< Improving Accuracy of LLM Application >

partnership with Lamni & Meta

LLM : trained on "public dataset" and task

↳ they not perform well on application requiring "proprietary data"

↓

But : Llama : open-source ! ⟶ Any one can fine-tune the model.

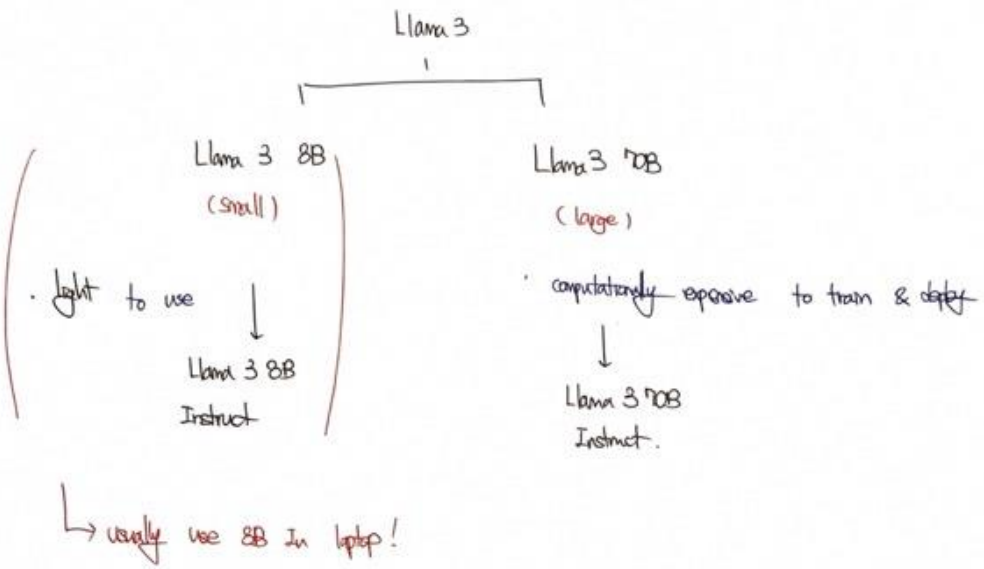
When Fine tuning doesn't work ?

↳ think as : not enough data ⟶ no ! , the way you tune should be revised .

Fine-tuning : slow & costly ⟶ But (LoRA , parameter efficient fine-tuning .
or
Lamni memory tuning
↓↓
cost ↓ , fast !)

In this course , learn the way

Improve Accuracy of LLM by SQL



↳ usually use 8B in laptop!

xxx hallucinations

- Incorrect or fabricated information generated by the model
- LLM thinks : something slightly right = right.
- Detrimental with facts & when you need precision

xxx Common approaches help but not enough

- Prompt engineering → 26%
- Self - reflection → 26 - 40%
- Retrieval Augmented Generation → 50%
  (RAG)
- Instruction Fine-tuning → 40 ~ 60%

How Can fine-tuning reduce hallucination?

- Embed facts into the model

cf) Hw, instruction Fine-tuning isn't the tool to Remove the hallucination.

- Memory Tuning (by Lamini) allows the models to recall a lot of facts precisely

Important! Without Compromising generalization & instruction-following

- Memory Tuning → 95%

↳ Reason: Sample from a distribution that looks more like this (A)

EX) A SQL Agent.

LLM: can be SQL Agent

↳ but makes hallucinations

↳ fine-tuning → Lift the Accuracy!

Is SQL Agent useful?

1. optimal UX

faster answer for business users.

2. Operational Efficiency

Less data team time spent answering simple questions.

3. Reliability

More reliable in breadth than business users writing their own queries.

Hallucination In SQL Agent.

1. Invalid SQL

   Missed column names / IDs, formats, or functions.

2. Malformed SQL

   Valid, but semantically wrong.

How to quickly improve accuracy

↳ New way: Lanni Memory Tuning: embeds facts about the database into Llama 3 very efficient.

↓

SQL Agent's workflow with Llama 3

User Question → SQL LLM → Run query → Response user sees

Diagnose hallucination

SQL query [invalid / malformed] → diagnose hallucination: Iterative.

- As you improve the model, you'll continue to dig for hallucination.
- this is how you point the LLM to what it can improve on.

Goal of Evaluation

- Is it improving ?
- where is it hallucination? → where can I fix more? (improve)

Good Evaluation

1. Quantitative
2. Point out what you can improve
3. Scalable & automated

Initial evaluation dataset.

- start small : 20 ~ 100 data
- Quality > Quantity. Iterative expansion
- Focus on areas of improvement

Practical tips for your evaluation dataset.

- Easiest examples that still fails.
- Use an "adversarial playground"
  ⇓
  let your LLM get break down.
  And check the boundary.
- Set a next accuracy target for your LLM.
  small target → big target.

Scary SQL LLM Specifically

- generated SQL can be a correct semantic match but not exact match
- Execute generated SQL against DB and compare SQL exact match

Fine-Tuning

Why Fine-Tuning?

- Fit more data than what fits into the prompt → storing info in weights
- Learn from the data, rather than just get access to it
- Deeper Control of the LLM to achieve what you want
- No Accuracy setting

Types of Fine-Tuning

Instruction Fine-tuning

↓

Get a pre-trained LLM to follow Instruction

Memory Tuning

↓

Get an LLM to not hallucinate

Why Pre-training is not enough?

Pre-training

- LLM reads the internet, one token at a time
- Reduces average error over example (generalization error)
- It creates powerful foundation models, learning a lot.

⇓

But, there's still a gap in usefulness

- LLM cannot follow instructions
  → It doesn't have a chat in "chatGPT"

- LLM hallucinates on facts
  → Pretty good at everything, but perfect at nothing.
  → nearly right answers are still wrong

Memory - Tuning : Teach perfect recall on facts

- Reduce error to zero on facts
- Near - perfect on facts , pretty good at everything else.
- Before : a probability distribution over the answer
- After : only commit to the right answer

Myth1 : Prompting & RAG will solve all my problems ?

Following Instructions : Use prompting to get the model to do a little bit better

Fact Recall : Use RAG to get the model to do a little bit better → shift probabilities conditioned on retrieved prompt, but still sampling from a distribution of similar but wrong fact.

Myth2 : Fine - Tuning is Still too expensive ?

Reality : cheaper than running huge prompts in RAG

PEFT (Parameter - efficient fine tuning) : Reduces cost dramatically !

MoME (Mixture of Memory Experts) : turns any LLM into a million - way mixture - of - expert adapters , reducing time by 240x

< Large Multimodal Model Prompting With Gemini >

Gemini : multimodal model from google Deepmind.

< Model Parameter > – In Gemini

Topk : The Top k possible next words

TopP : The Top P possible next words with a cumulative probability ≥P

Temperature : chose next word based on rt

└→ high Temperature → Randomness : ↑

low " → " : ↓

< Image Requirements, Image Caring >

Image to tokens : Use up to 3000 images

Image type : PNG or JPEG

Image Pixel : unlimited, but large images are scaled down and padded

Image to token : Each Image accounts for 258 tkens