Building Generative AI Applications

AN LLM CASE STUDY

Michelle Yi

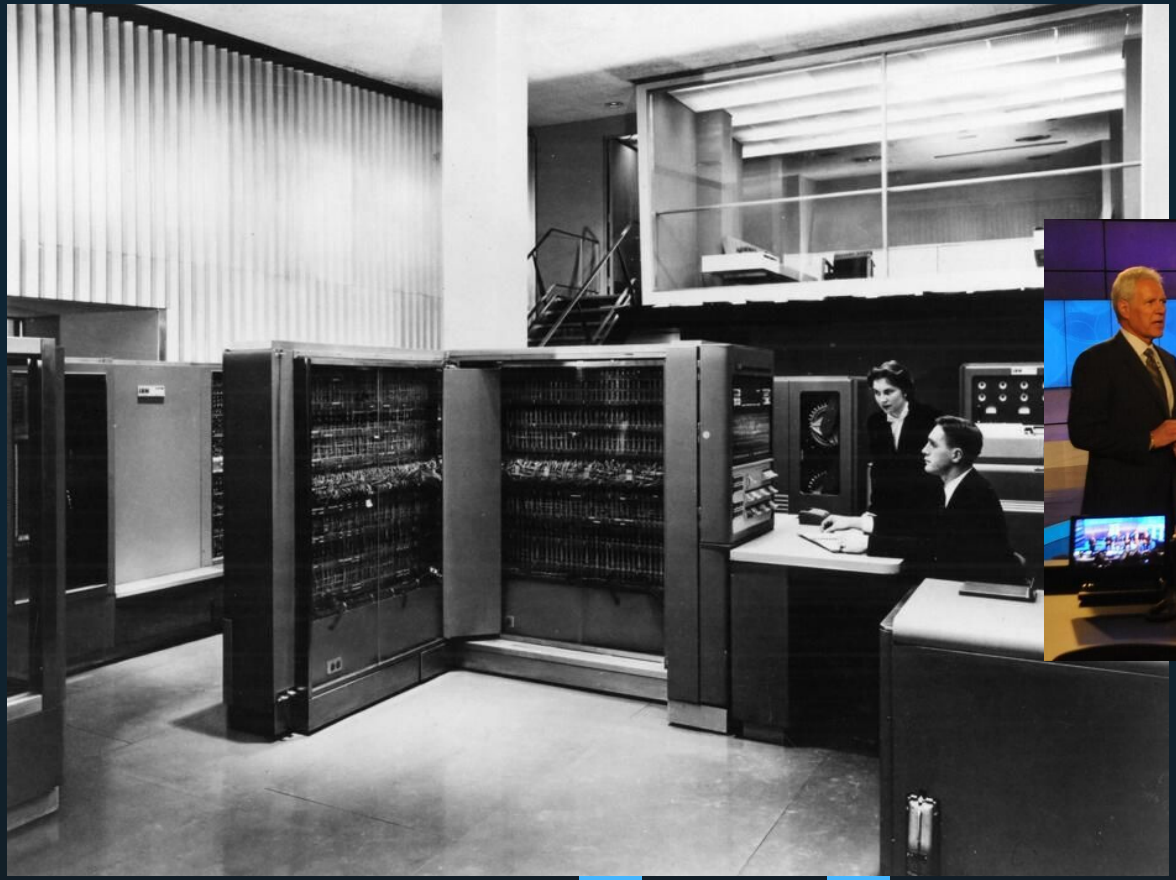
ODSC 2023

INTRODUCTION

Michelle Yi | Michelle@generationship.ai

Applied AI, Artist, Speaker

Board Member, Women in Data



https://www.techrepublic.com/article/ibm-watson-the-inside-story-of-how-the-jeopardy-winning-supercomputer-was-born-and-what-it-wants-to-do-next/

DESIRED OUTCOME

Take any LLM-related idea that you have and feel empowered to bootstrap it with open source tooling.

DESIRED OUTCOME

1. A framework for building LLM-driven applications
2. An architecture of how to get started


HOW IT ALL
STARTED

HOW IT ALL STARTED

https://dnd.wizards.com/

DUNGEONS AND DRAGONS

https://dnd.wizards.com/; https://www.usatoday.com/story/tech/gaming/2020/01/14/dungeons-dragons-role-playing-game-popular-again-why/4427635002/

DUNGEONS AND DRAGONS

https://dnd.wizards.com/; https://www.usatoday.com/story/tech/gaming/2020/01/14/dungeons-dragons-role-playing-game-popular-again-why/4427635002/

D&D BASIC RULES

Player's Basic Rules Version 0.2

Credits

D&D Lead Designers: Mike Mearls, Jeremy Crawford

Design Team: Christopher Perkins, James Wyatt, Rodney Thompson, Robert J. Schwalb, Peter Lee, Steve Townshend, Bruce R. Cordell
Editing Team: Chris Sims, Michele Carter, Scott Fitzgerald Gray
Producer: Greg Bilsland

Art Directors: Kate Irwin, Dan Gelon, Jon Schindehette, Mari Kolkowsky, Melissa Rapier, Shauna Narciso
Graphic Designers: Bree Heiss, Emi Tanji
Interior Illustrator: Jaime Jones

Additional Contributors: Kim Mohan, Matt Sernett, Chris Dupuis, Tom LaPille, Richard Baker, Chris Tulach, Miranda Horner, Jennifer Clarke Wilkes, Steve Winter, Nina Hess

Project Management: Neil Shinkle, Kim Graham, John Hay
Production Services: Cynda Callaway, Brian Dumas, Jefferson Dunlap, Anita Williams

Based on the original D&D game created by
E. Gary Gygax and Dave Arneson, with Brian Blume, Rob Kuntz, James Ward, and Don Kaye
Drawing from further development by
J. Eric Holmes, Tom Moldvay, Frank Mentzer, Aaron Allston, Harold Johnson, David "Zeb" Cook, Ed Greenwood, Keith Baker, Tracy Hickman, Margaret Weis, Douglas Niles, Jeff Grubb, Jonathan Tweet, Monte Cook, Skip Williams, Richard Baker, Peter Adkison, Bill Slavicsek, Andy Collins, and Rob Heinsoo

Playtesting provided by
over 175,000 fans of D&D. Thank you!

Additional consultation provided by
Jeff Grubb, Kenneth Hite, Kevin Kulp, Robin Laws, S. John Ross, the RPGPundit, Vincent Venturella, and Zak S.

Ability Score Summary

Strength
Measures: Natural athleticism, bodily power
Important for: Fighter
Racial Increases:
Mountain dwarf (+2)
Human (+1)

Dexterity
Measures: Physical agility, reflexes, balance, poise
Important for: Rogue
Racial Increases:
Elf (+2)
Halfling (+2)
Human (+1)

Constitution
Measures: Health, stamina, vital force
Important for: Everyone
Racial Increases:
Dwarf (+2)
Stout halfling (+1)
Human (+1)

This method of determining ability scores enables you to create a set of three high numbers and three low ones (15, 15, 15, 8, 8, 8), a set of numbers that are above average and nearly equal (13, 13, 13, 12, 12, 12), or any set of numbers between those extremes.

4. Describe Your Character

Ability Score Point Cost

Score	Cost	Score	Cost
8	0	12	4
9	1	13	5
10	2	14	7
11	3	15	9

Once you know the basic game aspects of your character, it's time to flesh him or her out as a person. Your character needs a name. Spend a few minutes thinking about what he or she looks like and how he or she behaves in general terms.

Using the information in chapter 4, you can flesh out your character's physical appearance and personality traits. Choose your character's **alignment** (the moral

Monsters

Guidelines for understanding the information found in a monster's statistics are presented below.

Statistics

A monster's statistics, sometimes referred to as its **stat block**, provide the essential information that you need to run the monster.

Size

A monster can be Tiny, Small, Medium, Large, Huge, or Gargantuan. The Size Categories table shows how much space a creature of a particular size controls in combat. See the player's D&D basic rules or the *Player's Handbook* for more information on creature size and space.

Size Categories

Size	Space	Examples
Tiny	2½ by 2½ ft.	Imp, sprite
Small	5 by 5 ft.	Giant rat, goblin
Medium	5 by 5 ft.	Orc, werewolf
Large	10 by 10 ft.	Hippogriff, ogre
Huge	15 by 15 ft.	Fire giant, treant
Gargantuan	20 by 20 ft. or larger	Kraken, purple worm

iconic constructs. Many creatures native to the outer plane of Mechanus, such as modrons, are constructs shaped from the raw material of the plane by the will of more powerful creatures.

Dragons are large reptilian creatures of ancient origin and tremendous power. True dragons, including the good metallic dragons and the evil chromatic dragons, are highly intelligent and have innate magic. Also in this category are creatures distantly related to true dragons, but less powerful, less intelligent, and less magical, such as wyverns and pseudodragons.

Elementals are creatures native to the elemental planes. Some creatures of this type are little more than animate masses of their respective elements, including the creatures simply called elementals. Others have biological forms infused with elemental energy. The races of genies, including djinn and efreet, form the most important civilizations on the elemental planes. Other elemental creatures include azers, invisible stalkers, and water weirds.

Fey are magical creatures closely tied to the forces of nature. They dwell in twilight groves and misty forests. In some worlds, they are closely tied to the Feywild, also called the Plane of Faerie. Some are also found in the Outer Planes, particularly the planes of Arborea and the Beastlands. Fey include dryads, pixies, and satyrs.

https://dnd.wizards.com/

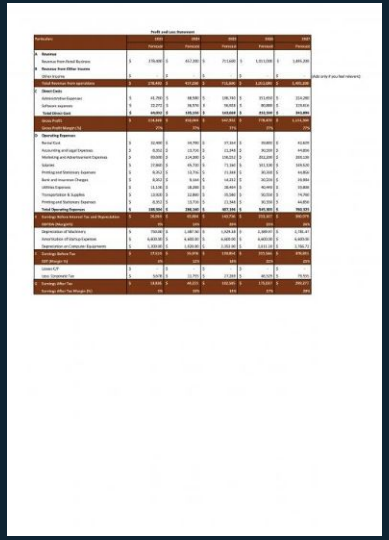
Campaign

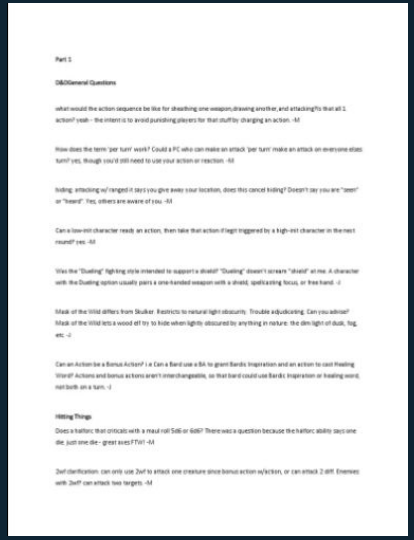
Name: _____
City: _____
Location: _____
Date: _____
Notes:

Name: _____
City: _____
Location: _____
Date: _____
Notes:

Name: _____
City: _____
Location: _____
Date: _____
Notes:

Name: _____
City: _____
Location: _____
Date: _____
Notes:





Part 1

D&D/General Questions

what would the action sequence be like for sheathing one weapon,drawing another, and attacking?Is that all 1 action? yeah - the intent is to avoid punishing players for that stuff by charging an action. -M

How does the term 'per turn' work? Could a PC who can make an attack 'per turn' make an attack on everyone elses turn? yes, though you'd still need to use your action or reaction. -M

hiding: attacking w/ ranged it says you give away your location, does this cancel hiding? Doesn't say you are "seen" or "heard". Yes, others are aware of you. -M

Can a low-init character ready an action, then take that action if/aegt triggered by a high-init character in the next round? yes. -M

Was the "Dueling" fighting style intended to support a shield? "Dueling" doesn't scream "shield" at me. A character with the Dueling option usually pairs a one-handed weapon with a shield, spellcasting focus, or free hand. -J

Mask of the Wild differs from Skulker. Restricts to natural light obscurity. Trouble adjudicating. Can you advise? Mask of the Wild lets a wood elf try to hide when lightly obscured by anything in nature. the dim light of dusk, fog, etc. -J

Can an Action be a Bonus Action? I.e Can a Bard use a BA to grant Bardic Inspiration and an action to cast Healing Word? Actions and bonus actions aren't interchangeable, so that bard could use Bardic Inspiration or healing word, not both on a turn. -J

Hitting Things

Does a halforc that criticals with a maul roll 5d6 or 6d6? There was a question because the halforc ability says one die, just one die - great axes FTW! -M

2wf clarification: can only use 2wf to attack one creature since bonus action w/action, or can attack 2 diff. Enemies with 2wf? can attack two targets -M

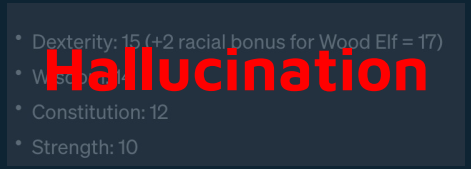
https://www.dndbeyond.com/posts/814-new-players-guide-how-to-make-awesome-handouts

01

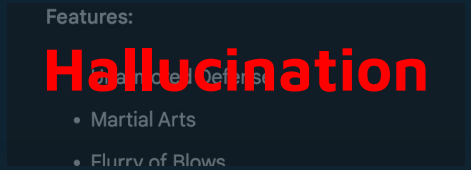
CREATING AN ASSISTANT DUNGEON MASTER

EXISTING SOLUTIONS

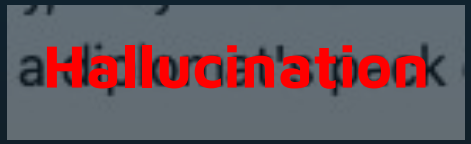
CHATGPT



BARD



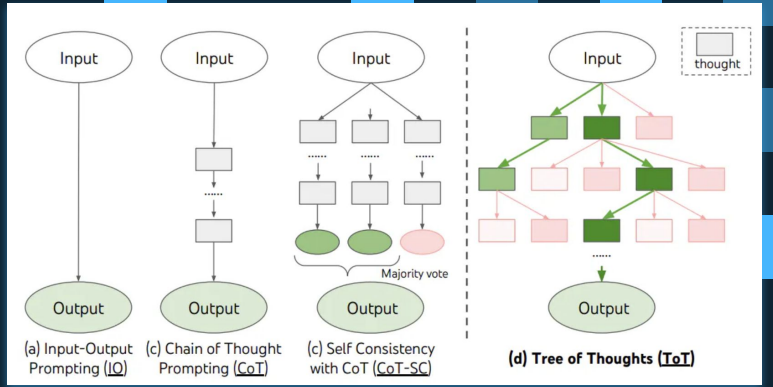
LLAMA (2)



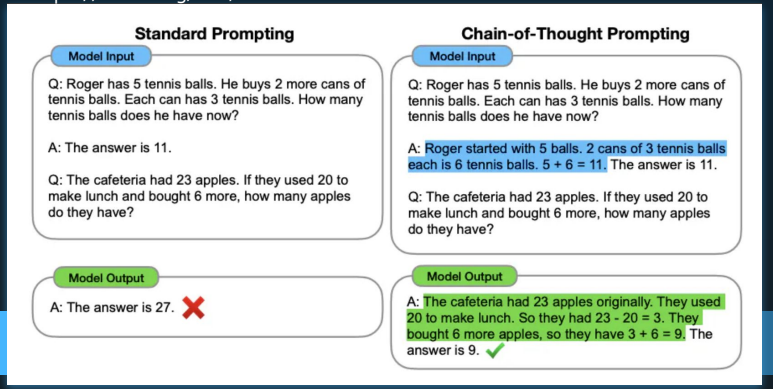
THEN WE DO PROMPT ENGINEERING

We try to hack commercial LLMS by using:

- Chain-of-thought prompting
- Few-shot prompting
- Tree of thoughts
- Retrieval augmented generation (RAG)
- Other trial and error



https://arxiv.org/abs/2305.10601



https://arxiv.org/abs/2201.11903

https://www.promptingguide.ai/techniques

THE PROBLEM

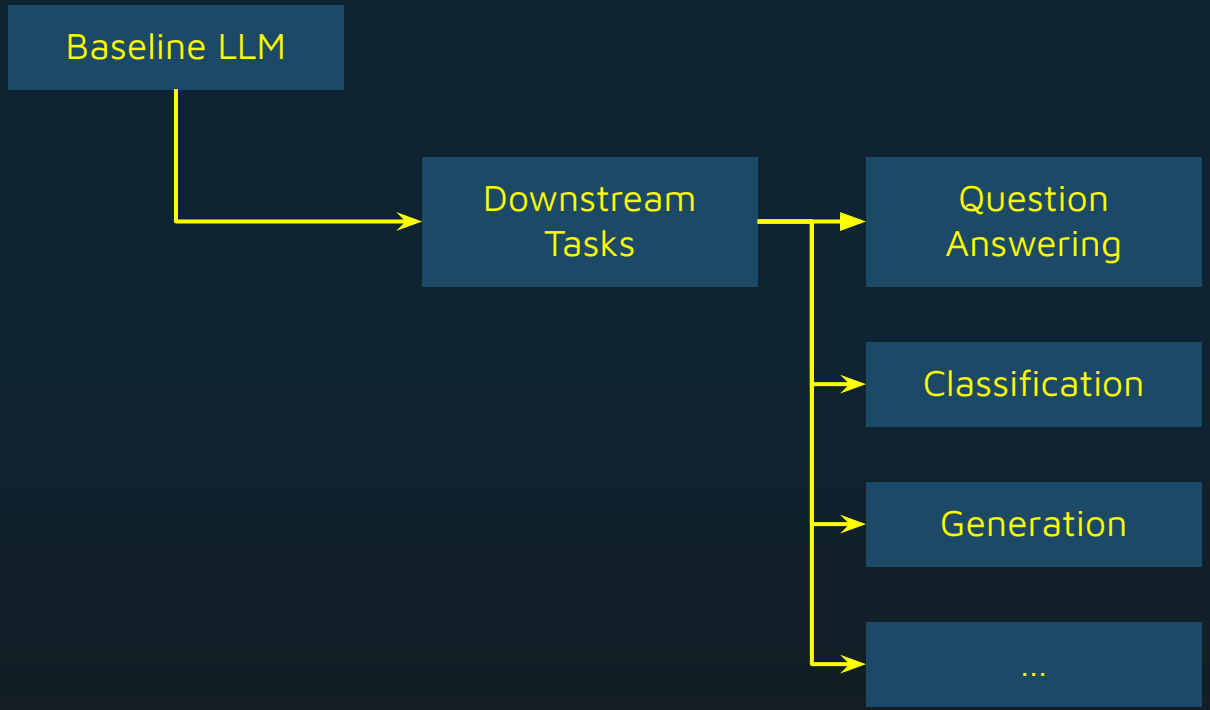
Prompt engineering and trying to solve for limited context windows may be the second step and not the first!

We can instead ask: How can we give LLMs better domain knowledge?

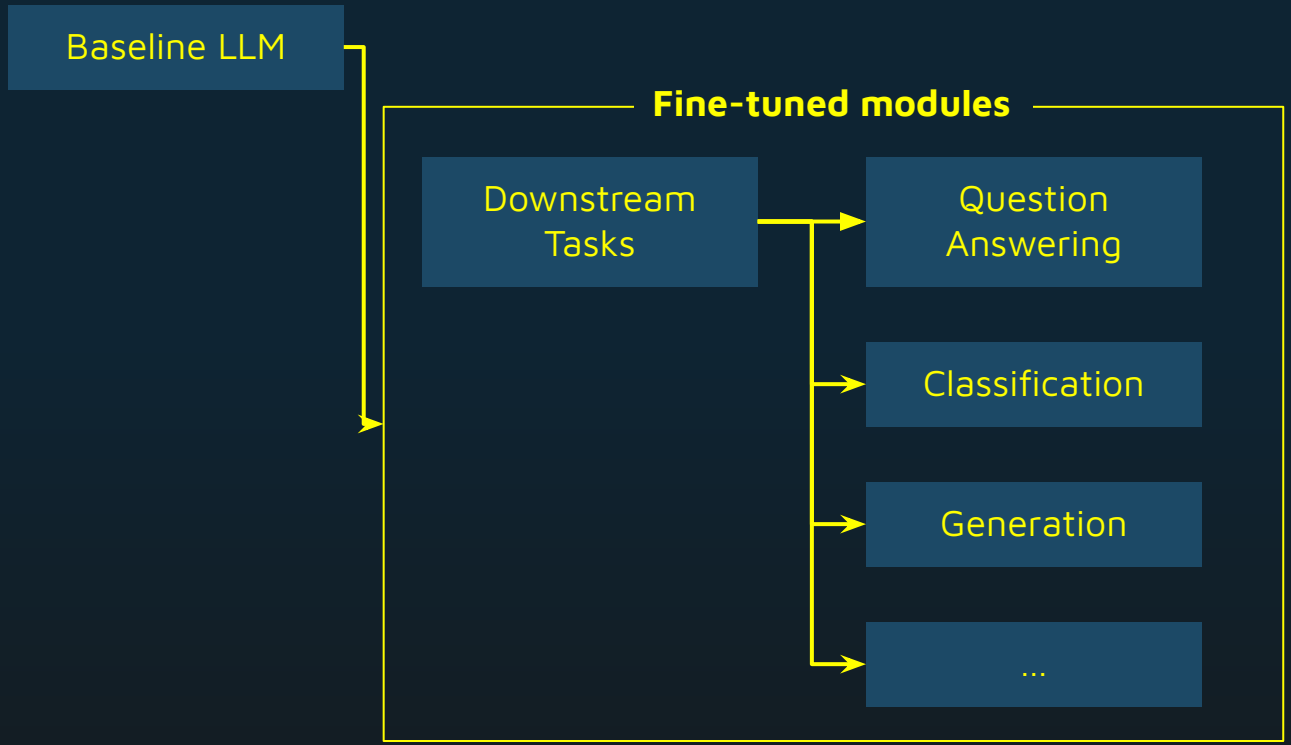
02

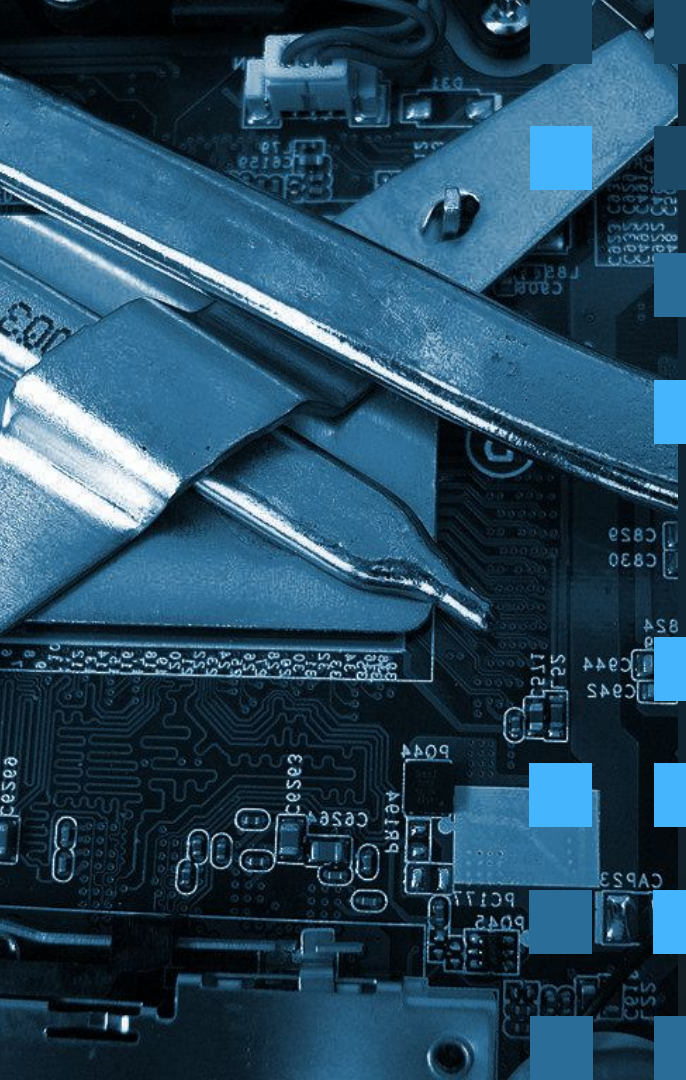
FINE-TUNING A MODEL

WHERE TO START



WHERE TO START





BUT WAIT, WHAT ABOUT GPUS?

Don’t I need a thousand of these?

LOW-RANK ADAPTATION OF LLMS

LoRA: Low-Rank Adaptation of Large Language Models

Edward Hu* **Yelong Shen*** **Phillip Wallis** **Zeyuan Allen-Zhu**
Yuanzhi Li **Shean Wang** **Lu Wang** **Weizhu Chen**
Microsoft Corporation
{edwardhu, yeshe, phwallis, zeyuana,
yuanzhil, swang, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu
(Version 2)

Abstract

An important paradigm of natural language processing consists of large-scale pre-training on general domain data and adaptation to particular tasks or domains. As we pre-train larger models, full fine-tuning, which retrains all model parameters, becomes less feasible. Using GPT-3 175B as an example – deploying independent instances of fine-tuned models, each with 175B parameters, is prohibitively expensive. We propose **L**ow-**R**ank **A**daptation, or LoRA, which freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture, greatly reducing the number of trainable parameters for downstream tasks. Compared to GPT-3 175B fine-tuned with Adam, LoRA can reduce the number of trainable parameters by 10,000 times and the GPU memory requirement by 3 times. LoRA performs on-par or better than fine-tuning in model quality on RoBERTa, DeBERTa, GPT-2, and GPT-3, despite having fewer trainable parameters, a higher training throughput, and, unlike adapters, *no additional inference latency*. We also provide an empirical investigation into rank-deficiency in language model adaptation, which sheds light on the efficacy of LoRA. We release a package that facilitates the integration of LoRA with PyTorch models and provide our implementations and model checkpoints for RoBERTa, DeBERTa, and GPT-2 at `https://github.com/microsoft/LoRA`.

1 Introduction

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB_{base} (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB_{base} (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB_{base} ($Adpt^D$)*	0.3M	$87.1_{\pm.0}$	$94.2_{\pm.1}$	$88.5_{\pm1.1}$	$60.8_{\pm.4}$	$93.1_{\pm.1}$	$90.2_{\pm.0}$	$71.5_{\pm2.7}$	$89.7_{\pm.3}$	84.4
RoB_{base} ($Adpt^D$)*	0.9M	$87.3_{\pm.1}$	$94.7_{\pm.3}$	$88.4_{\pm.1}$	$62.6_{\pm.9}$	$93.0_{\pm.2}$	$90.6_{\pm.0}$	$75.9_{\pm2.2}$	$90.3_{\pm.1}$	85.4
RoB_{base} (LoRA)	0.3M	$87.5_{\pm.3}$	**$95.1_{\pm.2}$**	$89.7_{\pm.7}$	$63.4_{\pm1.2}$	**$93.3_{\pm.3}$**	$90.8_{\pm.1}$	**$86.6_{\pm.7}$**	**$91.5_{\pm.2}$**	**87.2**
RoB_{large} (FT)*	355.0M	90.2	**96.4**	**90.9**	68.0	94.7	**92.2**	86.6	92.4	88.9
RoB_{large} ($Adpt^H$)†	0.8M	$90.3_{\pm.3}$	$96.3_{\pm.5}$	$87.7_{\pm1.7}$	$66.5_{\pm2.0}$	$94.7_{\pm.2}$	$91.5_{\pm.1}$	$72.9_{\pm2.9}$	$91.5_{\pm.5}$	86.4
RoB_{large} (LoRA)†	0.8M	**$90.6_{\pm.2}$**	$96.2_{\pm.5}$	**$90.2_{\pm1.0}$**	$68.2_{\pm1.9}$	**$94.8_{\pm.3}$**	$91.6_{\pm.2}$	**$85.2_{\pm1.1}$**	**$92.3_{\pm.5}$**	**88.6**
DeB_{XXL} (FT)*	1500.0M	91.8	**97.2**	92.0	72.0	**96.0**	92.7	93.9	92.9	91.1
DeB_{XXL} (LoRA)	4.7M	**$91.9_{\pm.2}$**	$96.9_{\pm.2}$	**$92.6_{\pm.6}$**	**$72.4_{\pm1.1}$**	**$96.0_{\pm.1}$**	**$92.9_{\pm.1}$**	**$94.9_{\pm.4}$**	**$93.0_{\pm.2}$**	**91.3**

10,000x reduction in trainable parameters for GPT-3, without any loss in performance.

Table 2: $RoBERTa_{base}$, $RoBERTa_{large}$, and $DeBERTa_{XXL}$ with different adaptation methods on the GLUE benchmark. We report the overall (matched and mismatched) accuracy for MNLI, Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for other tasks. Higher is better for all metrics. * indicates numbers published in prior works. † indicates runs configured in a setup similar to Houlsby et al. (2019) for a fair comparison.

https://arxiv.org/pdf/2106.09685.pdf%C2%A0

WHAT LORA DOES

Baseline LLM

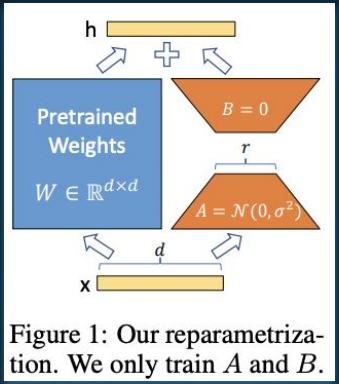


Figure 1: Our reparametrization. We only train A and B.

$$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x, y_{<t})\right)$$

Fine-tuned modules

Downstream Tasks

- Question Answering
- Classification
- Generation
- ...

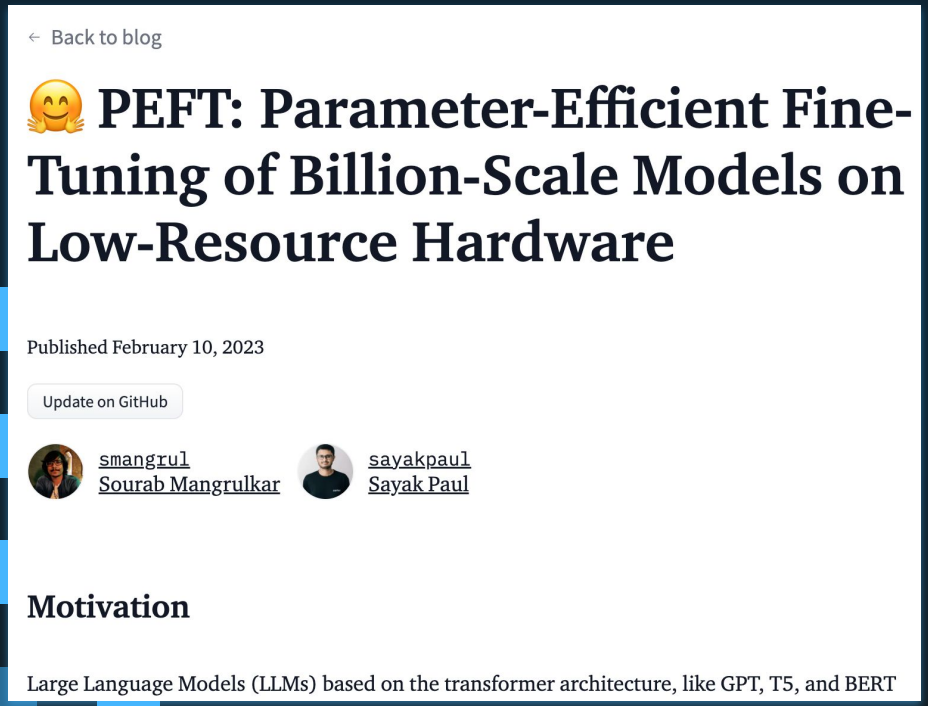
https://arxiv.org/pdf/2106.09685.pdf%C2%A0

WHY IT MATTERS

1. A pre-trained model can be shared and used to build many small LoRA modules for different tasks.
2. LoRA can boost training efficiency and lower hardware requirements up to three times by using adaptive optimizers that eliminate the need to calculate gradients or maintain optimizer states for most parameters.
3. No inference latency.
4. Interoperable with other techniques. After conducting a thorough analysis of previous research in this field, the authors discovered that this approach could be integrated with other adapter or quantization methods.

https://arxiv.org/pdf/2106.09685.pdf%C2%A0

EASY TO USE OPEN-SOURCE



← Back to blog

🤗 PEFT: Parameter-Efficient Fine-Tuning of Billion-Scale Models on Low-Resource Hardware

Published February 10, 2023

Update on GitHub

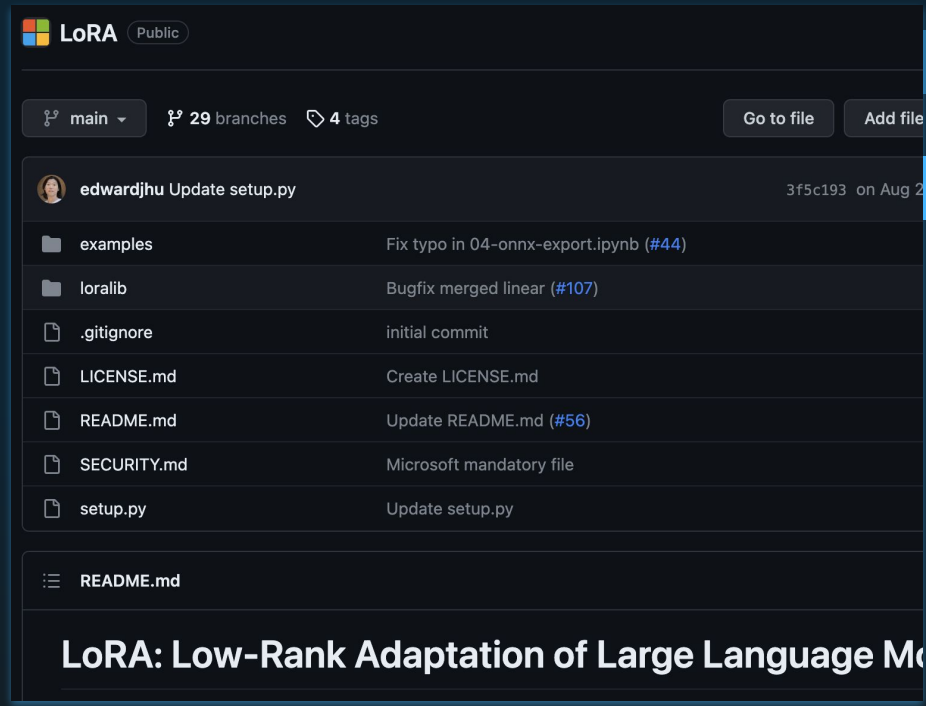
smangrul Sourab Mangrulkar

sayakpaul Sayak Paul

Motivation

Large Language Models (LLMs) based on the transformer architecture, like GPT, T5, and BERT

https://huggingface.co/blog/peft



LoRA Public

main 29 branches 4 tags Go to file Add file

edwardjhu Update setup.py 3f5c193 on Aug 2

examples	Fix typo in 04-onnx-export.ipynb (#44)
loralib	Bugfix merged linear (#107)
.gitignore	initial commit
LICENSE.md	Create LICENSE.md
README.md	Update README.md (#56)
SECURITY.md	Microsoft mandatory file
setup.py	Update setup.py

README.md

LoRA: Low-Rank Adaptation of Large Language Mo

https://github.com/microsoft/LoRA

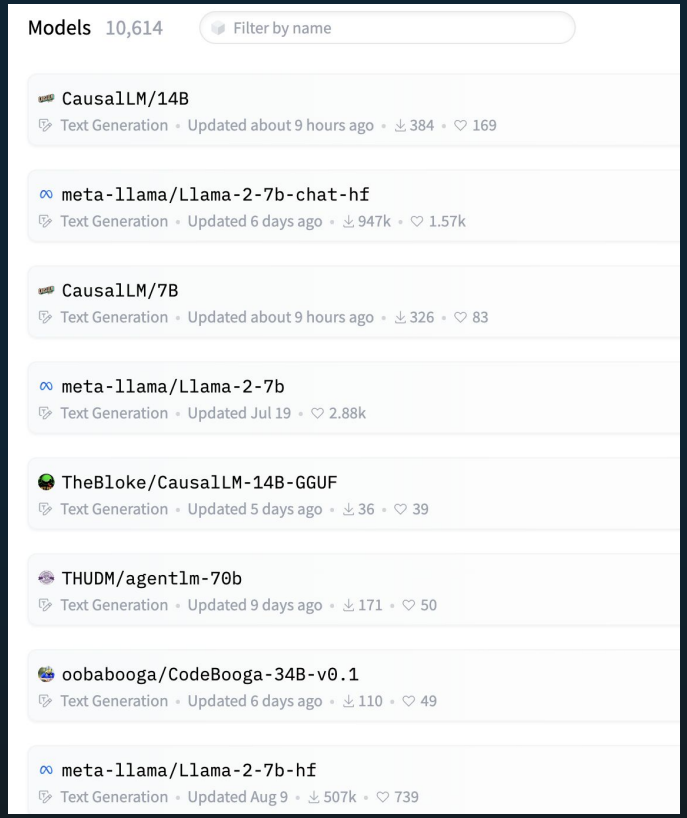
STEP 1: DATA

1. Select ethically and responsibly (no proprietary D&D books were used for training)
2. Engineer into the right format (**HINT**: **Use an LLM!**)
3. Human expert review (Dungeon Master)

Sample JSON outputs:

[{"instruction": " What is the name of the game mentioned in the text? ",
"input": "", "output": " Dungeons and Dragons. "},
{"instruction": " What is the meaning of the numbers and signs listed after the six ability scores? ",
"input": "", "output": " They show the score and the modifier for each ability. "},
{"instruction": " What are the skills listed under the behir's stat block? ",
"input": "",
"output": " Perception and Stealth. "},....]

STEP 2: MODEL SELECTION



Start small - for this example we start with Llama 7B open source, which we can use LoRA to fit into a single free-tier Colab GPU

https://huggingface.co/models?other=llama

STEP 3: FINE-TUNING

1. Set-up (Colab in this example)
2. Size the training job
3. Read Llama model (in this example: 7B-hf)
4. Configure LoRA parameters

1

```
import os
import sys

import torch
import torch.nn as nn
import bitsandbytes as bnb
from datasets import load_dataset
import transformers
from transformers import LlamaForCausalLM, LlamaTokenizer
from peft import (
    prepare_model_for_int8_training,
    LoraConfig,
    get_peft_model,
    get_peft_model_state_dict,
)
```

2

```
# Set random seed for reproducibility
RANDOM_SEED = 1234
transformers.set_seed(RANDOM_SEED)

# Fit into Kaggle T4*2
MICRO_BATCH_SIZE = 8
BATCH_SIZE = 128
GRADIENT_ACCUMULATION_STEPS = BATCH_SIZE // MICRO_BATCH_SIZE
EPOCHS = 3  # One epoch takes ~6 hours, and 2 epochs may exceed Kaggle 12-hour limit
LEARNING_RATE = 3e-4  # Following stanford_alpaca
CUTOFF_LEN = 512  # 256 accounts for about 96% of the data. Shorter input, faster training/less VRAM
LORA_R = 16  # Some LoRA parameters
LORA_ALPHA = 16
LORA_DROPOUT = 0.05
VAL_SET_SIZE = 2000
TARGET_MODULES = [
    'q_proj',
    'v_proj',
    'k_proj',
    'o_proj'
]
DATA_PATH = 'a
OUTPUT_DIR = '
```

3

4

```
# LoRA config.
config = LoraConfig(
    r=LORA_R,
    lora_alpha=LORA_ALPHA,
    target_modules=TARGET_MODULES,
    lora_dropout=LORA_DROPOUT,
    bias='none',
    task_type='CAUSAL_LM',
)
```

STEP 3: FINE-TUNING, Cont'd.

1. Generate and tokenize prompts
2. Training/evaluation data split
3. Train the model & peft setup
4. Save the fine-tuned model

1

```python
def generate_and_tokenize_prompt(data_point):
    """This function masks out the labels for the input, so that our loss is computed on
    response."""
    if data_point['input']:
        user_prompt = 'Below is an instruction that describes a task, paired with an inp
                      'provides further context. Write a response that appropriately com
                      'the request.\n\n'
        user_prompt += f'### Instruction:\n{data_point["instruction"]}\n\n'
        user_prompt += f'### Input:\n{data_point["input"]}\n\n'
        user_prompt += f'### Response:\n'
    else:
        user_prompt = 'Below is an instruction that describes a task. Write a response
                      'appropriately completes the request.'
        user_prompt += f'### Instruction:\n{data_point["instruction"]}\n\n'
        user_prompt += f'### Response:\n'

    # Count the length of prompt tokens
    len_user_prompt_tokens = len(tokenizer(user_prompt,
                                           truncation=True,
                                           max_length=CUTOFF_LEN + 1,
                                           padding='max_length')['input_ids'])
    len_user_prompt_tokens -= 1  # Minus 1 (one) for eos token

    # Tokenise the input, both prompt and output
    full_tokens = tokenizer(
        user_prompt + data_point['output'],
        truncation=True,
        max_length=CUTOFF_LEN + 1,
        padding='max_length',
    )['input_ids'][:-1]
    return {
        'input_ids': full_tokens,
        'labels': [-100] * len_user_prompt_tokens + full_tokens[len_user_prompt_tokens
        'attention_mask': [1] * (len(full_tokens)),
    }
```

2

```python
# Train/val split
if VAL_SET_SIZE > 0:
    train_val = data['train'].train_test_split(
        test_size=VAL_SET_SIZE, shuffle=False, seed=RANDOM_SEED
    )
    train_data = train_val['train'].map(generate_and_tokenize_p
    val_data = train_val['test'].map(generate_and_tokenize_prom
else:
    train_data = data['train'].map(generate_and_tokenize_prompt)
    val_data = None
```

3

```python
# HuggingFace Trainer
trainer = transformers.Trainer(
    model=model,
    train_dataset=train_data,
    eval_dataset=val_data,
    args=transformers.TrainingArguments(
        seed=RANDOM_SEED,  # Reproducibility
        data_seed=RANDOM_SEED,
        per_device_train_batch_size=MICRO_BATCH_SIZE,
        gradient_accumulation_steps=GRADIENT_ACCUMULATION_STEPS,
        warmup_steps=100,
        num_train_epochs=EPOCHS,
        learning_rate=LEARNING_RATE,
        fp16=True,
        logging_steps=20,
        evaluation_strategy='steps' if VAL_SET_SIZE > 0 else 'no',
        save_strategy='steps',
        save_steps=50,
        eval_steps=50 if VAL_SET_SIZE > 0 else None,
        output_dir=OUTPUT_DIR,
        save_total_limit=3,
        load_best_model_at_end=True if VAL_SET_SIZE > 0 else False,
        ddp_find_unused_parameters=False if ddp else None,
    ),
    data_collator=transformers.DataCollatorForLanguageModeling(tokenizer, mlm=False),
)
model.config.use_cache = False
```

4

```python
# PEFT setup
old_state_dict = model.state_dict
model.state_dict = (
    lambda self, *_, **__: get_peft_model_state_dict(self, old_state_dict
).__get__(model, type(model))
```

STEP 4: INFERENCE

1. Load model
2. Create prompt
3. Generate response
4. Formatting output (tokenizer)
5. Conversation storage/retrieval (vector DB, llama-index)

2

```
"""## Create Prompt"""

prompt_base = f"""Answer the generic questions without any help form the context.
But your goals are to give answer based on the relevant information given to you.

### READ FROM HERE FOR THE CONTEXT:
<<CONTEXTT>>

### PREVIOUS CONVERSATION:
<<CONVERSATIONN>>

### Instruction: <<QUESTION>>

Now answer here as:
### Response:
"""

PROMPT_TEMPLATE = f"""
Below is an instruction that describes a task. Write a response that appropriately completes the r

### Instruction: <<QUESTION>>
```

3

```
def generate_response(prompt: str, model: PeftModel) -> GreedySearchDecoderOnlyOutput:
    encoding = tokenizer(prompt, return_tensors="pt")
    input_ids = encoding["input_ids"].to(DEVICE)

    generation_config = GenerationConfig(
        temperature=0.1,
        top_p=0.75,
        repetition_penalty=1.1,
    )
    with torch.inference_mode():
        return model.generate(
            input_ids=input_ids,
            generation_config=generation_config,
            return_dict_in_generate=True,
            output_scores=True,
            max_new_tokens=256,
        )
```

4

```
def format_response(response: GreedySearchDecoderOnlyOutput) -> str:
    decoded_output = tokenizer.decode(response.sequences[0])
    # print(decoded_output)
    response = decoded_output.split("### Response:")[1].strip()
    return "\n".join(textwrap.wrap(response))

def ask_alpaca(prompt: str, model: PeftModel) -> str:
    prompt = create_prompt(prompt)
    response = generate_response(prompt, model)
    return format_response(response)
```

STEP 5: TESTING

1. Asking out of distribution questions
2. Unit testing to make sure outputs do not change (depending on stability of model)
3. Iterate on data and fine-tuning based on results

STEP 6: DEPLOYMENT OPTIONS

1. For fast iteration and testing, a notebook such as Colab is a good start.

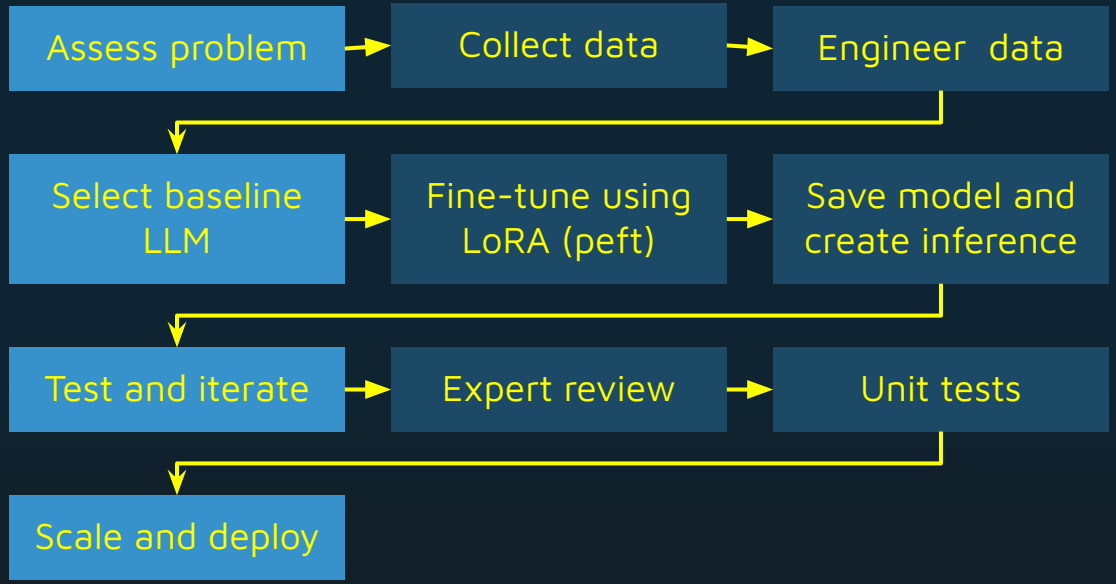
2. When you have a front-end and more robust applications around it, consider kubernetes and containerizing your model to deploy: HTTPS, pub-sub, batch depending on what downstream task you need the module to do and latency requirements.

3. Consider adding a vector DB for quick caching, auditing, and longer-term information retrieval based on how the assistant DM helps shape the world ("Remember this...").

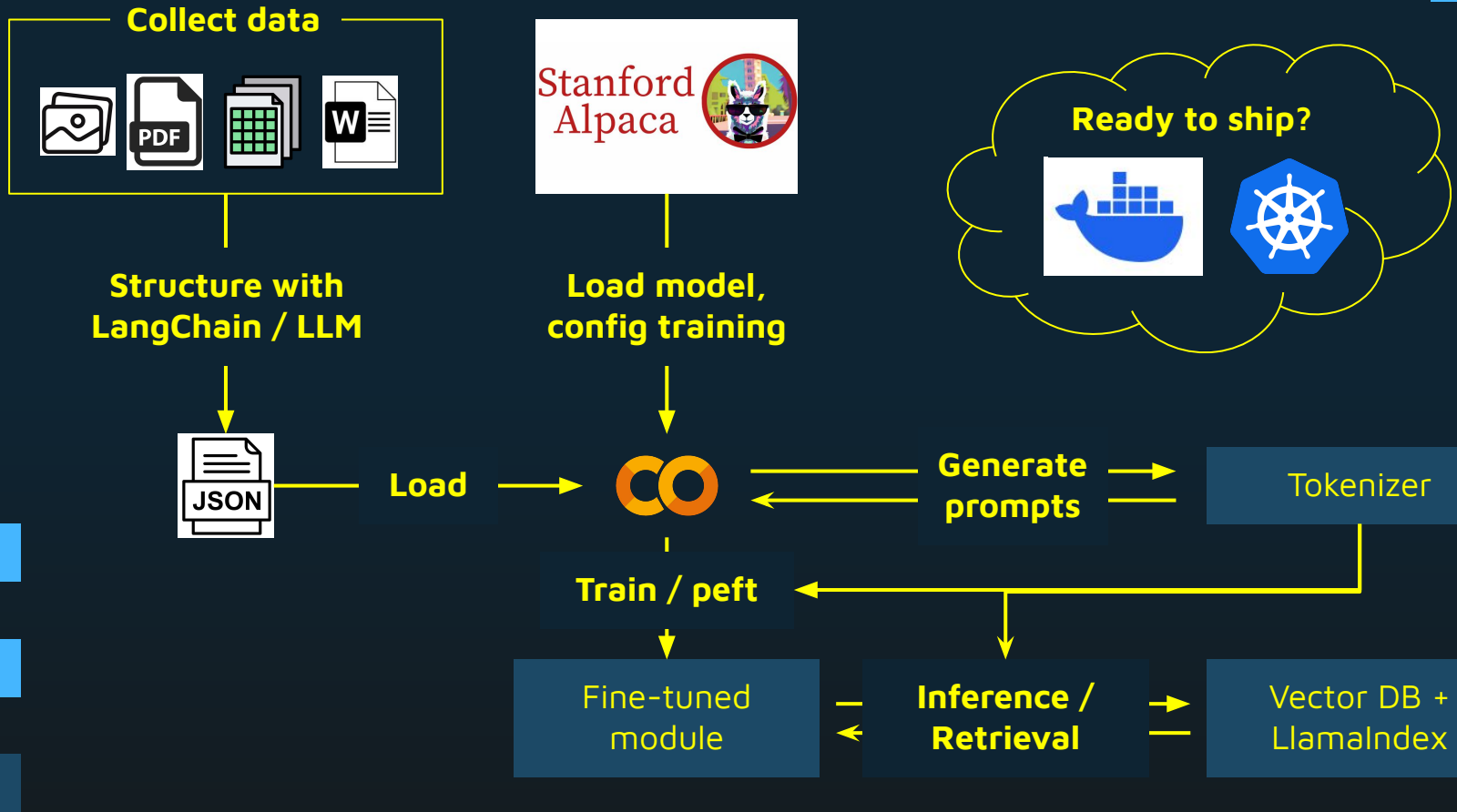
03

PUTTING IT ALL TOGETHER

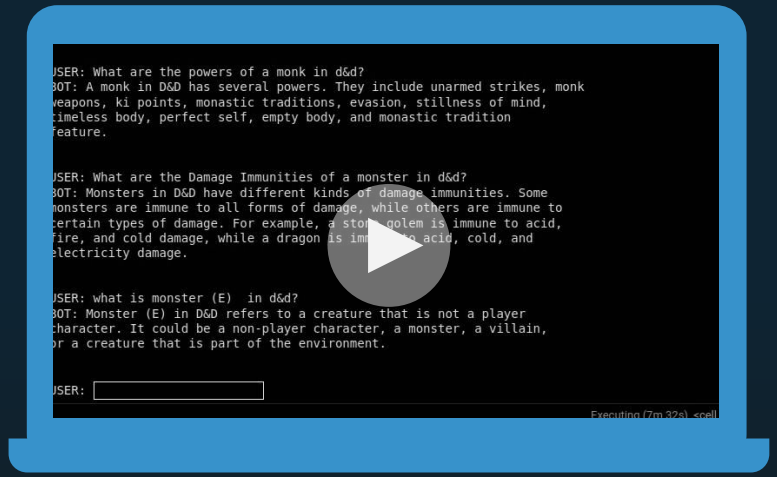
SUMMARY


Assess problem
Collect data
Engineer data
Select baseline LLM
Fine-tune using LoRA (peft)
Save model and create inference
Test and iterate
Expert review
Unit tests
Scale and deploy

SUMMARY, Cont'd.



SCUFFED DM IS BORN



So this is the story of both how “Scuffed DM” was born, but also how you can bootstrap your own fine-tuning projects.

Coming soon to an 8-bit front-end near you.

But wait, why scuffed?

DOWNSTREAM TASK: INFORMATION RETRIEVAL/Q&A

```
USER: What are the powers of a monk in d&d?
BOT: A monk in D&D has several powers. They include unarmed strikes, monk
weapons, ki points, monastic traditions, evasion, stillness of mind,
timeless body, perfect self, empty body, and monastic tradition
feature.


USER: What are the Damage Immunities of a monster in d&d?
BOT: Monsters in D&D have different kinds of damage immunities. Some
monsters are immune to all forms of damage, while others are immune to
certain types of damage. For example, a stone golem is immune to acid,
fire, and cold damage, while a dragon is immune to acid, cold, and
electricity damage.


USER: what is monster (E)  in d&d?
BOT: Monster (E) in D&D refers to a creature that is not a player
character. It could be a non-player character, a monster, a villain,
or a creature that is part of the environment.


USER:
```

Executing (7m 32s) <cell line: 12> > raw_inp

DOWNSTREAM TASK: GENERATION

USER: create a monk character based on base attributes
BOT: Name: Master Kung Fu Race/Class: Human/Monk Alignment: Lawful Good
Base Attributes: STR 12, DEX 14, CON 12, INT 12, WIS 12, CHA 12 Base
Skills: Acrobatics (Dex), Athletics (Str), History (Int), Insight
(Wis), Religion (Wis), Stealth (Dex) Starting Equipment: Dagger
(Simple Melee Weapon), Backpack (Explorer's Pack), 10 Darts

USER:

https://www.imdb.com/title/tt0133093/mediaindex

DOWNSTREAM TASK: STEP-BY-STEP COMBAT

Step 1: Initial Positioning

* Monk: HP 120/120, AC 18, Attack Bonus +7
* Paladin: HP 90/90, AC 16, Attack Bonus +5

Step 2: Paladin Casts Bless

* Monk: HP 120/120, AC 18, Attack Bonus +7
* Paladin: HP 90/90, AC 16, Attack Bonus +5, bless active

Step 3: Monk Dodges Paladin's Attack

* Monk: HP 120/120, AC 18, Attack Bonus +7
* Paladin: HP 90/90, AC 16, Attack Bonus +5

Step 4: Monk Lands a Flurry of Punches and Kicks

* Monk: HP 120/120, AC 18, Attack Bonus +7
* Paladin: HP 87/90, AC 16, Attack Bonus (damaged by monk's flurry)

Step 5: Paladin Casts Healing Word

* Monk: HP 120/120, AC 18, Attack Bonus +7
* Paladin: HP 90/90, AC 16, Attack Bonus +5 (healed by healing word)

Step 6: Monk Lands a Devastating Finishing Blow

* Monk: HP 120/120, AC 18, Attack Bonus +7
* Paladin: HP 63/90, AC 16, Attack Bonus +5 (damaged by monk's finishing blow)

https://dnd.wizards.com/; https://www.imdb.com/title/tt0133093/mediaindex

DOWNSTREAM TASK: STEP-BY-STEP COMBAT

Step 1: Initial Positioning

* Monk: HP 120/120, AC 18, Attack Bonus +7
* Paladin: HP 90/90, AC 16, Attack Bonus +5

Step 2: Paladin Casts Bless

* Monk: HP 120/120, AC 18, Attack Bonus +7
* Paladin: HP 90/90, AC 16, Attack Bonus +5, bless active

Step 3: Monk Dodges Paladin's Attack

* Monk: HP 120/120, AC 18, Attack Bonus +7
* Paladin: HP 90/90, AC 16, Attack Bonus +5

Step 4: Monk Lands a Flurry of Punches and Kicks

* Monk: HP 120/120, AC 18, Attack Bonus +7
* Paladin: HP 87/90, AC 16, Attack Bonus (damaged by monk's flurry)

Step 5: Paladin Casts Healing Word

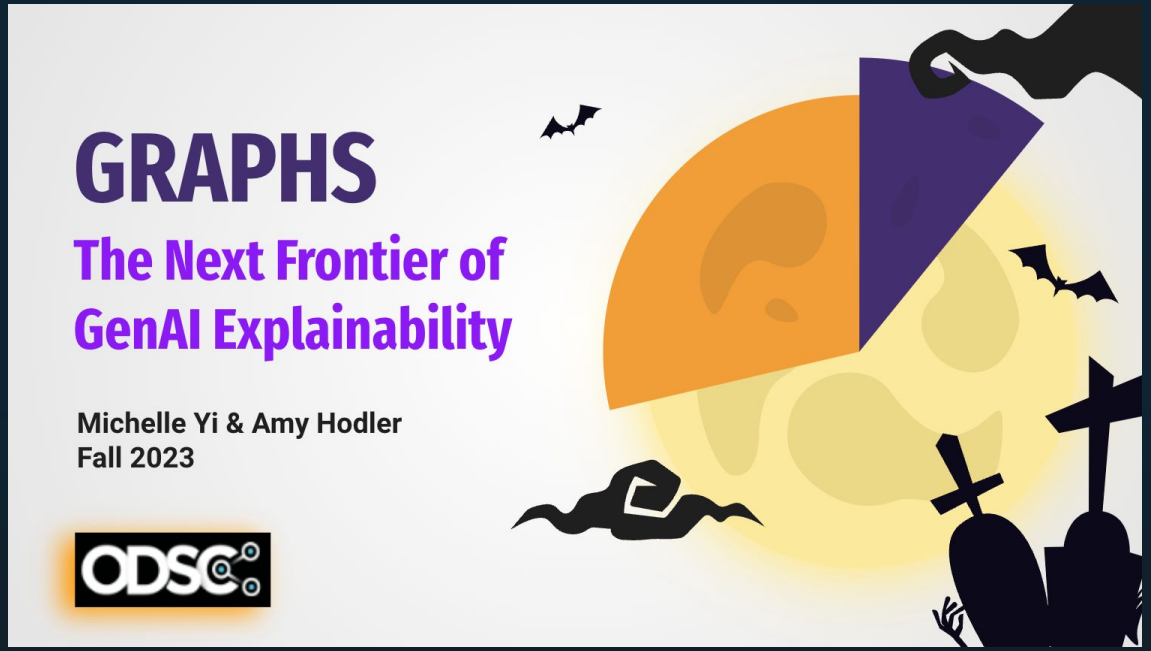
* Monk: HP 120/120, AC 18, Attack Bonus +7
* Paladin: HP 90/90, AC 16, Attack Bonus +5 (healed by healing word)

Step 6: Monk Lands a Devastating Finishing Blow

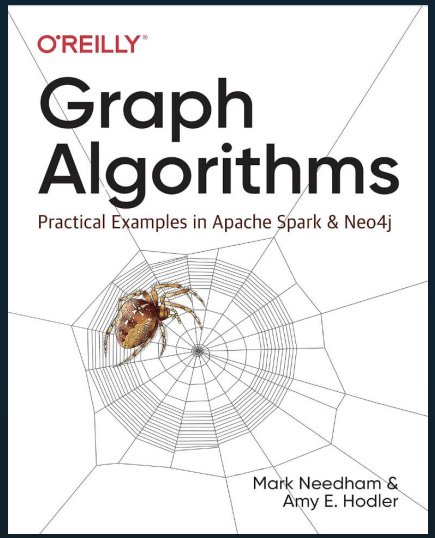
* Monk: HP 120/120, AC 18, Attack Bonus +7
* Paladin: HP 63/90, AC 16, Attack Bonus +5 (damaged by monk's finishing blow)

https://dnd.wizards.com/; https://www.imdb.com/title/tt0133093/mediaindex

WORKSHOP ON CAUSAL GRAPHS



3:45 PM Regency C



THANKS

Do you have any questions or want to collaborate?

Reach out below:

michelle@generationship.ai
https://www.linkedin.com/in/michelleyulleyi/