PROMPT ENGINEERING IS DEAD

A practitioner's approach to building LLM Apps
Presented June 12 , 2024

HI. I'M MATT

Today we'll dig into exciting research and tools to build better LLM apps

- I'm a practitioner with over 15 years of business, technology and data science experience. My primary focus today will be to present methods to help other practitioners.
- I'm not a researcher or affiliated with the amazing folks who do the real work behind the insights and tools we're discussing today. I will footnote many sources in this presentation - as not to take credit from whom it's due.
- The views expressed and examples are my own. I will not cover any exact use cases from my current or former employers.

DATA+AI SUMMIT

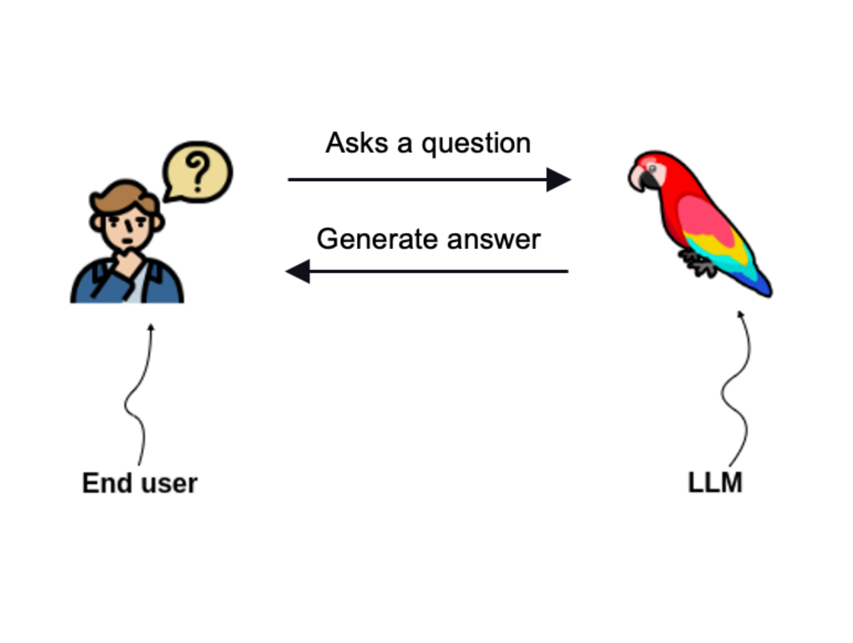
AGENDA

1. Why build agents
2. Prompting strategies & evaluating prompt quality
3. Why I love DSPy framework & using it with Databricks
4. Demonstration

BUILDING AGENTS LEVERAGING LANGUAGE MODELS

THE BLACK BOX APPROACH

User's may see the black box magic and assume we just need a magic prompt



Icon Credit: https://neo4j.com/developer-blog/advanced-rag-strategies-neo4j/

- Don't be fooled, a single LLM call (or RAG) and a magic prompt may get you 80% of the way to a great app, but **the last 20% require a different approach**
- There will be a future LLM abstraction, years from now, that only requires a single call to a black box. **Today's practical application of LLMs require more.**

THE BLACK BOX APPROACH

RAG is a step in the right direction, but still requires “prompt engineering”

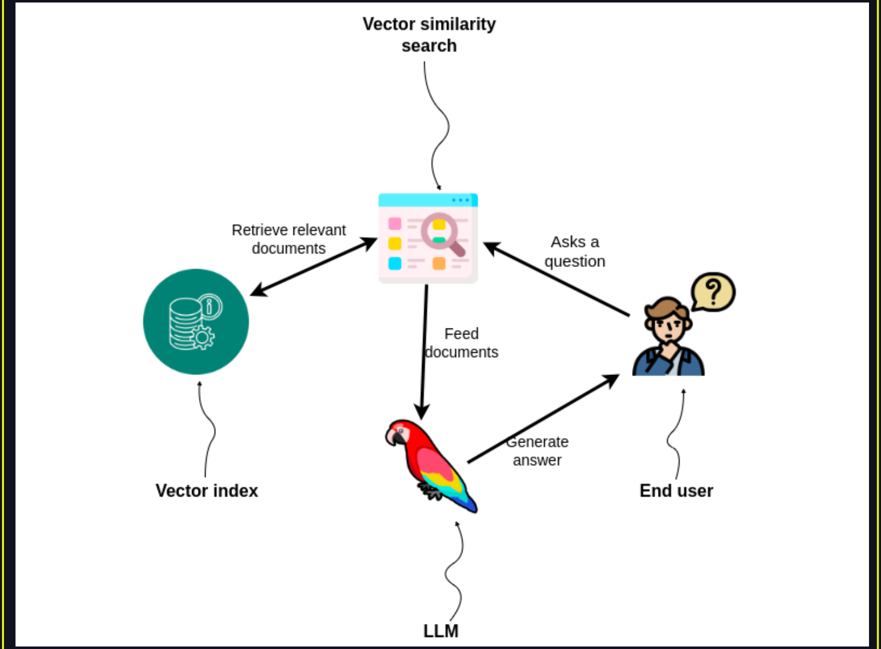


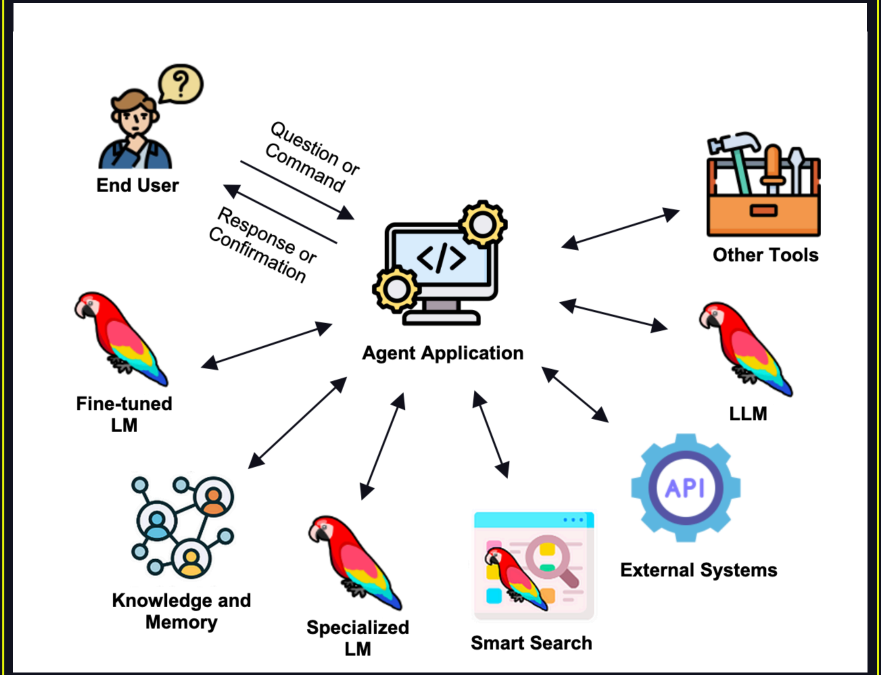
Image Credit: https://neo4j.com/developer-blog/advanced-rag-strategies-neo4j/

- Don’t be fooled, a single LLM call (or RAG) and a magic prompt may get you 80% of the way to a great app, but **the last 20% require a different approach**
- There will be a future LLM abstraction, years from now, that only requires a single call to a black box. **Today’s practical application of LLMs require more.**

THE AGENT APPROACH

AI agents who take a sequence of actions for us is the real promise of AI

- The value will be found in agents who interact with other systems and the world around us
- A LM app with language inputs and outputs can still leverage an agent approach
- An agent is intellectual property (IP) for your enterprise
- We can optimize performance and latency of agents



Icon credit: neo4j.com/developer-blog/knowledge-graphs-llms-multi-hop-question-answering; Flaticon.com

Sources and Resources:

- *changelog.com/practicalai/269*
- *writer.com/blog/larger-llms-vs-purpose-built-for-enterprise*

RESEARCH TO GUIDE THE WAY

These papers shaped my thinking on an agent approach

DSPy: [...] Self-improving Pipelines

- Framework for programmatically building pipelines and optimizing the outputs
- This will form the bulk of our examples today

Sources:

- Khattab et al., 2023
- *arxiv.org/pdf/2310.03714*
- *youtube.com/watch?v=NoaDWKHdkHg*
- *github.com/stanfordnlp/dspy*

Large Language Models As Optimizers

- Similar prompts can have very different outcomes
- The best prompt is specific to the task and model
- LLMs outperform humans at prompt optimization
- LLMs perform better in simpler problem spaces

Sources:

- Yang et al., 2023
- https://arxiv.org/pdf/2309.03409

ERAGent: Enhancing Retrieval-Augmented LMs...

- RAG systems benefit from question rewriting and retrieval filtering and re-ranking
- Personalization is possible with further agent tuning on historical interactions

Sources:

- Shi et al. 2024
- arxiv.org/pdf/2405.06683

PROMPTING STRATEGIES & EVALUATING PROMPT QUALITY

PROMPT ENGINEERING STRATEGIES

Many strategies have emerged to prompt the "right answer" out of a LM

- Zero Shot – directly instructing LM without any example
- Few Shot – prompting examples of how the LM should behave
- Ask Nicely - being encouraging helps LM perform better?!
- Chain of Thought – ask model to describe logic in output
- Chain of Density / Rewrite – Iterative prompts to refine output
- ReAct – allow LM to reason through action to take next
- Stepback Prompting – Generalizing fundamental question with LM before answering
- Prompt Injection – Jailbreaking, hacking, and other bad outcomes
- Chaining prompts – the basis for the agent approach we're discussing today
- And MANY more…

Many more examples at: promptingguide.ai/techniques

EVALUATING PROMPT QUALITY

The first step in prompt engineering is NOT writing a prompt!

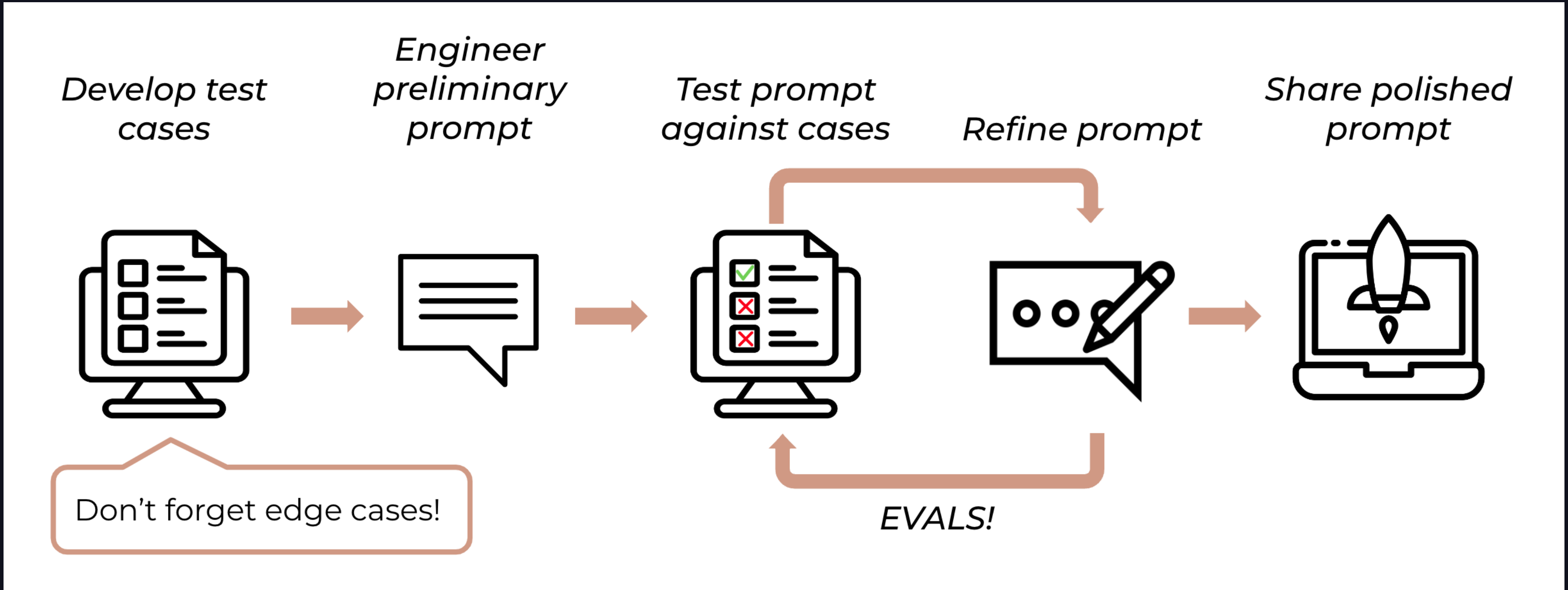


Image source: docs.anthropic.com/en/docs/prompt-engineering

EVALUATING PROMPT QUALITY

Let's learn the lessons of DevOps and QE. Automated testing is everything!

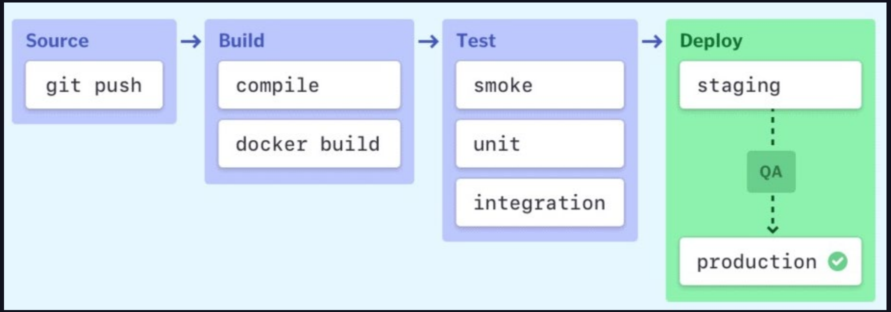


Image source: semaphoreci.com/blog/cicd-pipeline

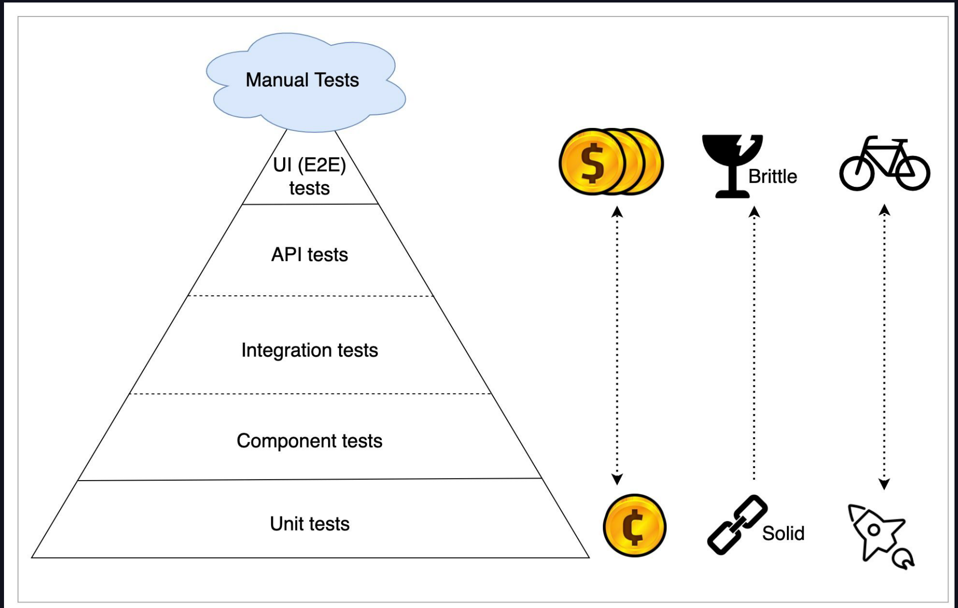


Image source: https://getmason.io/blog/post/test-pyramid/

Don't bother figuring out what special magic combination of words will give you the best performance for your task. Just develop a scoring metric then let the model optimize itself.

-- Rick Battle, VMware (paraphrase)
"Don't Start a Career as an AI Prompt Engineer." IEEE Spectrum May 2024 Issue

“A lot of people anthropomorphize [LLMs] because they ‘speak English.’ No, they don’t. It doesn’t speak English. It does a lot of math.”

-- Rick Battle, VMware (paraphrase)
“Don’t Start a Career as an AI Prompt Engineer.” IEEE Spectrum May 2024 Issue

LM EVALUATION STRATEGIES

Building good metrics == effective LM app.
It’s hard; that’s why you have a job

Standard Metrics

- Exact Match (numeric and categorization tasks)
- BLEU, ROUGE, METEOR, BERTScore
- Custom, hand-written

Libraries and SaaS

- RAGAs et al.
- SaaS tools

Cross-Model Evaluation

- LLMs are actually pretty good a evaluating themselves – as an in context task!
- RLHF-ish
- MLFlow.evaluate
- DSPy custom program

WHY I LOVE DSPy FRAMEWORK & USING IT WITH DATABRICKS

WHY DSPy?

DSPy makes it easy to follow the data science process when building LM apps

Why DSPy?

- Created by Omar Khattab et al. at Stanford
- I listened to an interview with Omar in 2023 and thought it brilliant. *I'm not affiliated with the project in any way.*
- Framework for implementing all the concepts we discussed so far

General Workflow

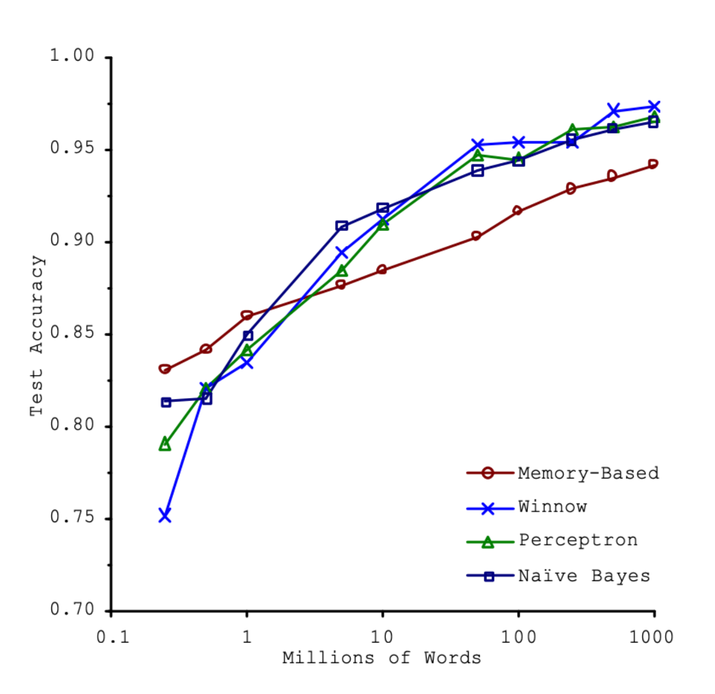
1. Define you task
2. Collect some data and LM/RM connection
3. Define your metric
4. Setup a pipeline
5. Compile/Optimize your program
6. Save your experiment and iterate

Source: dspy-docs.vercel.app/docs/building-blocks/solving_your_task

DATA MATTERS MOST

One of the most famous charts in Data Science, still holds true after 23 years

- In 2001, Microsoft Research published a paper noting accuracy came from more data rather than the algorithm
- I use DSPy because **it let's me focus on the data** – not the prompt or the code.



Scaling to Very Very Large Corpora for Natural Language Disambiguation. 2021. Banko and Brill

INTEGRATING WITH DATABRICKS

It’s simple to use DSPy in Databricks

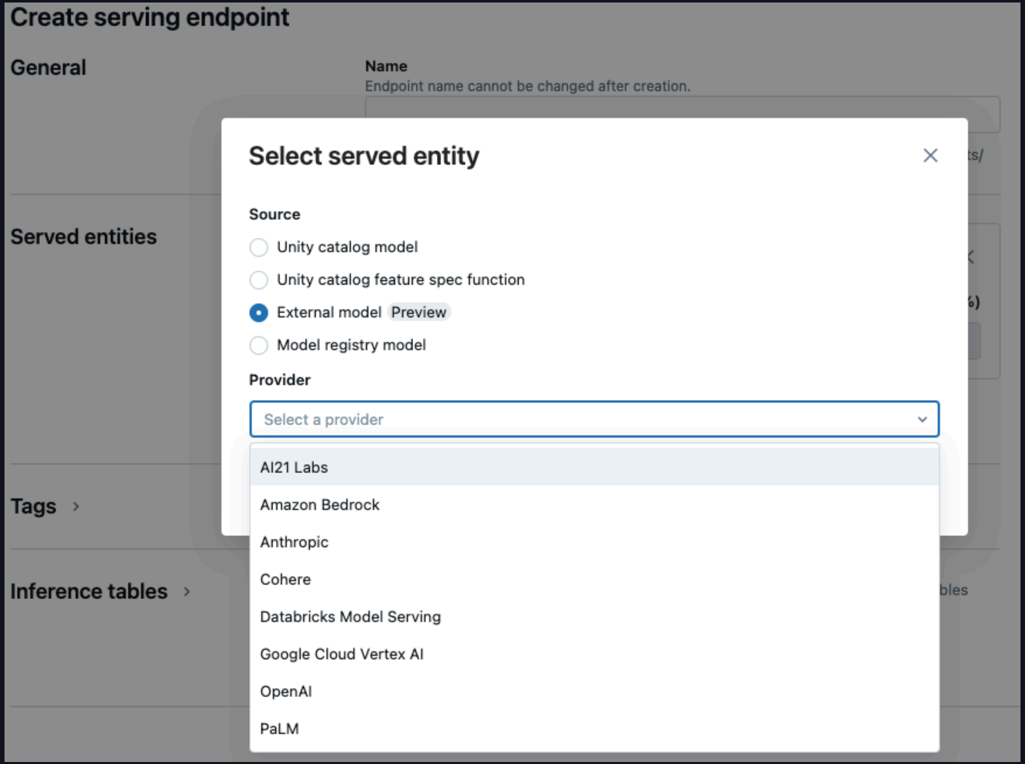
```
# 1. Install the libs
!pip install dspy-ai, databricks-vectorsearch
```

PYTHON

```python
# 2. Create configuration to Databricks’ served LM and/or Vector DB
import os, dspy

api_key = os.environ.get('DATABRICKS_TOKEN’)
workspace = ’[your workspace here]’
api_base = f’https://{workspace}.azuredatabricks.net/serving-endpoints’
model = 'databricks-mixtral-8x7b-instruct’, # name of served model

# Setup the clients
lm = dspy.Databricks(model, api_key, api_base, model_type='chat’)
retriever_model = DatabricksRM(databricks_index_name, databricks_endpoint, databricks_token, columns, k)

# Set config in dspy
dspy.settings.configure(lm=lm, rm=retriever_model)
```

SIDENOTE: EXTERNAL MODEL SERVING

Using Databricks' External Model Serving unifies interface and authorization



THREE IMPORTANT CONCEPTS IN DSPy

These are the building blocks to create agents

Signatures

- Defines the inputs and outputs of one component in your pipeline
- This takes the place of writing a prompt
- Examples: "input -> output"
    - "question -> answer"
    - "sentence -> sentiment"
    - "document -> summary"

Source: dspy-docs.vercel.app/docs/building-blocks/signatures

Modules

- Implements a prompt engineering strategy
- Is the learnable param(s) wrapped around a Signature (inspired by PyTorch modules)
- This is the layer that interacts with an LM or Retrieval
- Combine into a full program, and can run as zero-shot

Source: dspy-docs.vercel.app/docs/building-blocks/modules

Optimizers

- This is the brilliant part of this framework and results in better than human prompt writing results!
- Defines the prompt optimization method and metric
- You'll want train/test/holdout data at this point

Source: dspy-docs.vercel.app/docs/building-blocks/optimizers

Deeper Dive on Optimizers

There are MANY options to experiment with. Start simply and expand.

- Each Module in your program has multiple params to tune: prompt instructions and few shot demonstrations (and even LM weights, if desired)
- Thoughtful construction of the metric to optimize is key
- Key Optimizers, in order of complexity:
  - BootstrapFewShotWithRandomSearch – searches for best set of few shot prompt
  - MIPRO – optimizes prompt instructions and few shot demonstrations
  - BootstrapFinetune – fine tunes LM ‘s weights for optimization

DEMONSTRATION

DATA+AI SUMMIT

THANK YOU FOR LISTENING