* Week 3: Reinforcment learning from Human Feedback.

* RHLF (Reinforcment Learing from Human Feedback)

- models behaving badly

- - Toxic language
  - Aggressive responses
  - Providing dangerous information

The important human values- Helpfulness, Honesty & Harmlessness are sometimes collectively called as HHH.

- Fine tuning with human feedback.

- RLHF uses Reinforcment Learning (RL) to finetune a model.

- maximize Helpfulness
- minimize harm
- Avoid dangerous topics

- Reinforcment Learning :

~~In this framework, the~~

Reinforcment Learning is a type of machine Learning, in which a agent learns to make decisions related to a specific goal by taking actions in an enviorment, with the objective of maximizing some notion ~~of a~~ cumulitive reward

In this framework the agent continually learns from its experinces by taking actions, observing the resulting changes in the enviorment, & recieving rewards or penalties, based on the outcomes of its actions.

ex. Training a model to play Tic-Tac-Toe

* RLHF : obtaining feedback from humans.

Steps to fine tune LLM using RLHF.

- Select a model to work with & use it to prepare dataset for human feedback.
- Use the LLM along with prompt dataset to generate number of diffrent responses for each prompt.
- The prompt dataset is comprised of multiple prompts, each of which gets processed by LLMs to produce set of completions.
- Next step is to collect human feedback on labelers generated by the LLM.

- collect Human Feedback

- Define your model alignment criteriation. (eg. HHH)

- For the prompt-response sets that you just generated, obtain human feedback through labler work-force

- once human labelers have completed their assesments of the prom completion sets you have all the data you need to train the reward model.

- Prepare labeled data for training.

- Convert rankings into pairwise training data for the reward model.

* RLHF: Reward model

- Train the model to predict preffered completion.

- Use the reward model as binary classifier to provide reward value for each prompt completion. pair.

* RLHF: Fine-tuning with reinforcment learning

- First you'll ~~her~~ pass a prompt from prompt ~~dataset~~.

- Instruct LLM will generate a completion.

- Next, you will send this prompt completion & original prompt to reward model as a prompt ~~con~~ completion pair.

- The reward model evaluates the pair & returns a reward value.

- A higher value represents more aligned response. (for ex. 0.23 vs -0.53)

- we will then pass this reward value to reinforcement algorithm to update weights of the LLM.

- we can call this intermediate version of model as RL updated LLM.

- This series of steps toghether forms single iteration of the RLHF process.

- These iterations continue for a given number of epochs.

- If the process is working well you'll see higher reward score after each iterations.

- A popular choice of model for RL is proximal Policy Optimization (PPO)

* Proximal policy optimization (optional)

- PPO optimizes a policy in this case a LLM to be more aligned with human preference.

- First we will intialize PPO with instruct LLM.

- At a high level each cycle of PPO goes over two phases.
  i) Create completions
  ii) model update.

- The Goal is to minimize the value loss. i.e. the difference between the acteal future total reward & its approximation through the value funch.

- In phase 2 you make small updates to the model & evaluate the impact of those updates on your alignment goal.

- PPO makes sure the model updates stays within a small region called trust region.

- Calculate policy loss formula in word doc.

- Q-learning is a alternative technique.

* RLHF: Reward Hacking

- Reward Hacking is an interesting problem that can emerge in RL, where the agent learns ~~to~~ to cheat the system by favouring actions that maximize the reward recieved even if those actions doesn't align well with the original objective.

- To avoid reward hacking we can use a reference model.

- The completion from RL-updated model is compared with Reference model

- By comparing two completions you can ~~called~~ calculate Kullback-Leibler diveregence or KL-divergence for short.

- KL divergence is a statistical measure of how diffrent two probablities are.

- This is a compute expensive process.

- once you have calculated KL divergence you will add acid term to reward calculation.

- This will penalize RL updated model if it shifts too far from reference LLM.

* Scaling Human Feedback.

- Although you can use reward model to eliminate the need for human evaluation, the human effort required to produce the trained reward model in the first place is huge.
- methods to scale human feedback are an active area of research
- one idea to overcome this is to scale through model self supervision
- constitutional AI is one approach of scale supervision

- Constitutional AI is a method for training models using set of rules & principles that governs the models behaviour.

- Constituonal AI can also be helpful to address some unintended consequences of RLHF.

* LLM powered Applications

- model optimizations for deployment

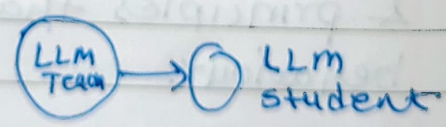
- LLMs present inference challanges in terms of computing & storage requirements as well as ensuring low latency for consuming application.

- one primary way to improve model performance is to reduce the size of LLMs.

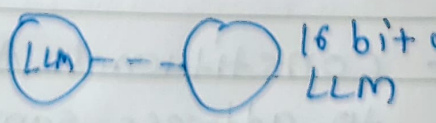
- This can allow for a quicker loading of the model which reduces inference latency.

LLM optimization techniques

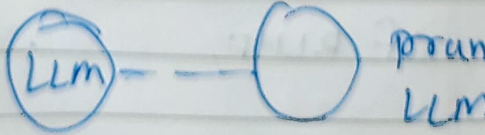
i) Distillation



ii) Quantization



iii) pruning



- Distillation.

- model distillation is a technique that focuses on having a larger teacher model train smaller student model.

- you start with fine tune LLM as teacher model & create smaller student model.

- you freeze the teacher model's weight & use it to generate completions for your training data.

- At the same time you generate completion using smaller student model.

- The knowledge distillation between teacher model & student model is achieved by minimizing a loss function called as distillation loss.

- Distillation applies tempreature parameter to the softmax function.

- with tempreature > 1, the probability distribution becomes broader & less strongly peaked.

- This softer distribution provides you with a set of tokens that are similar to ground truth boxes.

- Teacher models o/p → Soft labels
  student model's o/p → Soft prediction.

- In practice ~~student model~~ distillation is not as effective on generative decoder models. It's typically more effective on encoder only models.

- Quantization (post-training) or PTQ

- PTQ transforms model weights to a lower precision representation, such as 16-bit floating point or 8-bit Integer.

- Applied to model weight and/or activations

- In general, quantization approaches that involve model activations can have higher impact on model performance.

- Requires additional callibration to capture dynamic range.

- Pruning

- Remove model weights with values close or equal to zero.

- Some pruning methods require full model retraining while some uses PEFT) LoRA

- In practice, only small % in LLMs are zero weights.

* Using LLMs in Applications

- Models having difficulty

  - knowledge cutoff
  - Problems solving maths
  - Hallucinations

To solve this we need to connect LLMs to external Data sources or Applications

- Retrieval Augmented Generation (RAG)

- RAG is a great way to get rid off knowledge cut off.

- It gives LLM access to external data at inference.

- RAG isn't a specific set of technologies but rather a framework for providing LLMs access to data they did not see during training.

- At heart of this model's implementation is a model component called the Retriever.

- A query encoder takes user's input prompt & encodes it into a form that can be used to query the data source.

- The Retriever returns the best single or group of documents within the external data that are most relevant to input query.

- It then combines it with original query

- The new expanded prompt is then passed to LLM, which generates completion.

- Data preparations for RAG

- Data must fit inside context window
- Data must be in format that allows its relevance to be assessed at inference time : Embedding vectors.

- Interacting with external Application.

- In genaral connecting LLMs to external Application allows the model to interact with the broader world
- Extending their utility beyond language tasks.
- Trigger API calls.
- Perform calculations

classmate
Date
Page

- Helping LLMs reason & plan with chain of thought.

- one statergy that has demonstrated success is prompting the model to think more like human.

- Humans take step by step approach solving complex problems

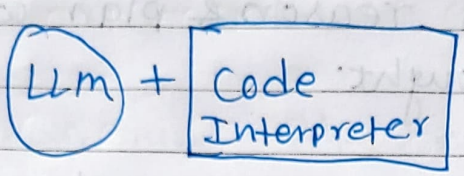
- Chain of thought prompting

- Program Aided Language models (PAL)

- LLMs can struggle with mathematics

- You can overcome this limitation by allowing your LLM to talk with external application which are good at this stuff like a python interpreter.

- One intresting framework for augmenting LLMs in this way is PAL.



- It makes use of chain of thought prompting.

- PAL Models

- ~~Each~~ prompt should contain one or more examples

- Each example should contain a question followed by reasoning steps in lines of Python code that solves the problem.

- Next, you will append the new question you would like to answer to prompt template.

- Next, you will pass this combined prompt to LLM, which generates completion in form of python scripts.

- Now, this script will be sent to python interpreter which will run the code & generate answer.

- you can automate this task using orchestration library.

• ReAct : combining Reasoning & Action

- Question → Problem that requires adv. reasoning
- Thought → a step that identifies how the model will tackle the prob. & identifies an action
- Action → an external task that model can carry out from allowed set of actions
- observation → the result of carrying out the action.

- It repeats the steps as many time as it wants to obtain the answer.

• Building up the ReAct Prompt.

Instructions + ReAct example + Question to be answered

Full prompt to be passed to LLM

- LangChain (Orchestration Library)
  - One solution being widely adopted is LangChain
  - The Langchain framework provides you with modular pieces that contain component neccesary to work with LLMs.

- Prompt Templates
- Memory → To store interacting with LLMs
- Tools → e.g. calls to external datasets & various APIs

- Building Generative Application.
  - Infrastructure Layer.

Training / Fine-tuning, serving, Application components

On premises or Cloud.

- LLM Models

  Foundation models as well as optimized models for specific task.

  Deployed on appropriate Infrastructure.

  Information sources (RAG)
  Database, Documents, Web

  Generated output & feedback.

- LLM Tools & Frameworks

  Langchain, model Hubs

- Application Interfaces.

  Websites, Mobile Apps, & API etc.

{ Refer to word Document for detailed E Diagram. }