[Day 1] - Intro to Unit 1

"Foundational LLM & text generation" Reading/Podcast

- Foundation = transformer arch.
  - → encoder & decoder
  - encoder ↓ takes input inform. / decoder ↓ gen. output piece by piece.
- Transformer layer
  - input text needs to be prepared into tokens.
  - tokens → dense vector (embedding) which captures meaning → pos. encoding (makes structure
- Self-attention: lets model figure out relationships between words & phrases.
  1) creating queries, keys & values
  2) calc scores
  3) normalization
  4) weighted values.
- Multi-head attention: mult sets of [which run in parallel]
- Layer normalization & residual connection
  - feedforward layer - (learns func. of input)
- Decoder only → masked self attention + gen text
- Mixture of experts MoE: experts, gating network, combination mechanism
- Keeps activity level @ each level, training faster
  - Short cuts, way for network to remember what it learned earlier.
- directly gen output token by token (simpler @ gen text)

Inference techs:
- Output approx: change output slightly (↑ efficiency) - quantization - distillation
- Output pres.: keep output same, optimize computation.
- general parallelizing
- prefix caching
- spec. decoding
- *Flash attention

Evaluating model performance
- Data spec.: designed for task
- consider whole system
- what does good mean?

Methods:
- human evaluation ($$$)
- AI (evaluator) *need to be calibrated.

speed up response (inference)
- quality vs. speed.

Transformer
- Sequence to sequence model capable of converting seq. from one domain into seq. in another domain.
  - og only contained encoder & decoder

Training
- Pretraining → fine tuning (smaller, focus on task) → reinforcement learning (human feedback) →

Parameter Eff.
- Parameter efficient fine tuning: train small part of model, makes fine tuning less expensive. → adapter based fine tuning.

Prompt Engineer.
- Designing input given to model to get desired output
  - techs: zero-shot, few shot, chain of thoughts

Sampling
- Sampling tech: greedy search, random samp., temp. top k, top p, best of end.

[Day 1] - Unit 1 comp.

"Prompt Engineering" Podcast / Redding

- config LLM output tail output length (affects costs / proc. times)
  - low token limit.
  - engineer prompts to be targeted to get concise response (ex: react technique)

Sampling

- How LLM chooses next word based on prob.
- **Temp**: "dial" of randomness
  - **lower**: pushes model towards most likely word
    * predictable, deterministic output. *precision, ...
  - **higher**: introduces randomness, diverse results.
    * exploring new ideas
- **Top k / Top p** (nucleus sampling): limit the next word to most prob. candidate
  - ↑ k = variety in output - ↓ k = focus - k = bad grammar
  - **Top p** (prob.): selects smallest set of words whose value adds up to p. (lower: restrictive; higher: all pos)
- Filter w/ Top k & p, temp make final choice

* Ideal: temp = 2, top p = 0.95, k = 30

Rep. Loop Bug

- Model gets stuck repeating phrases, happens @ low / high temps
  * too predictable or too random

Prompt Eng. Techs

- **General (zero shot)**: giving model task descript input w/o exs
- **Doc. prompts**: keeping track of what worked / didn't
- **One / few shot**: provide examples within prompt to guide model
  - quality of examples is crucial. (edge cases)
- **System contextual & role prompting**: giving model all details.
  - context & purpose - giving LLM persona (clear perspect)
- **Step back prompting**: starts w/ asking broader q, before specific
  * mitigates biases
- **Chain of thought**: boosts reasoning capabilities by prompting model to generate reasoning steps. (uses more tokens)
- **Self-consistency**: get model to generate mon reasoning paths for same prompt, choose most consistent answer. (expensive)
- **Tree of Thoughts**: LLM explores mult. reasoning paths
- **React**: reason & act, combining LLM reasoning cap. w/ ability to use external tools (search engine)

- **Automatic prompt Eng**: ai, automation of effective prompts.
  - effective for writing code.
- **Multi-modal prompting**: using images/audio
- **Best Practices**: provide examples, be specific, use simplicity, control max token length. (constraints) + documentation + experiment!

Codelabs

[prompting fund.]

Output length: affects cost & performance.
- more tokens = ↑ computation (↑ energy, latency, cost)

Temperature: controls degree of randomness in token selection. ↑ temp = ↑ # of candidate tokens from which next output is selected. = diverse results.
- temp @ 0 = greedy decoding (no diversity)

Top p: defined prob. threshold that once cum. exceeded tokens stop being selected as candidates.
0 → greedy decoding ; 1 → every token selected in ⓜ words.

Prompting: Zero shot (direct), enum (label ASPA), chain of thought (intermediate reasoning steps), reason & Act, thinking mode (generates thinking process)

[Evaluation & structured output]

Define an evaluator: "instruction following, groundness, fluency, verbosity or quality

- Use: quickly iterate through a prompt w/ small set of test documents, compare diff models, unify system does not regress
- Pointwise eval: eval a single input/output pair against criteria
  - used for eval. singular outputs in absolute sense (good/bad)
- Pairwise eval: comparing 2 outputs against each other
- LLM limits: LLMs have problems on certain tasks, an evaluator will not be able to accurately eval. this task.
  - solutions: connecting tools, understand limitations, include human evals to calibrate & determine baseline, not relying on "internal knowledge" (provide w/ input / tool)
- Improve confidence w/ mult. evals (divise), sum prompts.

Challenges

- GPU mem needed for inference on 3B param. model: 12GB
- Reinforcement learning from human FB: reward model to incent for human pref. response
- Gem. charac. - support context w/ of up to 2M tokens
- Gem config that controls randomness: temp.

Livestream

- Models w/ no try to pull in more inputs from usr
- Unix ai
- AI system design: prompt caching, on device vs. pinging a server.

CHIIKAWA

- prompt eng: input, temp.,
  ← evaluation ← crucial