Project: Improving OCR capabilities in Large Vision Language Models (LVLMs)

Target Deadline: Model and Paper release by Feb end, target conference: COLM 2024

Project motivation: OCR (Optical Character Recognition) is a fundamental vision capability that all Vision Language Models are not satisfactory. The open source LVLMs (including Robin) may fail for even basic images. Even the closed source models (OpenAI's GPT4 and Google's Gemini) fail at non-salient text or for non-printed and non-digital documents.

Goal is to get experience on the lifecycle of building a multi-modal LLM - working well on a narrow domain:

- Preparing Data, Training and Evaluating.
- Generating Synthetic Conversational Data.
- Training the model, for the time being treating it as a black box, with only an API access

Desired longer term skills to be developed:

- being comfortable with an open ended problem
- breaking down problems into smaller components
- having a structured approach to solving problems
- ability to learn using internet and AI tools
- observation skills - especially for data
- autonomy

Current task:

- Evaluate Gemini Pro (free tier) on DocVQA, PDFVQA and Infographic VQA.

Free AI Tools to use:

- github copilot (copilot is free for students)
- Phind.com
- Bard.google.com
- perplexity.ai
- replit's ghostwriter
- chat.openai.com
- Bing copilot:
- LocalLLaMas - LMStudio, ollama, oobabooga only if you have enough RAM (atleast 16GB) and high-end laptop (or GPU - choose exllama backend).
    - Check the top models on LLM leaderboard that are also served by LLM API providers like together.ai (if they host them by default then these are among most popular)
    - Models by nous research are pretty good.
    - Get thebloke's quantized models.

Free AI credits:

- GCP - default $300
- Together - $25
- Azure ?
- Google gemini pro has a free usage tier.

Components of Training a Neural Network:

1. What do you train? -> Neural Network
2. How do you train? -> Optimizer + Back Propagation
3. What do you train it on? -> Data + Training Loss/Reward/Objective function
4. Where do you train it on? -> GPU (also the issues with scaling and all)
5. How well does it perform? -> Evaluating/Benchmarking performance.

1. and 2. remains mostly the same (from GPT1 to GPT3.5 at least). Only the number of parameters for NN increases.
4. - Once this is solved, it doesn't need to be iterated upon a lot.
3 and 5 - Most of the work is always done here. Data + Training Objective/Loss/Reward is what took OpenAI 3 years to get from GPT3 to GPT3.5. Evaluations and benchmarking tell you what to improve the model on.

(Try drawing analogy to a linear regression)
A Neural Network with unfrozen (i.e. trainable) parameters will represent a set/space of functions.
And with frozen parameters it will represent one function from the entire space of function that optimizes the objective over the data.

Things to keep in mind about LLMs/LVLMs:

- LLMs/LVLMs are machines - they are pieces of code/software
- An LLM performs well on a task because it has been trained to.
  - Just like a human cannot recognize/read Japanese script unless taught, neither can your LVLM.
  - If you do not have pictures of Japanese script in your data, model will never learn anything about it.
- LVLMs are dumber than humans
  - Simply getting LVLMs to blurt/read all text (i.e. `write out every word written in the image`) that is present in an image won't be sufficient for them to auto-generalize (in most cases) to answer all queries about the image-text (queries such as `write out every second word in the image`, for example)
- Since they are software, they follow Garbage In-Garbage Out
  - See: ChatGPT Fine-tuning Fiasco
    https://www.linkedin.com/posts/drjimfan_finetuning-cautionary-tale-what-you-get-activity-7128036026454855680-yLzz/

    - Bonus 1: Why is this comment not a good explanation of the phenomenon https://www.linkedin.com/feed/update/urn:li:activity:7128036026454855680?commentUrn=urn%3Ali%3Acomment%3A%28activity%3A7128036026454855680%2C7128039977862500352%29&dashCommentUrn=urn%3Ali%3Afsd_comment%3A%287128039977862500352%2Curn%3Ali%3Aactivity%3A7128036026454855680%29
    - Bonus 2: How would you go about training on slack data (without RAG at inference time), so that model knows about your slack conversation knowledge? (Hint: First it might be worth defining what do you want the model to behave like)
- It is unfair to expect LLMs to decode in one time-step what usually takes us several (and it leads to less interpretability and more hallucinations):
    - Can you immediately tell me what is `sin(1263*3474)`?--- Neither can an LLM in a single step, let it use (and train it to use) Chain of Thought (to write down intermediate steps) or give it a python interpreter (where it can evaluate this.
    - What other such tasks can you think of? … Here's one: Counting the number of objects in an image or number of "o"s in this sentence.
    - There is a small catch here - with large enough models (i.e. complex enough set of mathematical functions), one could train an LM to learn to compute `f(x,y) = sin(x,y)` by sampling a very large number of x and y
- LLMs do not and will not generalize out of distribution (OOD)
    - Common misconception: Performing well on unseen examples is not the same as OOD Generalization.
        - For example, an AlexNet classifier performs well on the test set, and it has not been trained on all space of pixels - this does not mean it is generalizing OOD.
    - Current best way to deal with OOD is to make it In-Distribution somehow.
        - Remember that GPT has been trained on nearly all internet data and is constantly being trained on user generated data - this leaves very little space of input text that is OOD for it.
    - If anyone claims that ChatGPT generalizes out of distribution --- ask them if they know what input-output pairs has ChatGPT been trained on.
- The set of queries most systems gets is a long tailed distribution.
    - Do not be tempted to cover all sets of possible queries (i.e. the long tail).
        - For example `generate poem based on given image where every second word is apple and every fifth word is orange`, is typically not desired for even creative writing purpose (unless you really expect people to use model for use cases like these -> if a small set of people use it, then RLHF/DPO can be used post deployment for long-tail)
    - Focus on the ones that will be most commonly asked by people (including you), for your real world use cases. Most people ask the same or similar queries.
        - A testament to long tailed distribution in real world: Sundar Pichai testified 6 years back to US congress that google search has seen 85% of the

        queries it gets it has seen before -- imagine what percent of the remaining 15% would be typos, grammatical errors, paraphrasing, and other in-distribution instances.

    - The long tail distribution is (as of now) best handled later by RLHF/DPO upon model deployment --- This is not possible for autonomous driving/agents but where an adult human checks the output (like in our case), this is completely fine.

Other things worth knowing:

- Sampling Bias: For example survivorship bias in case of Aeroplane example
- Recall vs Precision in binary classifiers
- Universal approximation theorem for Neural Network
- Understanding Deep Learning requires rethinking generalization - No need to go into The VC dimension and Radamacher complexity. Just understand why the results were surprising.
- Train-Val-Test set
- You should know how LLMs are being trained.
- GPT-3 vs InstructGPT vs ChatGPT