separate from the combustion products. Non-combustion heat sources such as ***solar power***, nuclear power or geothermal energy may be used. The ideal thermodynamic cycle used to analyze this process is called the Rankine cycle. In the cycle, …

Given a question
Along with geothermal and nuclear, what is a notable non-combustion heat source?

Select text from the article to answer (or declare no answer)
"solar power"

References: [Rajpurkar et al., 2016](#) and https://rajpurkar.github.io/SQuAD-explorer/

Evaluation metrics at the cutting edge

ChatGPT and InstructGPT (predecessor) used similar techniques

1. Target application
   a. NLP tasks: Q&A, reading comprehension, and summarization
   b. Queries chosen to match the API distribution
   c. Metric: human preference ratings
2. Alignment
   a. "Helpful" → Follow instructions, and infer user intent. Main metric: human preference ratings
   b. "Honest" → Metrics: human grading on "hallucinations" and TruthfulQA benchmark dataset
   c. "Harmless" → Metrics: human and automated grading for toxicity (RealToxicityPrompts); automated grading for bias (Winogender, CrowS-Pairs)
      i. Note: Human labelers were given very specific definitions of "harmful" (violent content, etc.)

Reference: https://arxiv.org/abs/2203.02155

Module Summary

Fine-tuning and Evaluating LLMs - What have we learned?

- Fine-tuning models can be useful or even necessary to ensure a good fit for the task.
- Fine-tuning is essentially the same as training, just starting from a checkpoint.
- Tools have been developed to improve the training/fine-tuning process.
- Evaluating a model is crucial for model efficacy testing.
- Generic evaluation tasks are good for all models.
- Specific evaluation tasks related to the LLM focus are best for rigor.

Time for some code!