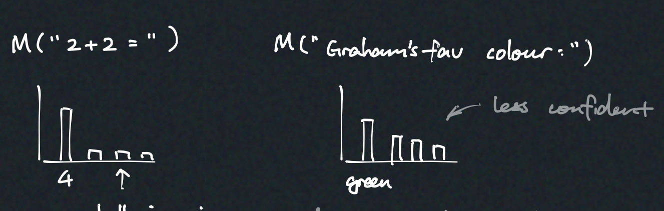
Lec 6 Generation Algorithms

Given model M, pretrained, lots of params
└ defines a conditional prob distribution

M("2+2=") M("Graham's fav colour:")



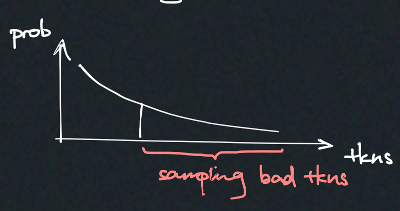
hallucination ← shown to always happen

Problem: how to get good output given hallucinations?

Sampling

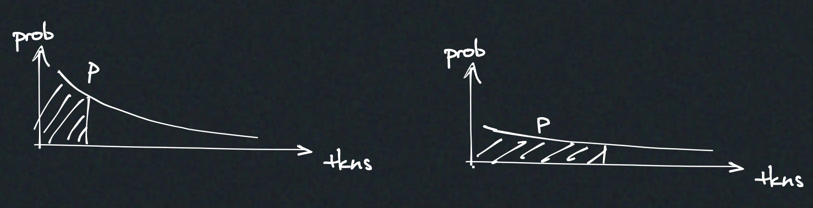
▹ Ancestral sampling $y_i \sim P(y_j \mid X, y_1, \dots, y_{j-1})$
└ sample according to dist at time step

Problem: long tail with 32,000 tokens



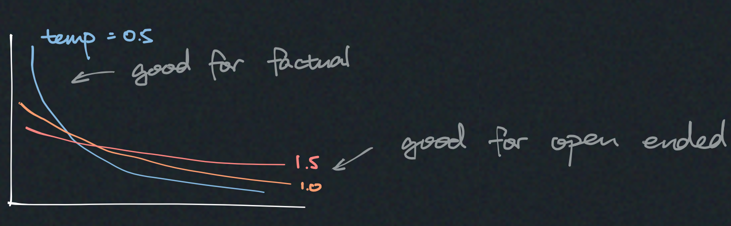
▹ Top-k sampling cut off tail and sample from the most probable

▹ Top-p (nucleus) sampling only take top p prob



▹ ε-sampling only sample tkns with > ε prob

▹ Temperature: morphing dist
before softmax, divide by temp



▹ Contrastive decoding (CD) — use expert model to "fix" dist from small, amateur model

then choose output the expert find more likely than the amateur
i.e. expert log prob − amateur log prob

idea: if this is large, probably because expert learnt sth the amateur didn't
also if both model have degenerate case, CD cancels them out

Mode decoding method

When we want to search for the most likely seq

▹ Greedy decoding: $y_j = \operatorname{argmax} P(y_j \mid X, y_1, \dots, y_{j-1})$

Problem: local optimal not always global opt.

▹ Beam search

Better approximation than greedy

Problems - Less diversity

▹ Diverse beam search

→ score candidates differently s.t. we get more diversity
eg. if 2nd rank very similar to 1st rank, lower its score and encourage sth different
→ Can define similarity func

▹ Stochastic beam search

→ Keep scoring the same but sample from them
without replacement

Minimum Bayes Risk (MBR)

Do we really want the highest thing?

The cat sat down	0.3	△
-- ran away	0.001	
-- grew wings	0.25	△ ←
-- sprouted off	0.2	△

Really want △ over △?
△ similar & add up more than △

Maybe ← has lower risk in terms of semantic

Estimate prob: keep sampling, monte carlo

Estimate risk: eval. agreement btwn candidates

MBR: $\hat{y} = \underset{y' \in \mathcal{Y}_h}{\operatorname{argmax}} \sum_{y \in \mathcal{Y}_e} G(y, y')$ low risk

high prob
cmp to pseudo-references

Sim metric G ex. ROUGE, BEER, ...

Problems: sampling many things expensive

Beam search known to sometimes degrade downstream metric perf.
Because true mode sometimes bad, ex. "I don't know" may have very high prob
Also, training may not lead to mode being good

▹ Output ensembling

→ multiple models, rather than outputs MBR from same model, we look at sim btwn models

▹ Self-consistency

1. Prompt for chain of thought
2. Sample many outs
3. Extract answers from each
4. Return most frequent

MBR-ish

Constrained Generation

E.g. M("Suggest activity") but don't want climbing
... doesn't quite work on LLMs.

▹ Decode-side solution
└ set prob of "climbing" to 0
... but what about "bouldering" or other semantically simlar results

▹ Sample many, map check if about climbing, then filter

... expensive
also hard if model really likes climbing

▹ FUDGE — Future Discriminator ← train additional classification model
└ at decode step, modify dist by estimation on whether the seq will end up being formal vs informal, e.g. estimator can be trained on labelled data

▹ Reinf. Learning from Human feedback RLHF

Aligned LM combines original LM & reward model (use some data source)

▹ Reward-augmented decoding — no reinforcement learning involved
└ each decode step, predict future reward & adjust tkn probs, like FUDGE

→ Maybe just ask the model itself whether it violates the constraints

Human-in-the-loop Decoding

Decoding with some human intervention in between
- World Craft — story gen with human interaction
- Human edits to {apply own style / fine grain replacement / specific edit request} and model continues

- Regeneration button

- Multiple generations for human to choose

Tree of thought paper — use model to decide which option to keep generating on

Practical Considerations

Decoding takes most time generally

▹ Speculative decoding
└ use small model to speculate next tkns
larger model rejects or accept
if mostly accept, small model jumps forward quickly while large model verifies blocks by block

▹ Fast inference / constraint inference libs

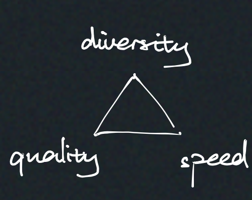
vLLM disco Outlines ...

Summary

- At decode step, adjust prob dist
- In decode process, choose btwn branches

Useful for
- Match constraints
- Use data source not used in training
- Compensate for worse model eg. GPT2

Tradeoffs



Decoding behaviour matters!

Decode method usually more predictable than fine tuning