Women in Data Science Worldwide | Puget Sound

Finetune LLMs

Riya Joshi

Data Scientist 2 – Microsoft

Agenda

What are LLMs?

Why LLMs?

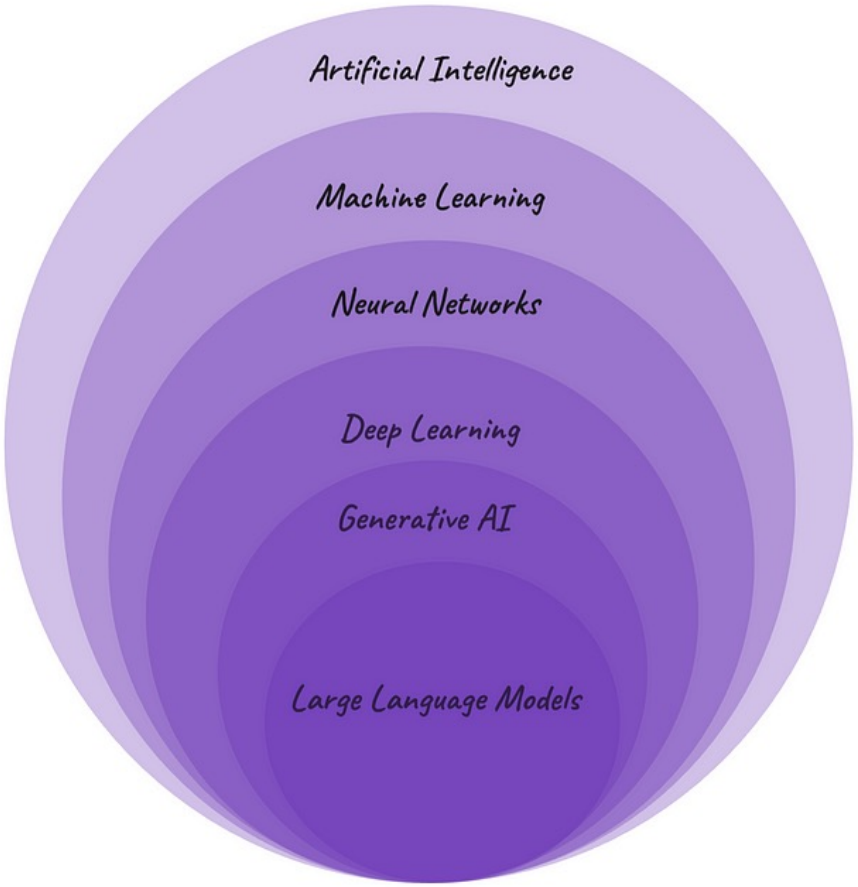
How to use LLMs

Demo on Finetuning

Prerequisites : Working knowledge of neural networks and NLP

What are LLMs ?

Background



Src: https://medium.com/womenintechnology/ai-c3412c5aa0ac

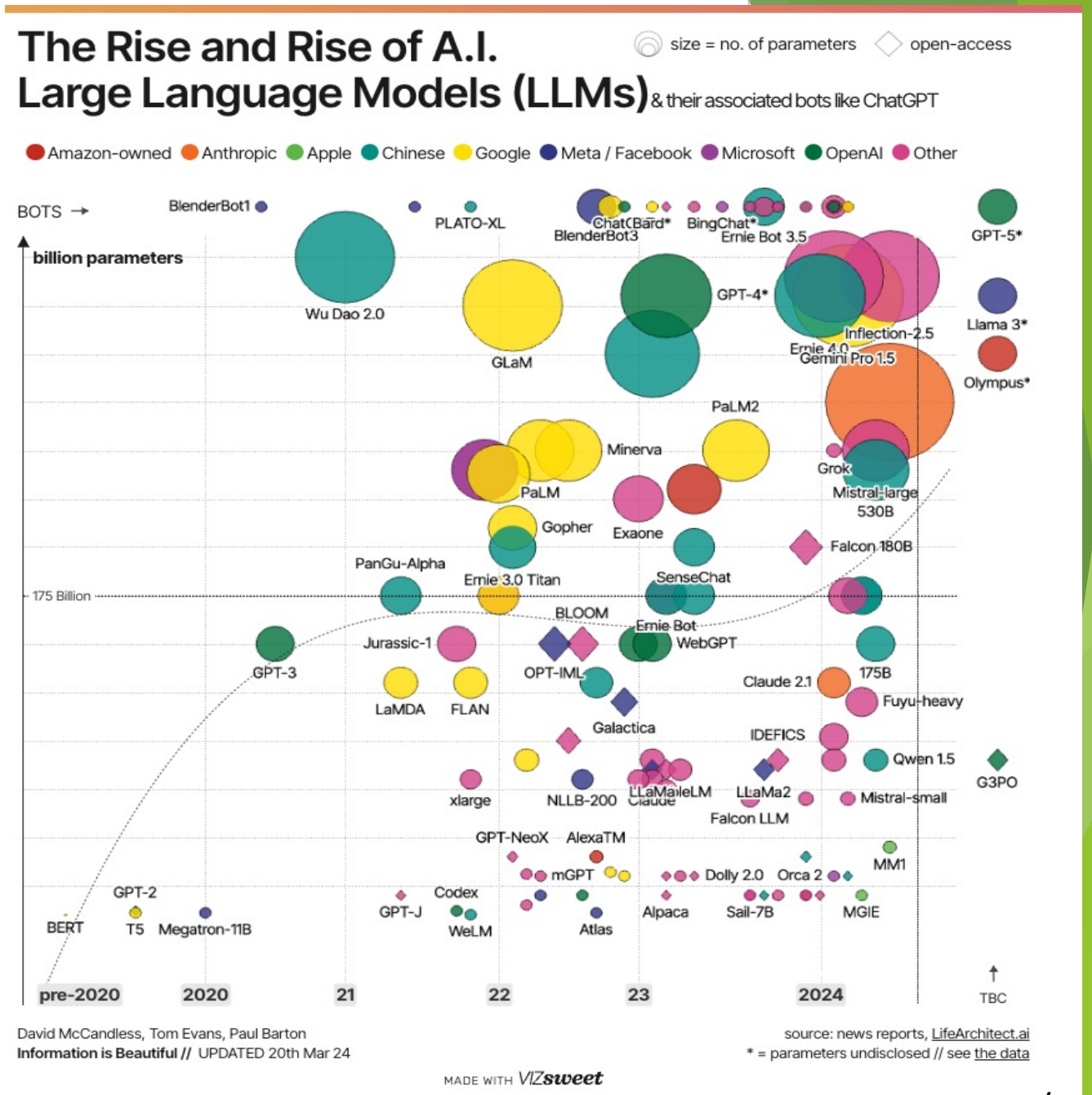
- Artificial Intelligent : Computer Science discipline that aim to create machines that can solve tasks with human-level performance.
  - [Subpart] AGI (artificial general intelligence) - Can solve tasks *like* human being *without manual intervention*.
- Machine Learning : Trains a model on input data. Model learns from pattern in input data and make predictions on new unseen data.
- Neural Networks : Subset of machine learning. Mimics the behavior of biological neurons. The neurons are connected to one another in layers. Input layers, hidden layers and output layers. Input flows from one direction to another.
- Deep Learning : Neural networks with several hidden layers. This depth of layers provide the ability to learn intricated patterns and solve complex tasks.
- Generative AI : Type of AI that can generate content – text, audio, images etc.
- Large Language models : Form of Generative AI that can generate human like text after it has been trained on extensive data during pretraining.

LLM Architecture

- Huge Deep learning models with billions of parameters.
- Transformer based models that use self attention to predict the next word [learn more]
- Trained on vast amount of data across the web
- Long story short – LLMs can learn and understand grammar, language and context very well

LLMs evolution

- The size of LLMs skyrocketed after 2022.
- These LLMs are too big to handle.
- Smaller versions like Mistral -8B, Llama3-8B and Phi-3-8B have gained popularity because of ability to be finetuned



https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt/

WHY LLMs?

Popular Use Cases

- Content Generation
- Chat Bots/Assistants
- Language Translation
- Information Retrieval
- Text Summarization

HOW TO USE LLMs ?

- **Zero Shot learning:** Prompting the model to give desired results

Prompt:

```
Classify the text into neutral, negative or positive.
Text: I think the vacation is okay.
Sentiment:
```

- **Few Shot learning:** Guiding the model with some examples of input and desired outputs to generate similar outputs for unseen data.

Prompt:

```
A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word
whatpu is:
We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word
farduddle is:
```

- **Domain adaptation:** Impart domain specific information to LLMs that was not learnt during pretraining

Domain adaptation

- LLMs are pretrained on huge corpus of publicly accessible data – Wikipedia articles/books/blogs etc.
- LLMs understand language and context well but might lack in domain knowledge for niche tasks. For example:
  - Chatbot trained on generic data might not answer appropriately for specific health records for patients.
  - Generic LLM did not have access to company specific proprietary data during training and hence can't be effectively utilized.
  - LLMs tend to hallucinate (make up information) for complex tasks or tasks they don't understand and can lead to inaccurate results even after few shot prompting.

Types of Domain Adaptation

Domain specific pre-training

- Re-train the LLM model from scratch on domain specific data.
- Very costly in terms of compute and training time

Domain specific finetuning

- Finetune the pre-trained model on smaller domain specific data
- Only few parameters are updated
- Cost effective and faster method to get LLMs to adapt to new domains and data

Retriever Augmented Generation (RAG)

- No training required
- Nonparametric/external knowledge base is added to provide additional context to the LLMs.
- Read more in detail from my blog

FINETUNING

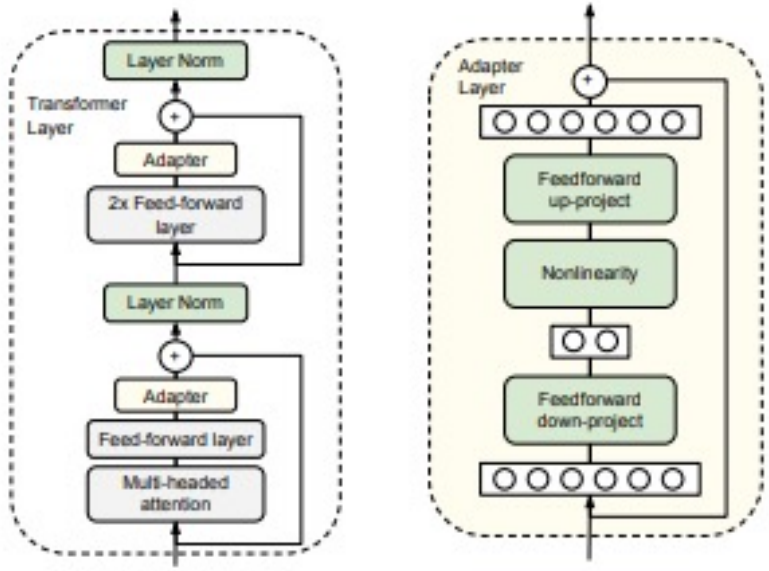
Finetuning allows to leverage existing model and its capability to comprehend language and adapt it to solve specific tasks.

- Full finetuning: finetuning all the model parameters on domain specific data.
  - Can be very expensive as LLMs have billions of parameters.
  - Can lead to catastrophic forgetting, i.e. losing reacquired knowledge from pre-training
- Parameter efficient Finetuning: Finetune only few parameters
  - Cheaper and less computer intensive
  - Only few parameters are updated that eliminates the risk of catastrophic forgetting and adapt the model to target task.

Parameter Efficient Finetuning (PEFT)

Adapters

- Adapters are one of the most prominent ways to achieve parameter efficient finetuning.
- Adapters are new modules added between layers of pretrained network.
- Only new parameters are trained, freezing the original parameters.
- Model remembers the previous tasks and use small number of parameters to learn the new task.

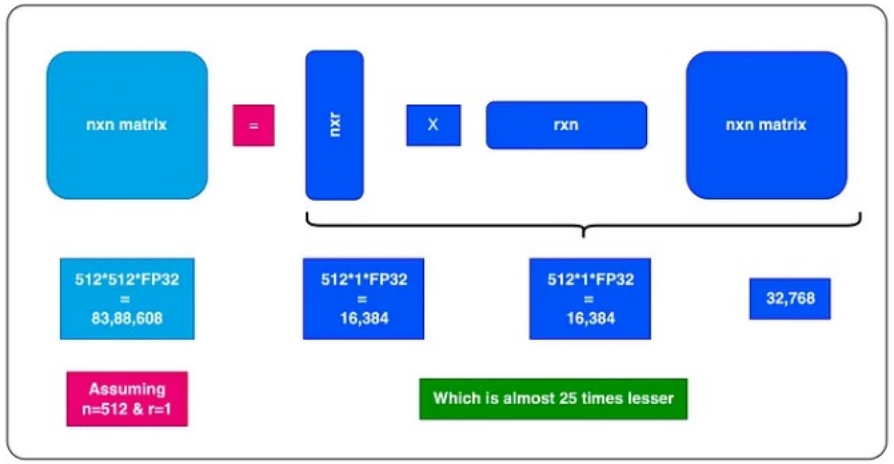


https://arxiv.org/pdf/1902.00751

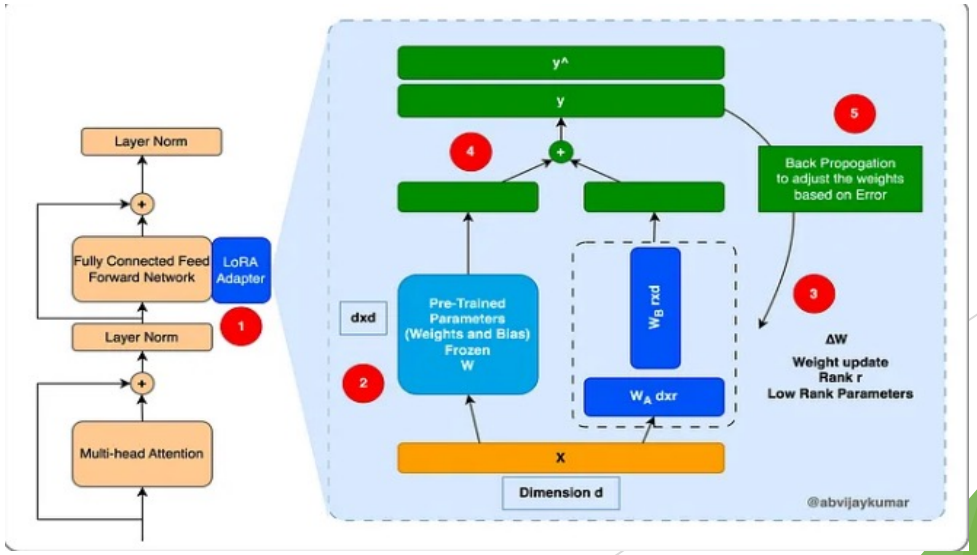
Type of adapters

- LoRA (Low-rank Adaptation)
  - LoRA converts the original matrix to low rank matrices, reducing the parameters.
  - It then just updates low rank matrices parameters, keeping the original matrix frozen
- QLoRA (Quantized LoRA)
  - Same as LoRA except the data types changes from FP32 to int4

Reduced parameters after LoRA



Finetuning with LoRA



https://abvijaykumar.medium.com/fine-tuning-llm-parameter-efficient-fine-tuning-peft-lora-qlora-part-1-571a472612c4

DEMO

- Use case : Binary Classifier to detect sarcasm
- Compare few shot prompting v/s finetuning on mistral 7B model
- Full code : Colab
- Note: to replicate the code and finetune again, atleast 1 T4 GPU is required

Load Dataset

```
train_dataset = Dataset.from_pandas(train_data_subset)
test_dataset = Dataset.from_pandas(test_data_subset)
print(train_dataset)
print(test_dataset)
```

```
Dataset({
    features: ['label', 'comment', 'parent_comment'],
    num_rows: 1000
})
Dataset({
    features: ['label', 'comment', 'parent_comment'],
    num_rows: 100
})
```

Data Example:

Parent comment:

*The article itself is only worse than your summary of it. A grubby little commentary by someone who still believes Orwell got it right when he though his prophesy was of a *Socialist* society, and not a warning about Capitalism as it is seen today.*

Comment:

we should definitely run away from 'tyranny' into the hands of capitalism

Label:

1 (Sarcasm)

Load base model

Load 7B mistral model and the quantization configuration. Quantization helps to finetune on limited GPU

```
## Base model - Pretrained LLM you want to eventually finetune
base_model_name = 'mistralai/Mistral-7B-v0.1'

## Quantization Config
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.bfloat16
)
## Load the pretrained model after quantization
model = AutoModelForCausalLM.from_pretrained(base_model_name, quantization_config=bnb_config)
```

Few shot Prompting

```
def generate_eval_prompt(data_point):
    full_prompt =f"""You are a sarcasm detection bot for social media posts. Your task is to assess the comment and its
    Sarcasm
    Not Sarcasm

    ####
    Here are some examples:
    comment: Most of the Bernie people got what they wanted on the platform, or most of it.
    reply: dont trigger them with facts!
    Category: Sarcasm

    comment: The Dallas Morning News: What you need to know about the enemies of the American people the president warne
    reply: What a brilliant and well written article that was.
    Category: Not Sarcasm

    If the text doesn't fit into any of the above categories, classify it as:
    Not Sarcasm
    <<<
    comment:
    {data_point["parent_comment"]}
    reply:
    {data_point["comment"]}
    >>>
    Catgeory:
    """
    return full_prompt
```

- Prompt should be concise and clear
- Demonstrate the task by giving examples and expected output
- Separate sections of instructions, examples and input data clearly

Finetuning code

```python
# based on config
training_args = transformers.TrainingArguments(
    fp16=False, # specify bf16=True instead when training on GPUs that support bf16
    do_eval=False,
    bf16=False,
    optim="paged_adamw_8bit",
    #evaluation_strategy="epoch",
    gradient_accumulation_steps=8,
    #gradient_checkpointing=True,
    #gradient_checkpointing_kwargs={"use_reentrant": False},
    learning_rate=2.0e-05,
    log_level="info",
    weight_decay=0.001,
    logging_steps=10,
    logging_strategy="steps",
    lr_scheduler_type="constant",
    # max_steps=1000000,
    num_train_epochs=4,
    output_dir=output_dir,
    overwrite_output_dir=True,
    per_device_eval_batch_size=1, # originally set to 8
    per_device_train_batch_size=1, # originally set to 8
    # push_to_hub=True,
    # hub_model_id="zephyr-7b-sft-lora",
    # hub_strategy="every_save",
    # report_to="tensorboard",
    save_strategy="steps",
    save_steps=1000,
    seed=42,
    warmup_ratio=0.3
)

# based on config
config = LoraConfig(
    r=8,
    lora_alpha=16,
    target_modules=[
        "q_proj",
        "k_proj",
        "v_proj",
        "o_proj",
        "gate_proj",
        "up_proj",
        "down_proj",
        "lm_head",
    ],
    bias="none",
    lora_dropout=0.05,  # Conventional
    task_type="CAUSAL_LM",
)
```

Setting up the finetuning hyperparameters like learning rate, optimizer etc.

LoRA config setup

```python
trainer = SFTTrainer(
        model=model,
        #model_init_kwargs=model_kwargs,
        args=training_args,
        train_dataset=tokenized_train_dataset,
        #eval_dataset=eval_dataset,
        dataset_text_field="text",
        tokenizer=tokenizer,
        packing=False,
        peft_config=config,
        max_seq_length=512
    )
trainer.train()
```

Trainer class that takes above configurations

Evaluation

Few shot Prompting

	precision	recall	f1-score	support
0	0.67	0.16	0.26	50
1	0.54	0.74	0.62	50
2	0.00	0.00	0.00	0
accuracy			0.45	100
macro avg	0.40	0.30	0.29	100
weighted avg	0.60	0.45	0.44	100

Predicted Labels:

- 0: Not Sarcasm
- 1: Sarcasm
- 2: None (Garbage values)

Produces lots of garbage outputs

Example:

Parent comment:
The article itself is only worse than your summary of it. A grubby little commentary by someone who still believes Orwell got it right when he though his prophesy was of a *Socialist* society, and not a warning about Capitalism as it is seen today.

Comment:
we should definitely run away from 'tyranny' into the hands of capitalism

Output : саркам (None)
Actual : Sarcasm

Finetune model

	precision	recall	f1-score	support
0	0.73	0.94	0.82	50
1	0.92	0.66	0.77	50
accuracy			0.80	100
macro avg	0.83	0.80	0.80	100
weighted avg	0.83	0.80	0.80	100

Predicted Labels:

- 0: Not Sarcasm
- 1: Sarcasm

Perform much better than just prompting

Example:

Parent comment:
The article itself is only worse than your summary of it. A grubby little commentary by someone who still believes Orwell got it right when he though his prophesy was of a *Socialist* society, and not a warning about Capitalism as it is seen today.

Comment:
we should definitely run away from 'tyranny' into the hands of capitalism

Output : Sarcasm
Actual : Sarcasm

Conclusion

- Binary classification is rather an easy task for model like LLM
- Sarcasm detection though has been proven to be a difficult task with only 84% human accuracy.
- Model is not able to perform great with just few shot prompting and has accuracy of just 45%. Model also hallucinates and generates garbage values.
- Finetuning improves the results quite a bit to an accuracy of 80% and prevents hallucination.
- Same approach of finetuning can be applied to any other domain problem

THANK YOU!

Reach out to me at : LinkedIn
My medium publication for NLP projects : Pal4AI