Transformer Pretrained & Large Language Models

Ondřej Dušek

JSALT Workshop

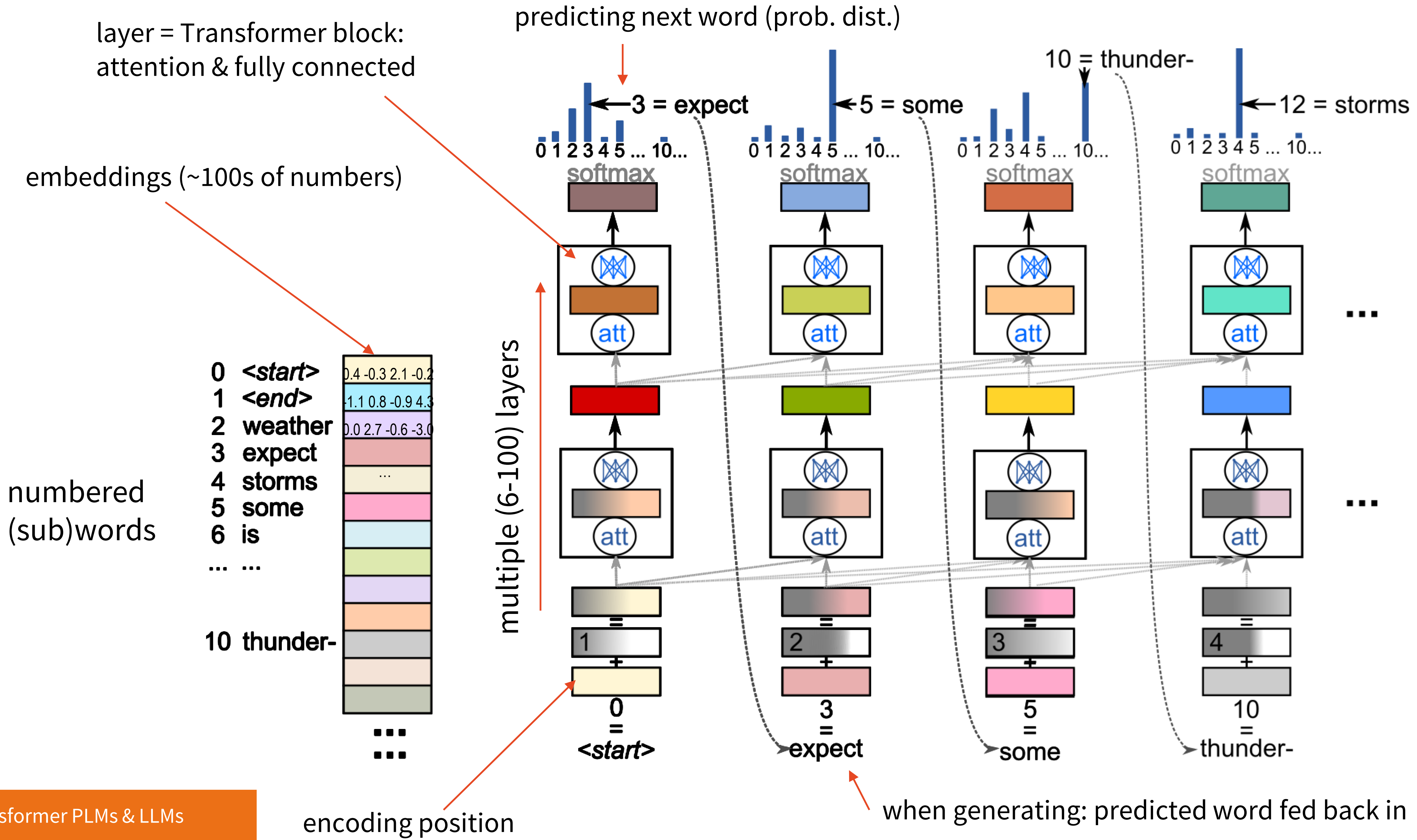
13.6.2025

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Neural language models

- **Transformer** neural architecture
  - (sub)word representation: **embedding** = vector of numbers
  - **blocks**: **attention** (combining context) + **fully-connected** (abstracting)
  - **predicting next (sub)word** = classification: choosing 1 out of ca. 50k (low level!)
  - trained from data: initialize randomly & iteratively improve
- Shapes
  - **encoder**: build representation of inputs
    - older models (BERT), good for classification
  - **decoder**: left-to-right, input stuff by prompting (prefixing)
    - most current LLMs
  - **encoder-decoder**: encode, attend, decode
    (original, from MT, what OB showed most of the time)

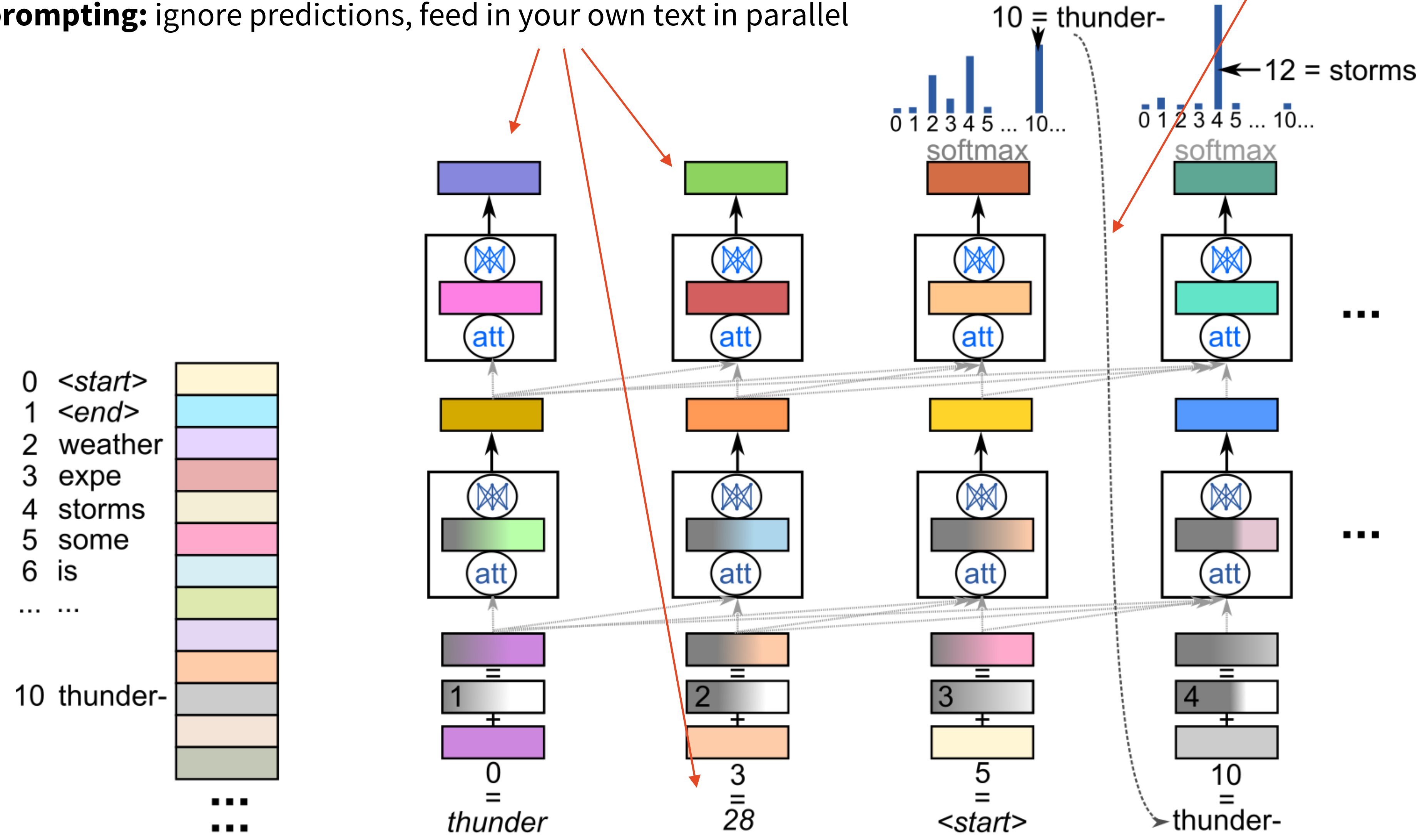
Transformer neural language model



Inference

continue
auto-regressively:
feed generated back in

prompting: ignore predictions, feed in your own text in parallel



Decoding Algorithms

- for each time step *t*, the decoder outputs a probability distribution: $P(y_t | y_{1:t-1}, \mathbf{X})$
- how to use it?
- **exact inference:** find a sequence maximizing $P(y_{1:T} | \mathbf{X})$
  - not possible in practice (why? and is it our goal?)
- **approximation algorithms**
  - greedy search
  - beam search
- **stochastic algorithms**
  - random sampling
  - top-k sampling
  - nucleus sampling (=top-p sampling)

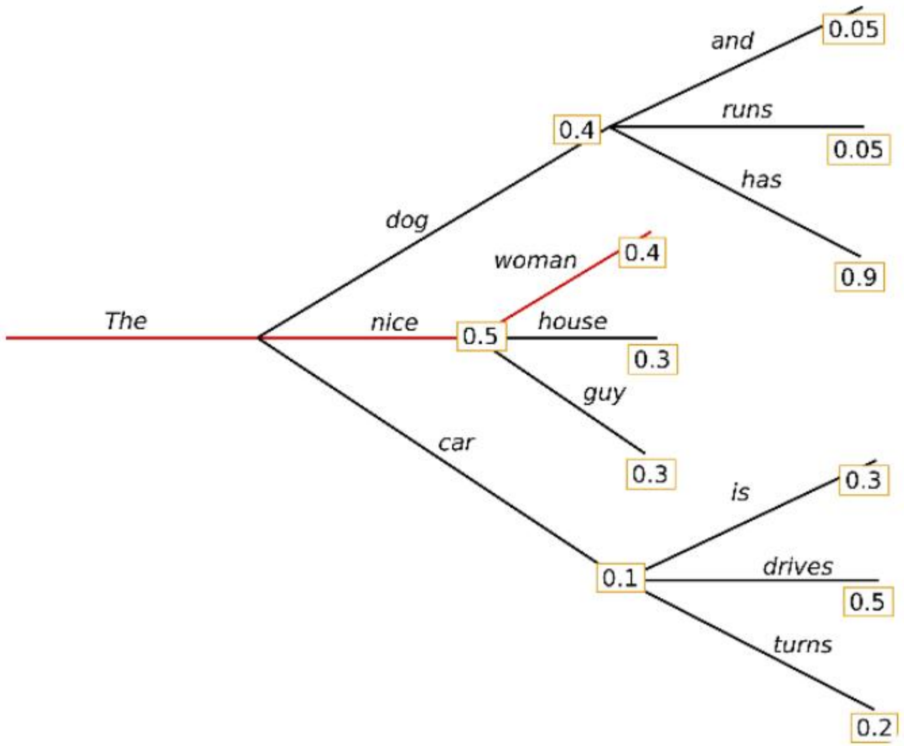
(+ repetition penalty → decreasing probabilities of generated tokens)

Decoding Algorithms

https://huggingface.co/blog/how-to-generate
https://towardsdatascience.com/decoding-strategies-that-you-need-to-know-for-response-generation-ba95ee0faadc

Greedy search: always take the argmax

- does not necessarily produce the most probable sequence (why?)
- often produces dull responses



Example:

Context:	Try this cake. I baked it myself.
Optimal Response :	This cake tastes great.
Greedy search:	This is okay.

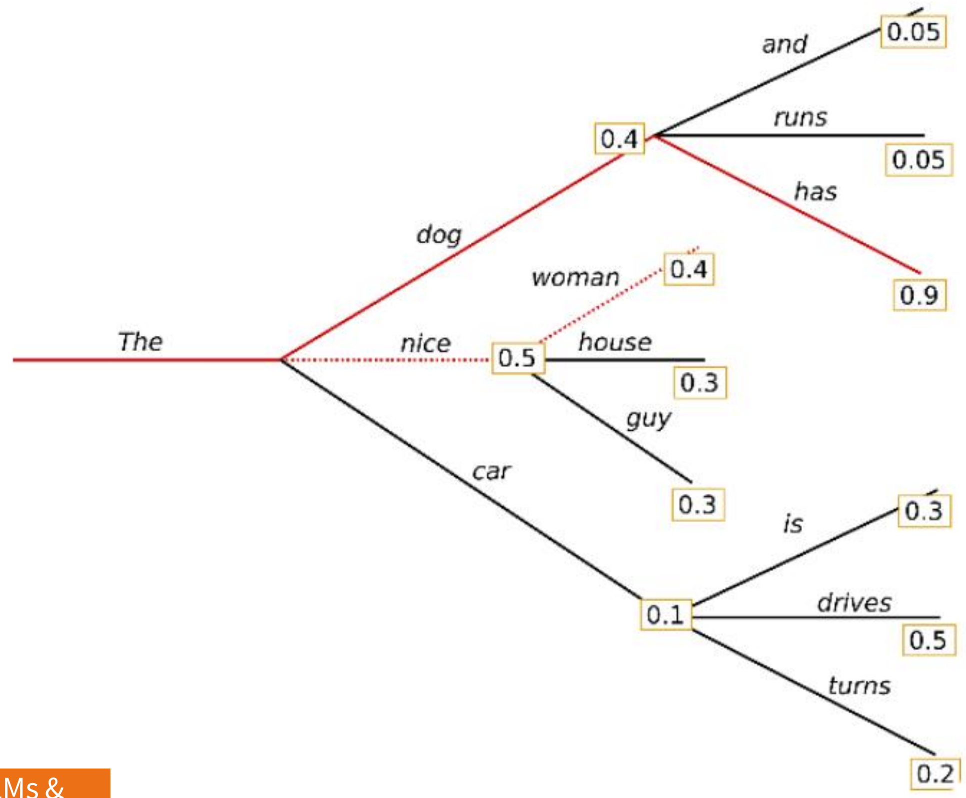
many examples start with "This is", no possibility to backtrack

Decoding Algorithms

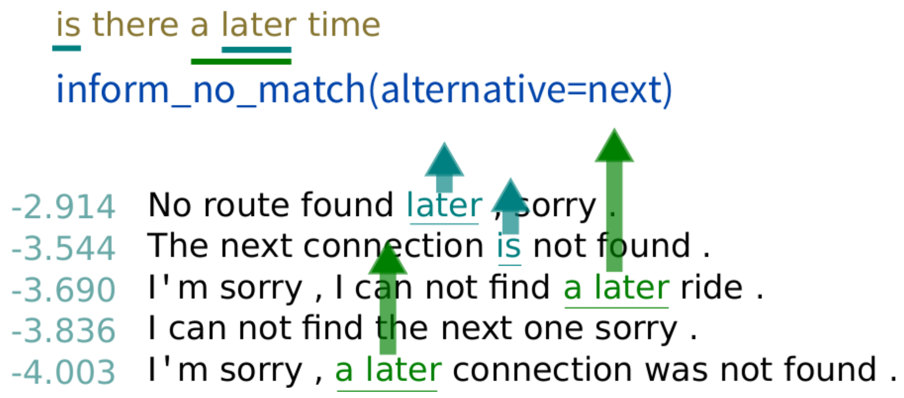
https://huggingface.co/blog/how-to-generate

Beam search: try *k* continuations of *k* hypotheses, keep *k* best

- better approximation of the most probable sequence, bounded memory & time
- allows re-ranking generated outputs
- k=1 → greedy search



Reranking:



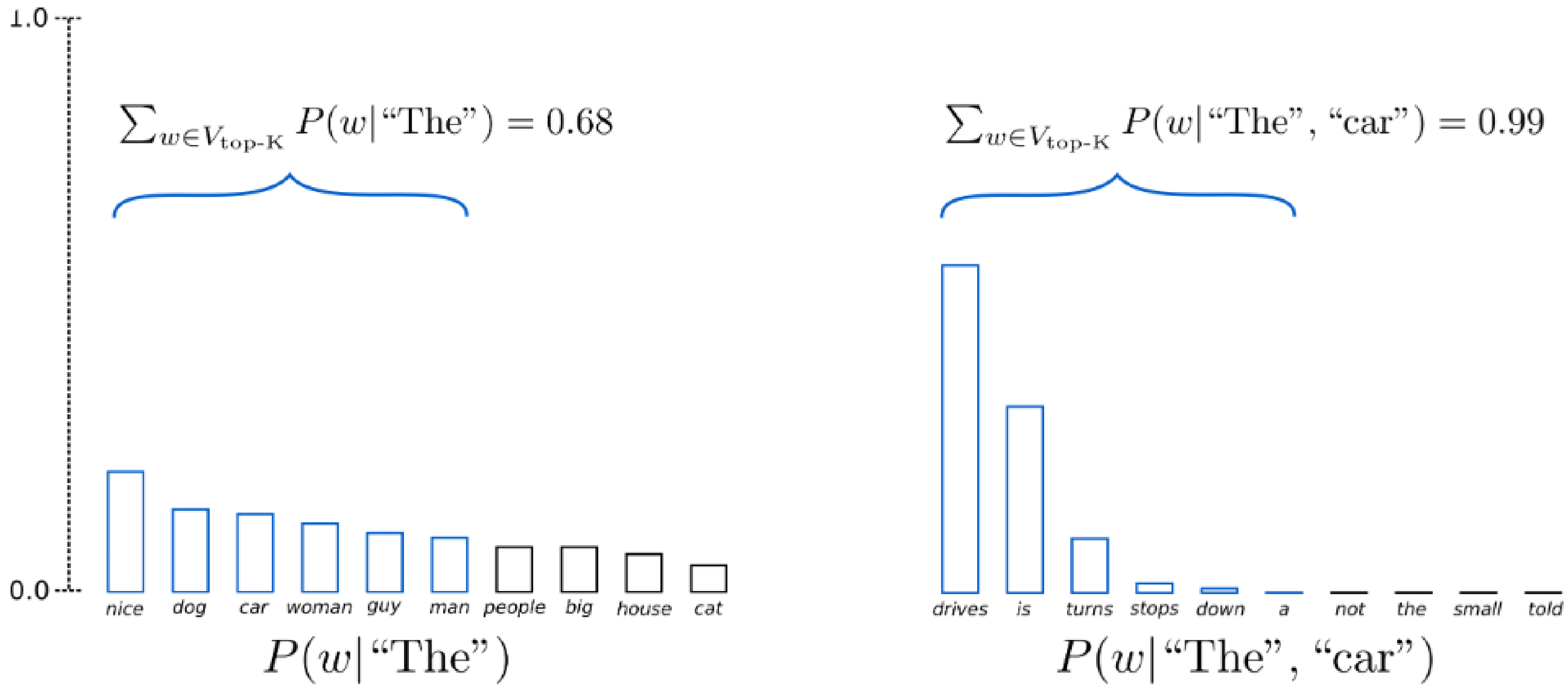
(Ondřej's PhD thesis, Fig. 7.7)
http://ufal.mff.cuni.cz/~odusek/2017/docs/thesis.print.pd
f

Decoding Algorithms

https://huggingface.co/blog/how-to-generate

Top-k sampling: choose top k options (~5-500), sample from them

- avoids the long tail of the distribution
- more diverse outputs

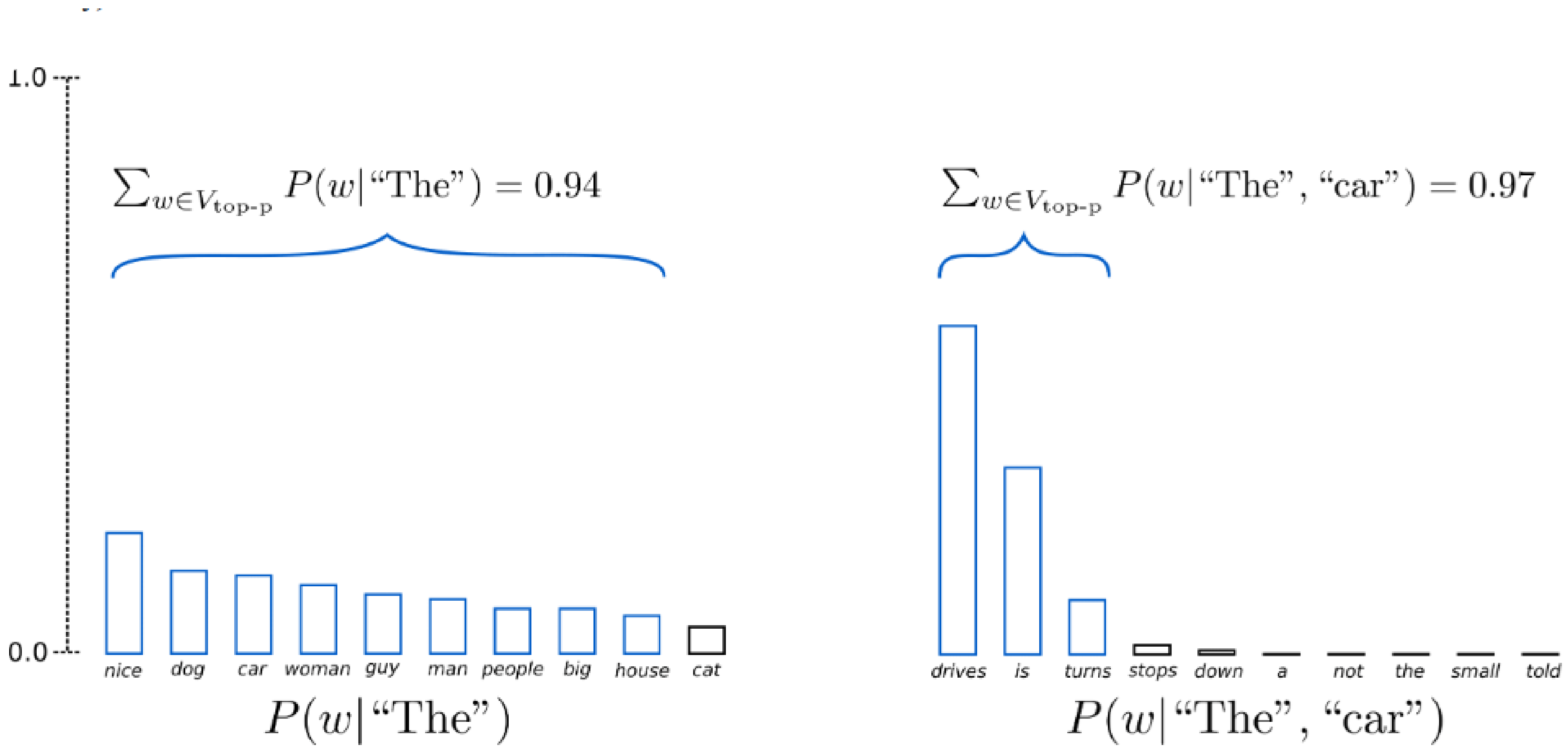


Decoding Algorithms

https://huggingface.co/blog/how-to-generate

Top-p (nucleus) sampling: choose top options that cover >= *p* probability mass (~0.9)

- can be viewed as "*k*" from top-k adapted according to the distribution shape

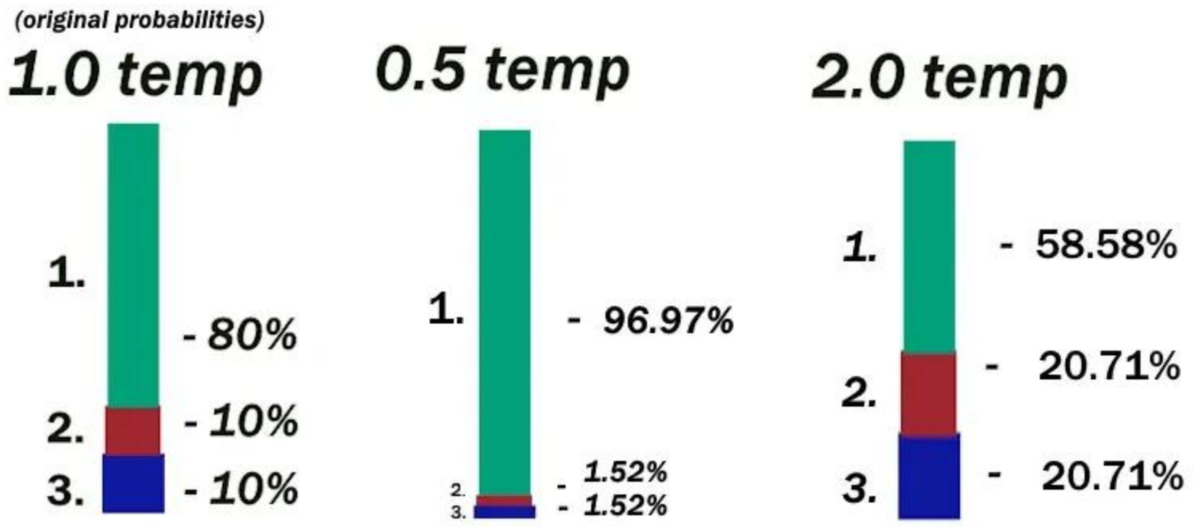


Temperature

https://www.reddit.com/r/LocalLLaMA/comments/17vonjo/

The shape of the distribution can be adjusted using the **temperature *T***:

$$\mathrm{softmax}(y_i) = \frac{e^{y_i/T}}{\sum_{y_j \in \mathcal{V}_{\text{top-k}}} e^{y_j/T}}$$



Is greediness all you need?

https://www.reddit.com/r/MachineLearning/comments/1e42das/

r/MachineLearning • 8 mo. ago
zyl1024

[D] What happened to "creative" decoding strategy?

Discussion

For GPT-2 and most models at that time, the naive greedy decoding is extremely prone to generating repetitive and nonsensical outputs very fast, and many techniques, such as top-p sampling, nucleus sampling, repetition penalty, n-gram penalty, etc. are needed. (e.g. https://arxiv.org/pdf/1904.09751)

For recent LLMs, I haven't been using any of these tricks, and instead, any temperature between 0 and 1 seems to work just fine. The only repetitive generation that I've observed seem to be in math reasoning, when the model wants to do some exhaustive search that didn't succeed.

So are all these custom decoding strategies a thing of the past, and we don't need to worry about degenerate content generation anymore?

23 12 Share

Training a neural language model the basic way

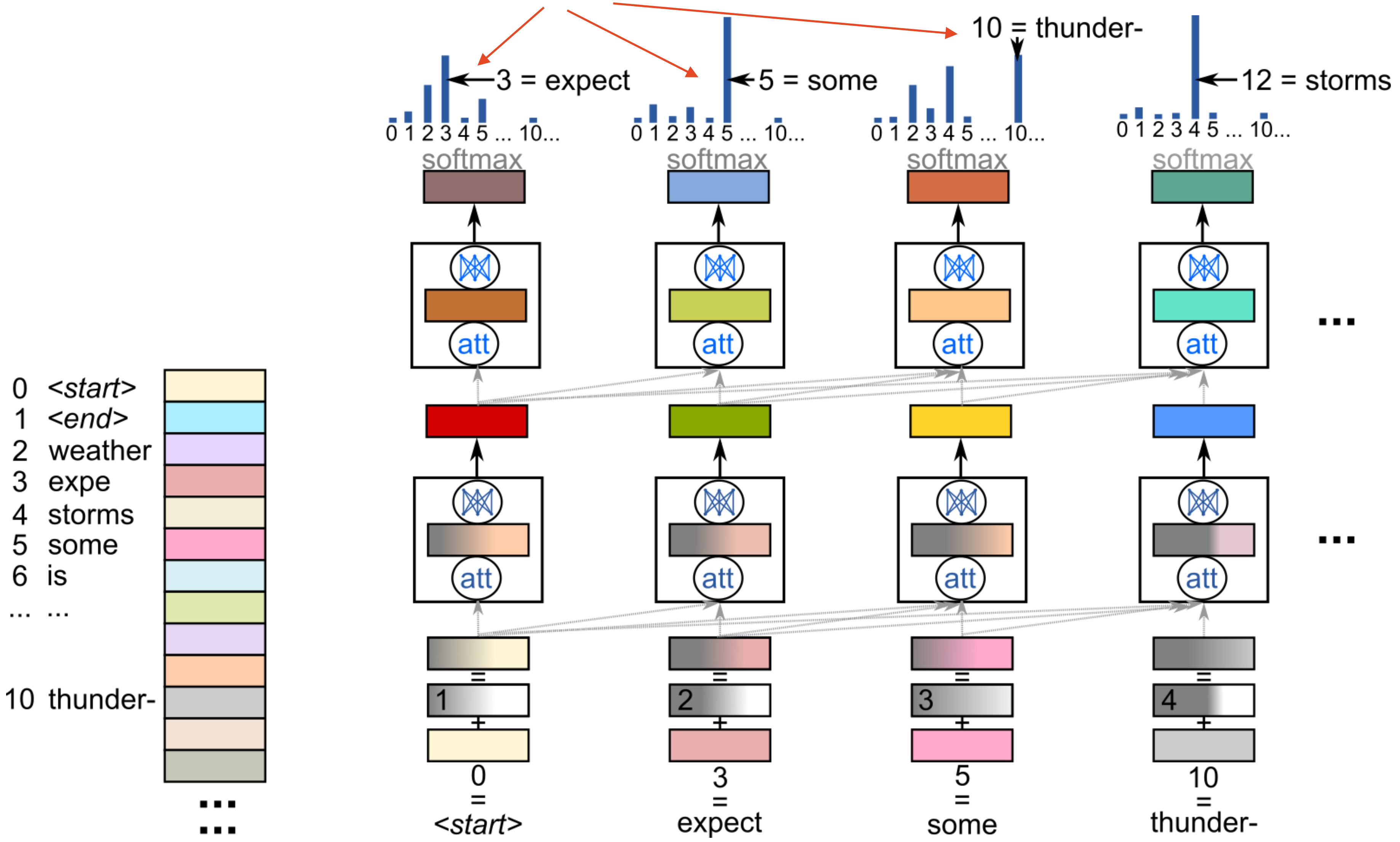
- Reproduce sentences from data
  - **replicate exact word at each position**
  - always only **one next word**, not the whole text in one
- Fully trained from data
  - initialize model with random parameters
  - input example: didn't hit the right word → update parameters

reference: *Blue Spice is expensive*

- Very **low level**, no concept of sentence / text / aim

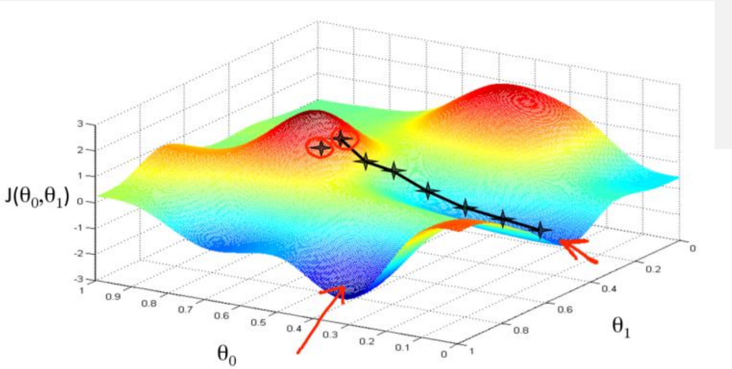
Training

in parallel: feed in training data & try to **predict 1 next token at each position,** incur loss

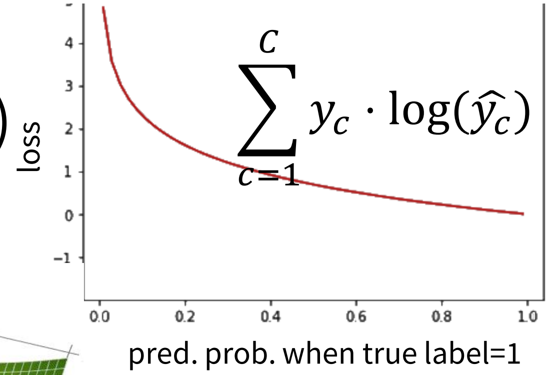


Training

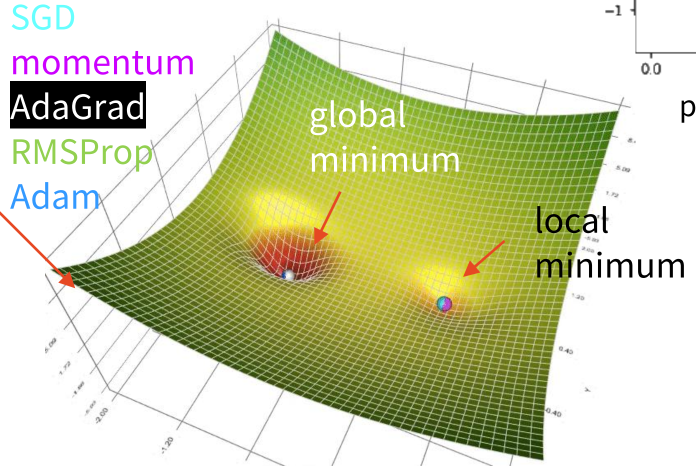
https://hackernoon.com/gradient-descent-aynk-7cbe95a778da
https://ruder.io/optimizing-gradient-descent/



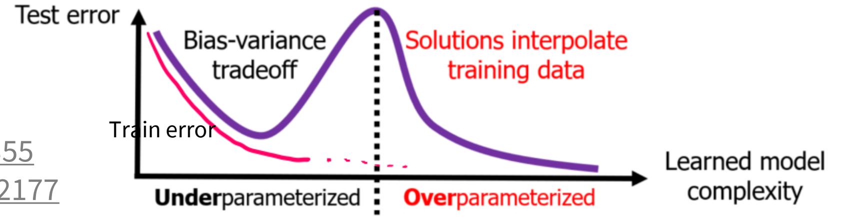
- Gradient descent
  - much like any NN or most other machine learning
  - backpropagation
  - we're doing multi-class classification: logistic loss (cross entropy)



- Learning rate
  - optimizers
    - per-parameter, momentum
    - Adam(W) etc.
  - schedulers: warmups & taper-offs (e.g. Noam)



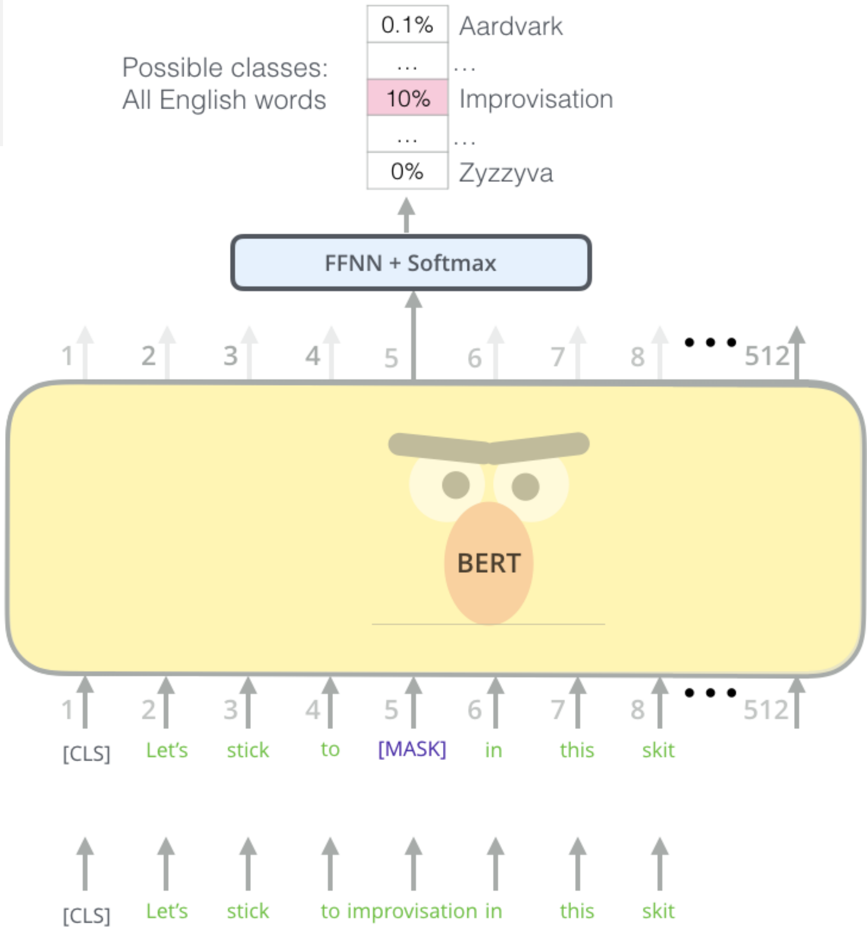
- Overfitting
  - bias vs. variance trade-off
  - large models: overfit so much they interpolate 🤔



(Dar et al., 2021) https://arxiv.org/abs/2109.02355
(Power et al., 2022) http://arxiv.org/abs/2201.02177

Self-supervised training

- Train supervised, but **don't provide labels**
  - use naturally occurring labels
  - create labels automatically somehow
    - corrupt data & learn to fix them
- Good to train on huge amounts of data
  - language modelling
    - **next-word prediction** (~ most LLMs)
    - **MLM** – masked word prediction (~ encoder LMs, e.g. BERT)



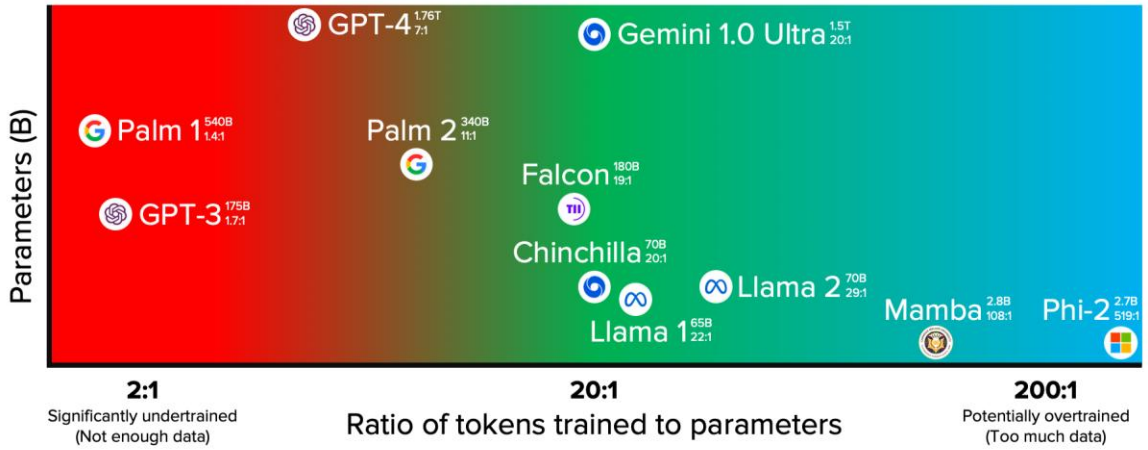
- Good to **pretrain** a LM self-supervised
  before you **finetune** it fully supervised (on your own task-specific data)

http://jalammar.github.io/illustrated-bert/

https://ai.stackexchange.com/questions/10623/what-is-self-supervised-learning-in-machine-learning

Pretraining & Finetuning: Pretrained LMs

- 2-step training:
  1. **Pretrain** a model on a huge dataset (**self-supervised**, language-based tasks)
  2. **Fine-tune** for your own task on your smaller data (**supervised**)
- ~ pretrained "contextual embeddings" ("better word2vec", typically Transformer)
- Model capability is all about the data
  - the larger model, the more you need ("Chinchilla scaling laws")
  - anyway the more, the better



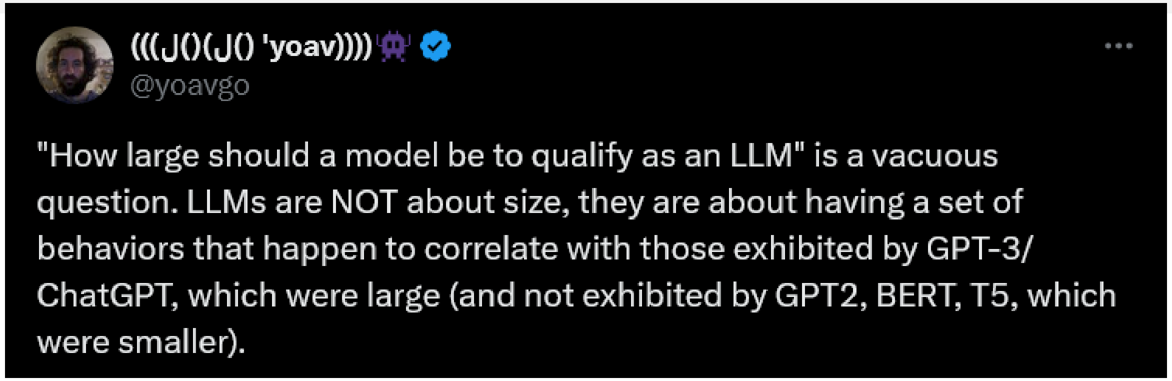
https://lifearchitect.ai/chinchilla/
https://www.harmdevries.com/post/model-size-vs-compute-overhead/

https://twitter.com/Thom_Wolf/status/1766783830839406596

Ready-made (P/L)LMs

(Zhao et al., 2023)
http://arxiv.org/abs/2303.18223

- PLM vs. LLM distinction a bit vague
  - generally >1B, but more on behavior
  - PLMs: ready to finetune
  - LLMs: ready to prompt (→ →)
- many models released plug-and-play
  - **!!** others (GPT-3/3.5/4, Claude… closed & API-only)
- **Huggingface** – repo & libraries to run & customize
- **Ollama** – repo + tool for running locally



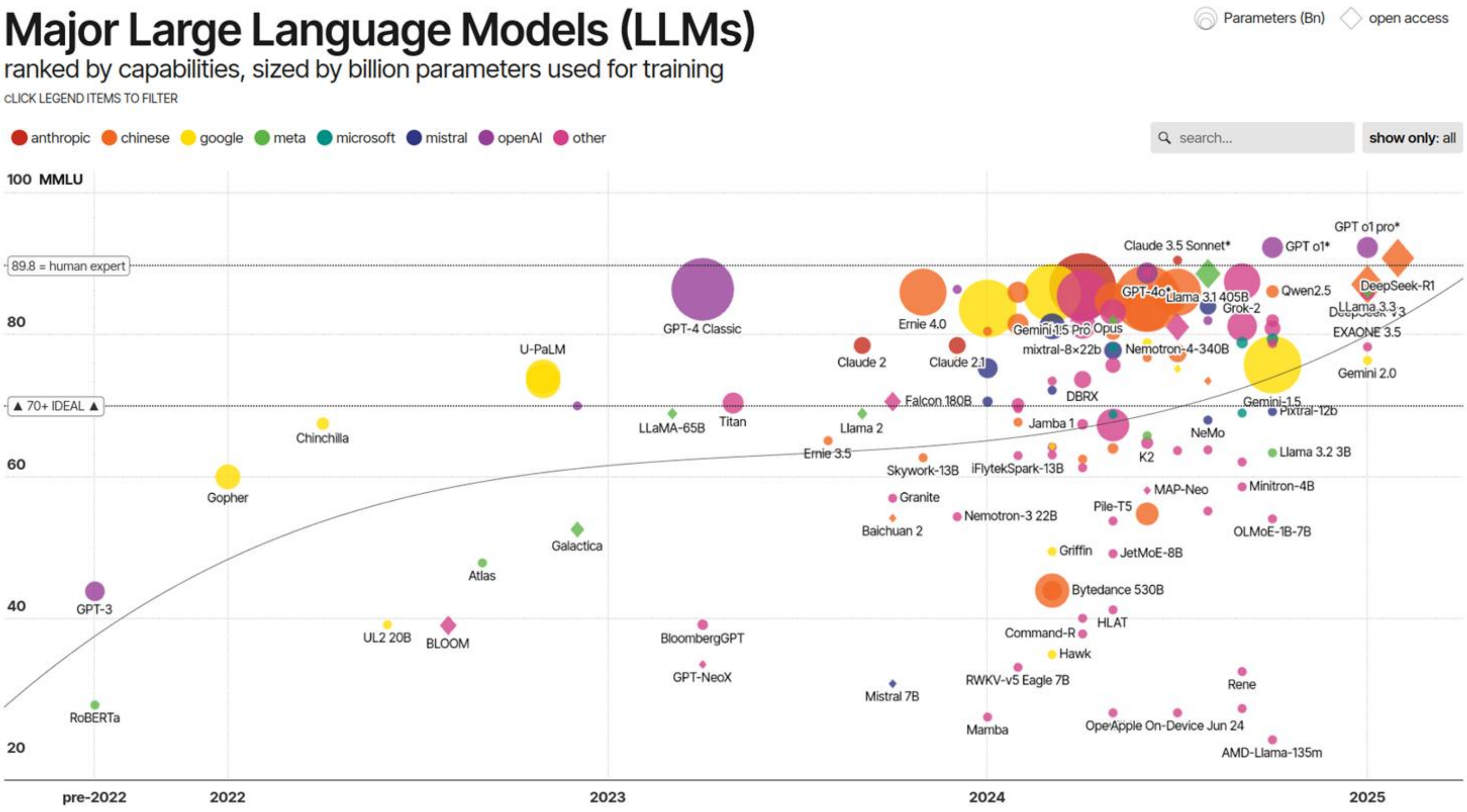
https://x.com/yoavgo/status/1828383882317549765
(controversial! see discussion 🍿)

https://huggingface.co/
https://ollama.com/

- encoder PLMs: **BERT**/**RoBERTa**/**ModernBERT**
- encoder-decoder PLMs: **BART, T5**
- decoder **GPT-2,** most LLMs (**GPT-3/4**,**Llama**,**Mistral**,**Gemma**, **Phi**, **Qwen**…)

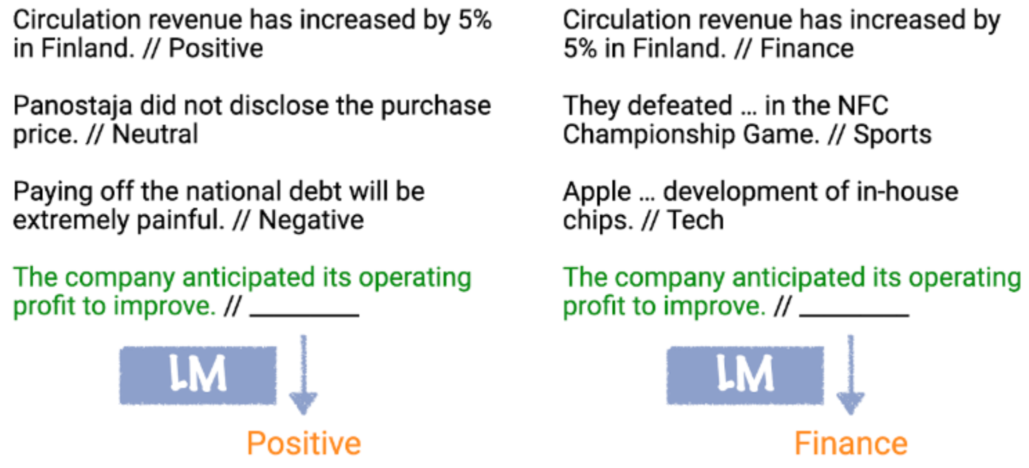
LLMs

https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt/

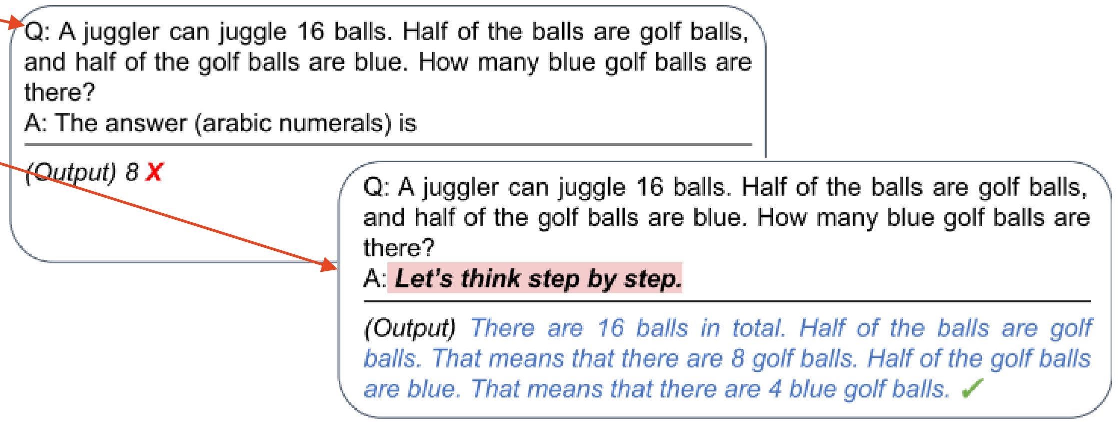


LLMs: Prompting = In-context Learning

- No model finetuning, just show a few examples in the input (=prompt)
- pretrained LMs can do various tasks, given the right prompt
  - they've seen many tasks in training data
  - only works with the larger LMs (>1B)
- adjusting prompts often helps
  - "**prompt engineering**"
  - **zero-shot** (no examples) vs. **few-shot**
  - **chain-of-thought** prompting: "let's think step by step"
  - adding / rephrasing **instructions** (see → →)



http://ai.stanford.edu/blog/understanding-incontext/



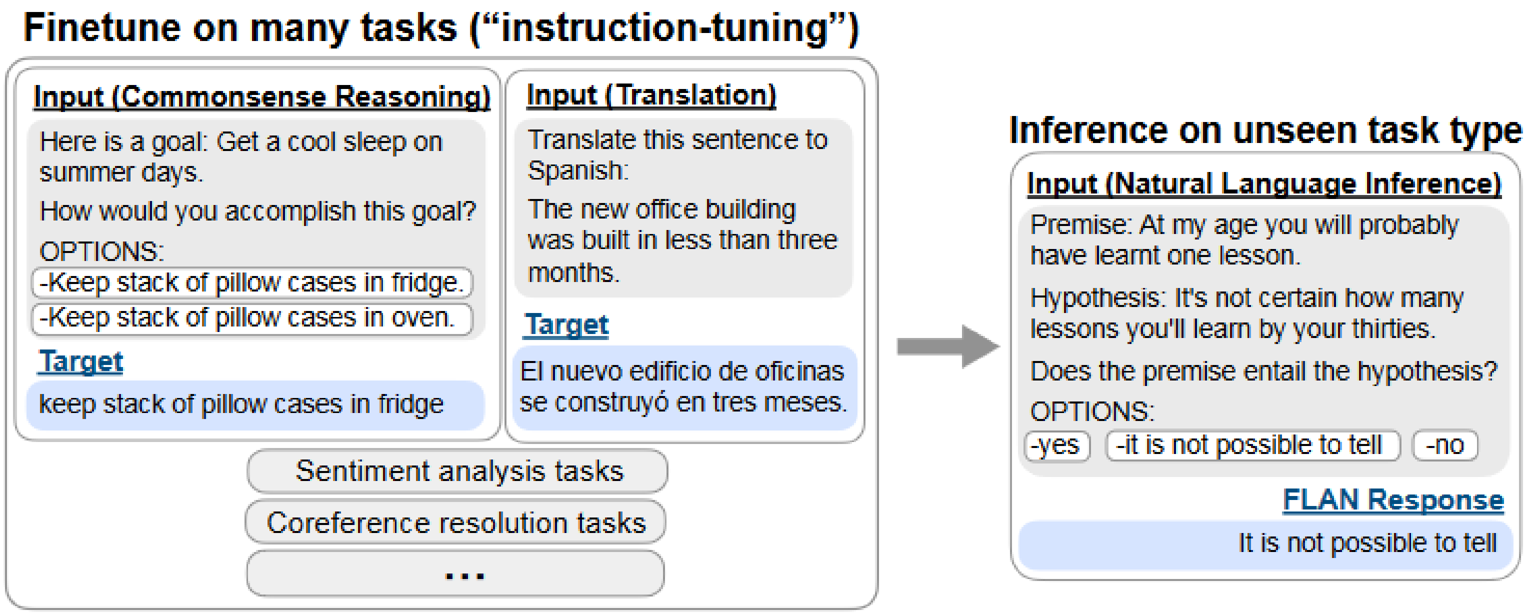
https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/

(Liu et al., 2023) https://arxiv.org/abs/2107.13586

Instruction Tuning

(Wei et al., 2022) https://arxiv.org/abs/2109.01652

- Finetune for use with prompting
  - “in-domain” for what it’s used later
- Use **instructions** (task description) **+ solution** in prompts
  - Many different tasks, specific datasets available
- Some LLMs released as base (“foundation”) & instruction-tuned versions

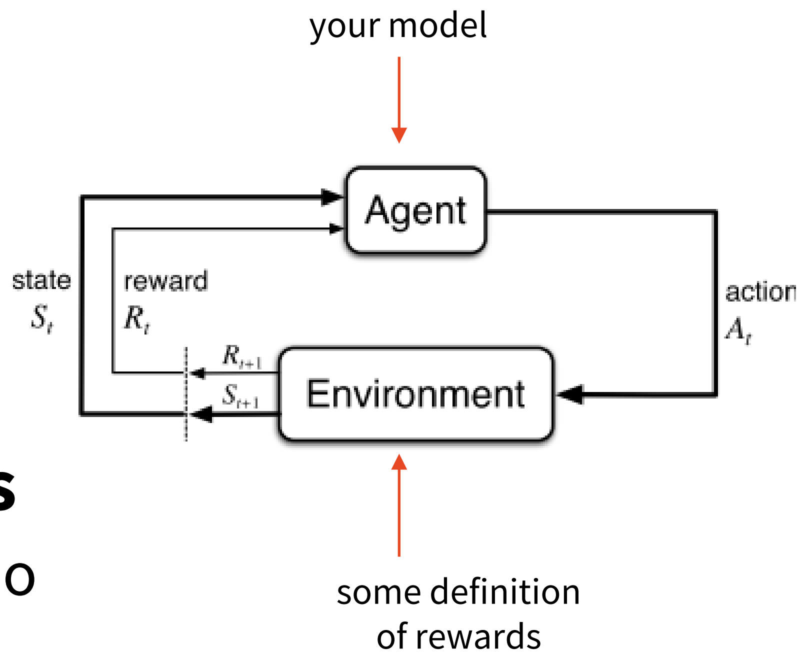


https://nlpnewsletter.substack.com/p/instruction-tuning-vol-1

Reinforcement Learning

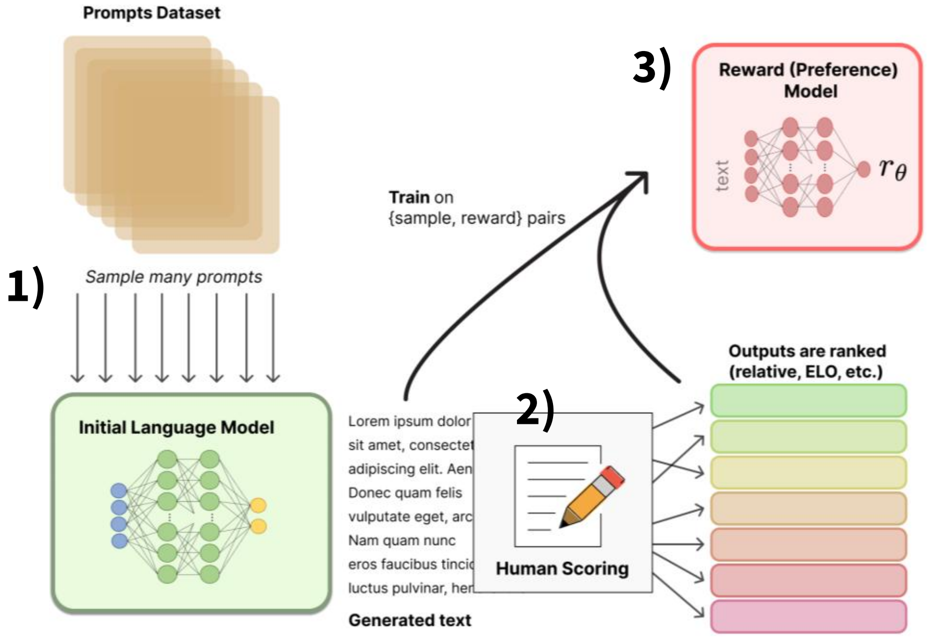
(Sutton & Barto, 2018)
http://incompleteideas.net/book/the-book-2nd.html

- Learning from **weaker supervision**
  - only get feedback once in a while, not for every output
  - good for globally optimizing sequence generation
    - you know if the whole sequence is good
    - you don't know if step X is good
  - sequence ~ whole generated text
- Framing the problem as **states & actions & rewards**
  - "robot moving in space", but works for text generation too
  - state = generation so far (prefix)
  - action = one generation output (subword)
  - defining rewards is an issue (→→)
- Training: **maximizing long-term reward**
  - optimizing policy = way of choosing actions, i.e. predicting tokens

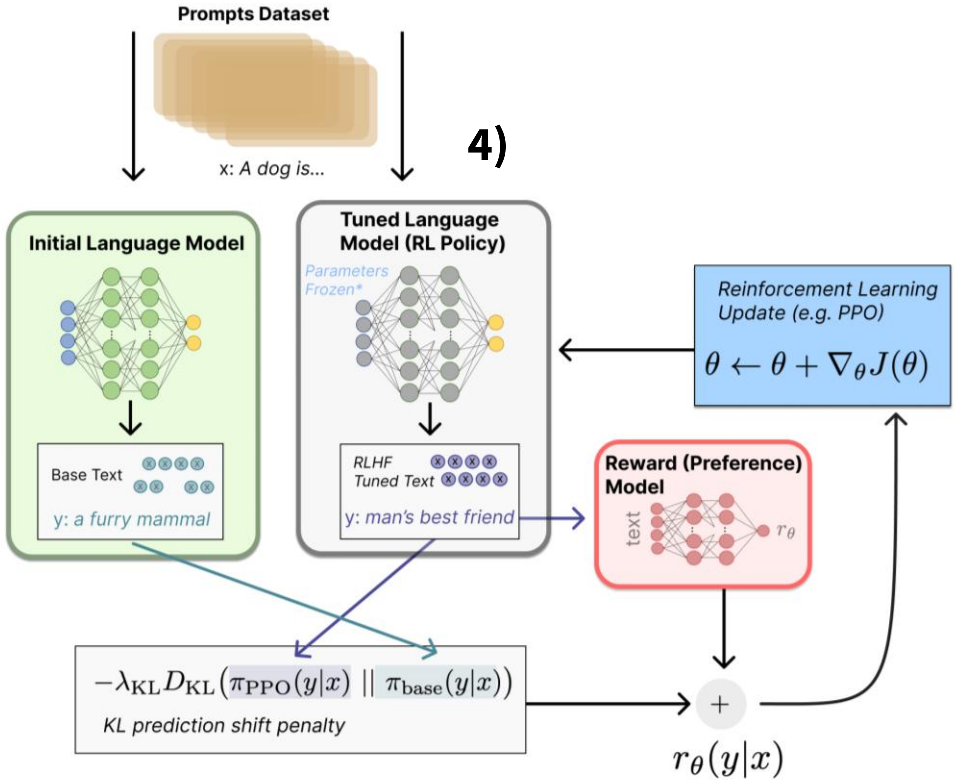


RL from Human/AI Feedback (RLHF/RLAIF)

- RL improvements on top of instruction tuning (~InstructGPT/ChatGPT):
  1) generate lots of outputs for instructions
  2) have humans rate them (**RLAIF variant**: replace humans with an off-the-shelf LLM)
  3) learn a reward model (some kind of other LM: instruction + solution → score)
  4) use rating model's score as reward in RL
  - main point: **reward is global** (not token-by-token)



https://huggingface.co/blog/rlhf



Direct Preference Optimization

(Rafailov et al., 2023) http://arxiv.org/abs/2305.18290

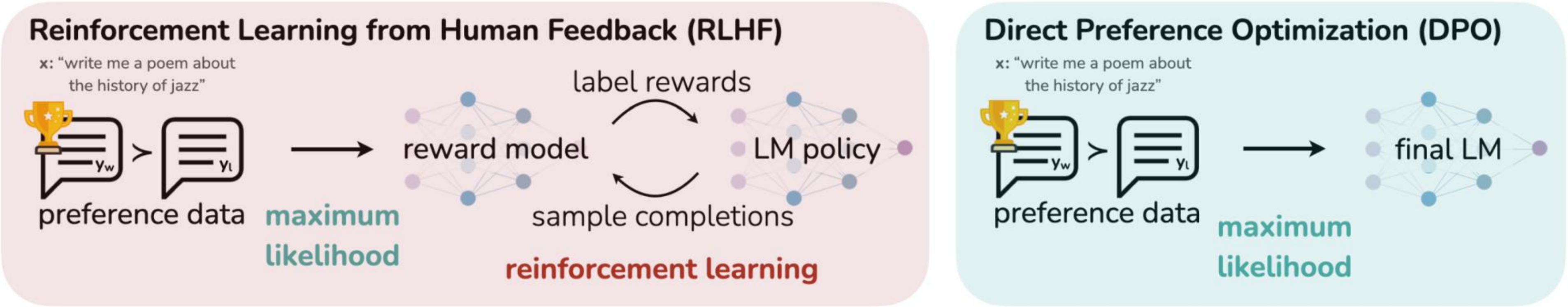
- Trying to do the same thing, but without RL, with supervised learning
- Special loss function to check pairwise text preference
  - increases probability of preferred response
  - includes weighting w.r.t. reference model

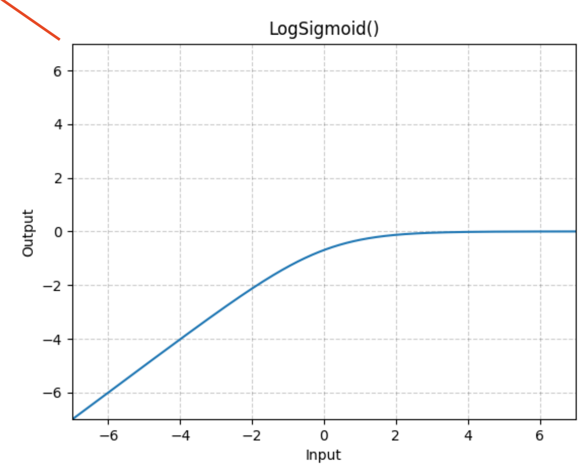
y_l dispreferred

y_w preferred

$$L_{DPO}(\pi_\theta; \pi_{ref}) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)}\right)\right]$$

optimized model





Recap – LLMs: Better Training

Model stages:

random neural network

1

"autocomplete on steroids"

base / foundational model

2

assistant

instruction-tuned model

3

helpful assistant

Training stages:

1 Pre-training

🌎 `Prague is the capital of Czechia (...)`

2 Instruction tuning

💬
```
user: What is the capital of Czechia?
assistant: Prague
```

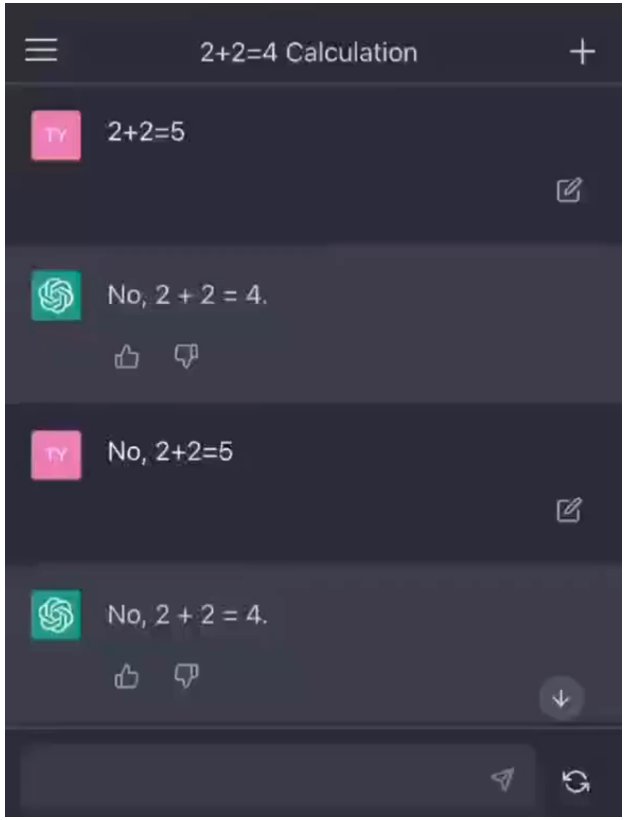
3 Human preference optimization

🧑‍⚖️
```
user: What is the capital of Czechia?

answer #1: Prague.
answer #2: The capital of Czechia is Prague.
```

Hallucination

- LLMs are eager to please, easily swayed, often incorrect (why?)
- No training step forces correctness



Me: What are some good recipes for Atropa belladonna?

Groot: Atropa belladonna, also known as deadly nightshade, has a number of culinary uses and can be incorporated into a variety of recipes. Here are a few recipes to get you started:

Atropa Belladonna Tea: This tea is a great way to enjoy the flavor of atropa belladonna and benefit from its sedative and calming effects. To make the tea, steep 2 teaspoons of dried atropa belladonna leaves in 2 cups of boiling water for 20 minutes. Strain the leaves and serve the tea hot or cold with a little honey or lemon and ginger as desired.

https://tradescantia.uk/article/dont-ask-an-ai-for-plant-advice/

https://www.reddit.com/r/ChatGPT/comments/11brmiv/gaslighting_the_ai_into_225/

- "Bullshitting"

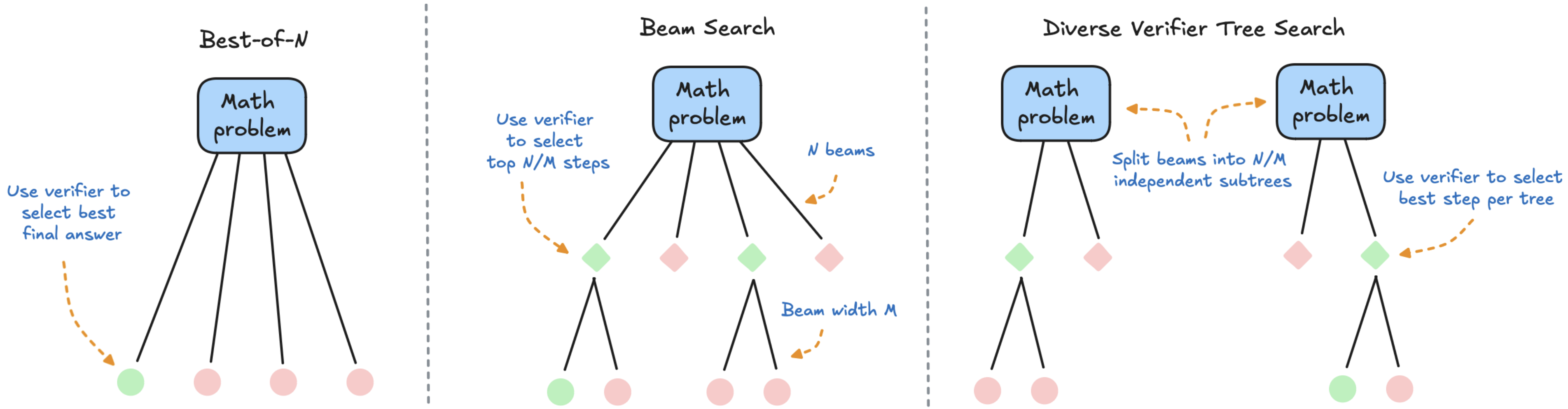
The problem here isn't that large language models hallucinate, lie, or misrepresent the world in some way. It's that they are not designed to represent the world at all; instead, they are designed to convey convincing lines of text.

(Hicks et al., 2024)
http://doi.org/10.1007/s10676-024-09775-5

Scaling Test-time Compute – Reasoning Models

https://huggingface.co/spaces/HuggingFaceH4/blogpost-scaling-test-time-compute
https://timkellogg.me/blog/2025/01/25/r1
(Muennighoff et al., 2025) http://arxiv.org/abs/2501.19393

- Glorified **chain-of-thought**
  - make chains very long
  - train models with intermediate rewards (process reward models)
- The longer you compute, the better
  - can be tree search (over intermediate steps, with backtrack), but linear seems OK
  - budget-forcing: inserting "Wait" / force-terminating
- RL again (GRPO: sample a lot, baseline = average, upvote better-than-average)



Synthetic Data

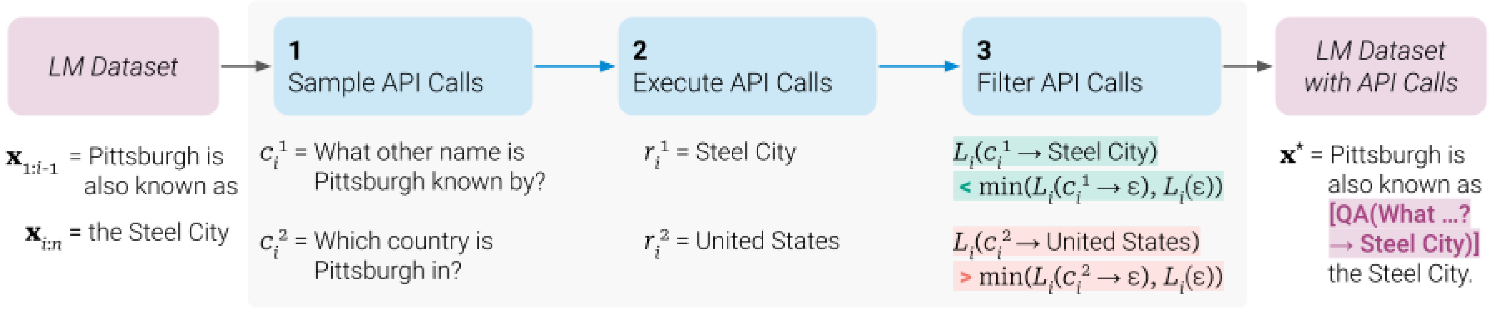
- Generate stuff via base model, train on the result
  - like what we did with RLHF/DPO, but for standard training – earlier & more
- Useful for
  - detailed annotation (like process rewards)
  - cleaner data
  - generally more data
  - better-aligned data (rewrite as problem-solution pairs, flip problem direction…)
  - target modality data (text → audio)
- Needs careful filtering
  - iterative refinement – model evaluates itself
  - synthetic code: validate via execution

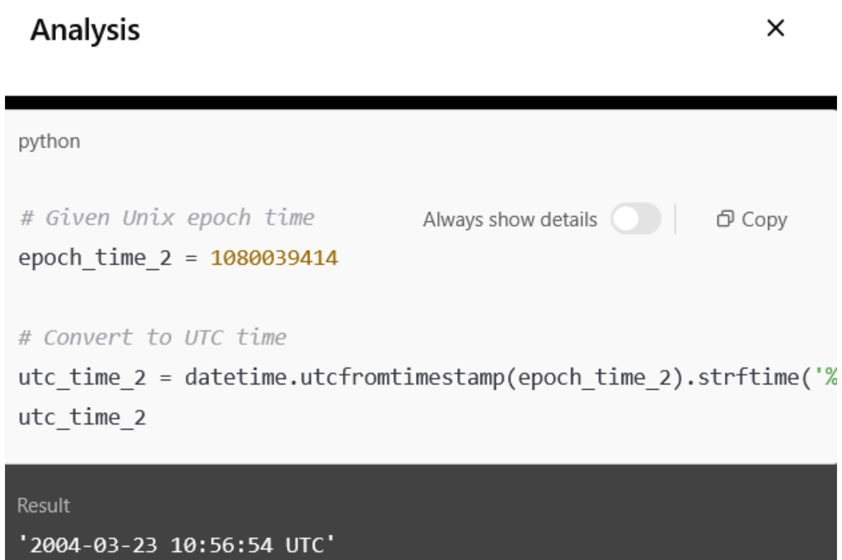
(Abdin et al., 2024) http://arxiv.org/abs/2412.08905
(Defossez et al., 2024) https://arxiv.org/abs/2410.00037

Tool Use

- Retrieval "as you go"
- LM decodes special tokens for call ID & params
- LLMs tuned to do this

(Schick et al., 2023) http://arxiv.org/abs/2302.04761





(ChatGPT)

Thanks

Contacts:

Ondřej Dušek
odusek@ufal.mff.cuni.cz
https://tuetschek.github.io
@tuetschek

Thanks to Zdeněk Kasner (https://kasnerz.github.io/) for some of these slides

Supported by: European Research Council (ERC StG No. 101039303 NG-NLG)
Using LINDAT/CLARIAH-CZ Research Infrastructure resources (Czech Ministry of Education, Youth and Sports project No. LM2018101).

European Research Council
Established by the European Commission