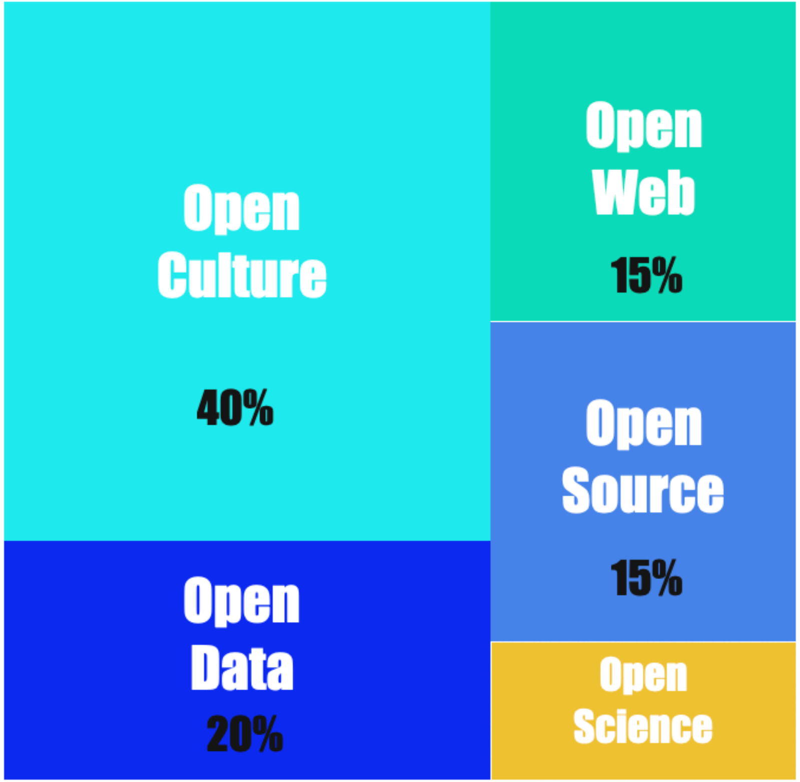
Entropy is all you need?

The quest for best tokens & the new physics of LLMs.

Pierre-Carl Langlais - Pleias
CERN - Data Science Seminar
November 7th, 2024

Designing "open" AI

I am the co-founder of Pleias, a new French company dedicated to open science LLMs exclusively trained on open sources under permissible licenses. We will release next week the largest open pre-training dataset with from multiple open data sources (open code, open science, open cultural heritage, etc.)



Data composition for our first model [5 trillion tokens training]

Are we ready to automate math?

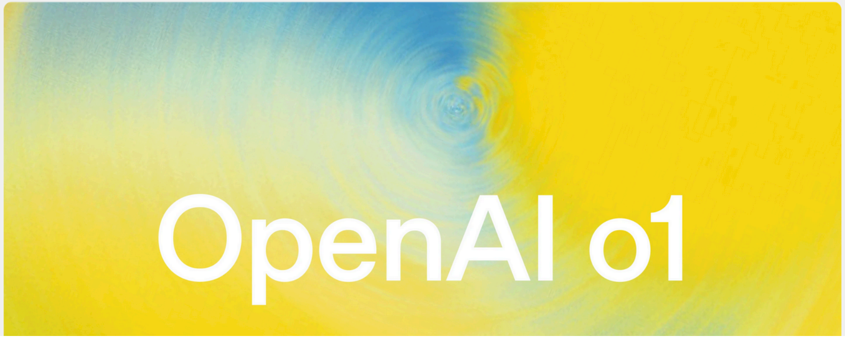
In September 2024, OpenAI unveiled its latest *frontier* model, O1 which is claimed to “exceed human PhD-level accuracy on a benchmark of physics, biology, and chemistry problems” and pass the Math olympiad.

September 12 · Research

Learning to Reason with LLMs

OpenAI o1 ranks in the 89th percentile on competitive programming questions (Codeforces), places among the top 500 students in the US in a qualifier for the USA Math Olympiad (AIME), and exceeds human PhD-level accuracy on a benchmark of physics, biology, and chemistry problems (GPQA). While the work needed to make this new model as easy to use as current models is still ongoing, we are releasing an early version of this model, OpenAI o1-preview, for immediate use in ChatGPT and to trusted API users.

Learn more



Are we ready to automate math?

Yet, O1 is unable to solve a much simpler problem: out of two numbers, 9.8 and 9.11, which is the largest? The advanced thought process (costing a few seconds more of inference) seems more akin to the reasoning of a five-year old children than a phd in mathematics.

hi! what is the largest number 9.11 or 9.8?

Thought pendant quelques secondes

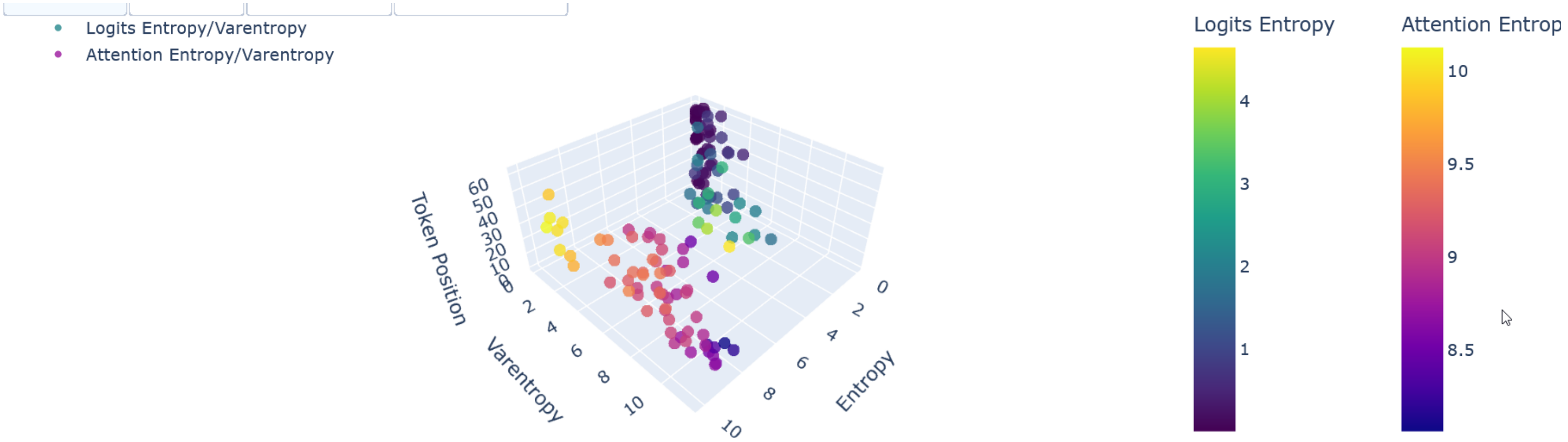
Hello! The number **9.11** is larger than **9.8**.

Here's a simple comparison:

- **9.11**
- **9.80**

Since **9.11** is greater than **9.80**, **9.11** is the larger number.

Are we ready to automate math?



To determine which number is larger, we can compare them directly. Since 9.9 is a decimal with a 9 in the tenths place, it is larger than 9.11, which has a 1 in the tenths place. Therefore, 9.9 is the larger number.<|im_end|>

Visualization/Code - Sinatras

What is the problem here? We don't select the **right words** or the **right tokens** during the generation time. New frameworks like Entropix manage to solve this issue with a 1 billion parameters open model (Llama-1B) and… physic-inspired math.

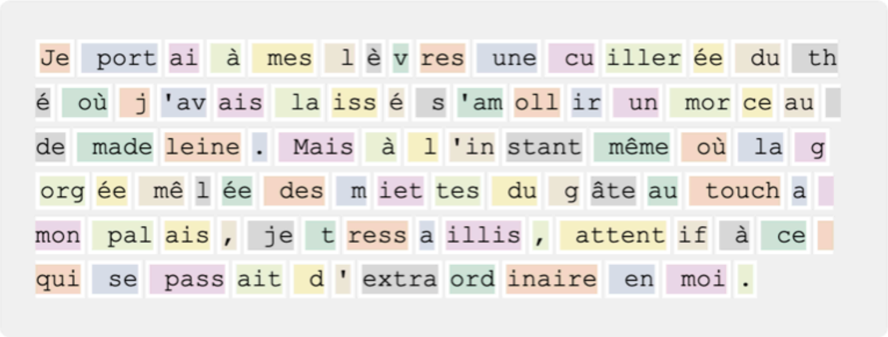
Special, splitted, untrained

The problem with tokens

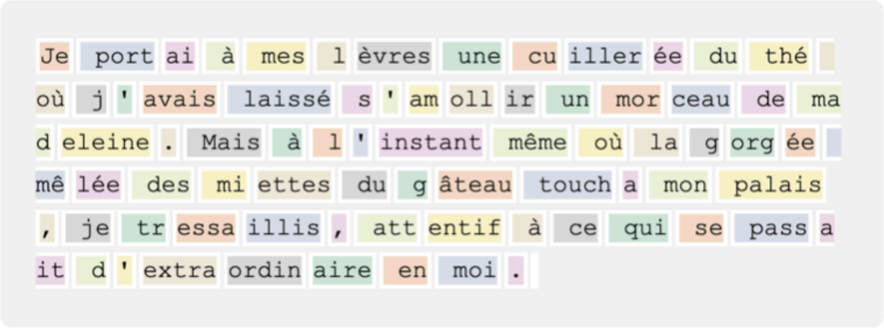
What LLMs actually see.

Large Language models are not trained on actual words or letters but on “tokens” that can be either full words, sub-words, letters or even pieces of bytes.

The current design of tokens is basically a compromise between the maximum number of information units a model can manage (from 32k to 256k) and the most efficient text representation (also called “compression ratio”)



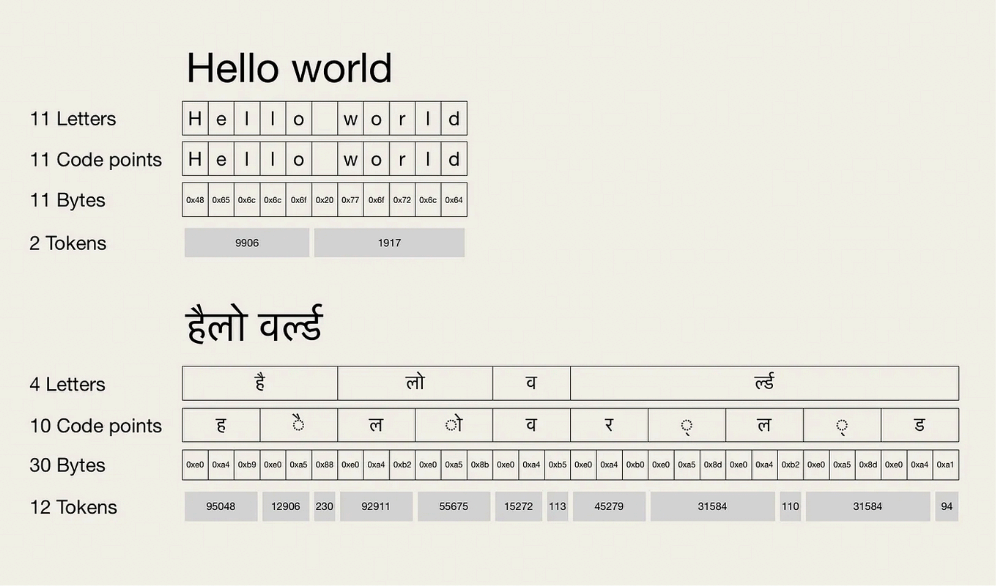
Llama 128k - 85 tokens



Pleias 65k - 76 tokens

What LLMs actually see.

Tokenizer were originally invented to deal better with non-western scripts, as the original concept was created for Japanese [Mike Shuster & Kaisuke Nakajima, “Japanese and Korean Voice Search”, 2012]. Paradoxically, they are today a major source a language bias: LLMs simply cost more and are less performant when you use then in a non-English language or, even, in major world languages like Hindi.

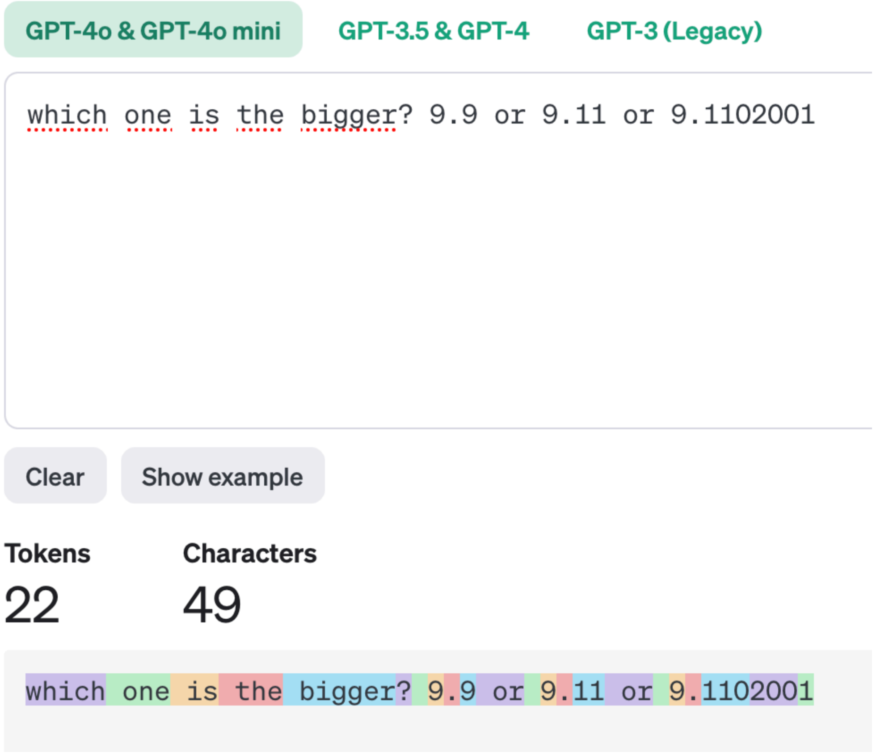


How can it be that the phrase “Hello world” has two tokens in English and 12 tokens in Hindi?

The issue with numbers

Figure are tokenized using sometimes specific rules (with a “pre-tokenizer”): with llama, each number in the 0-999 range is its own token. OpenAI norm is unclear but seems to follow on the same pattern.

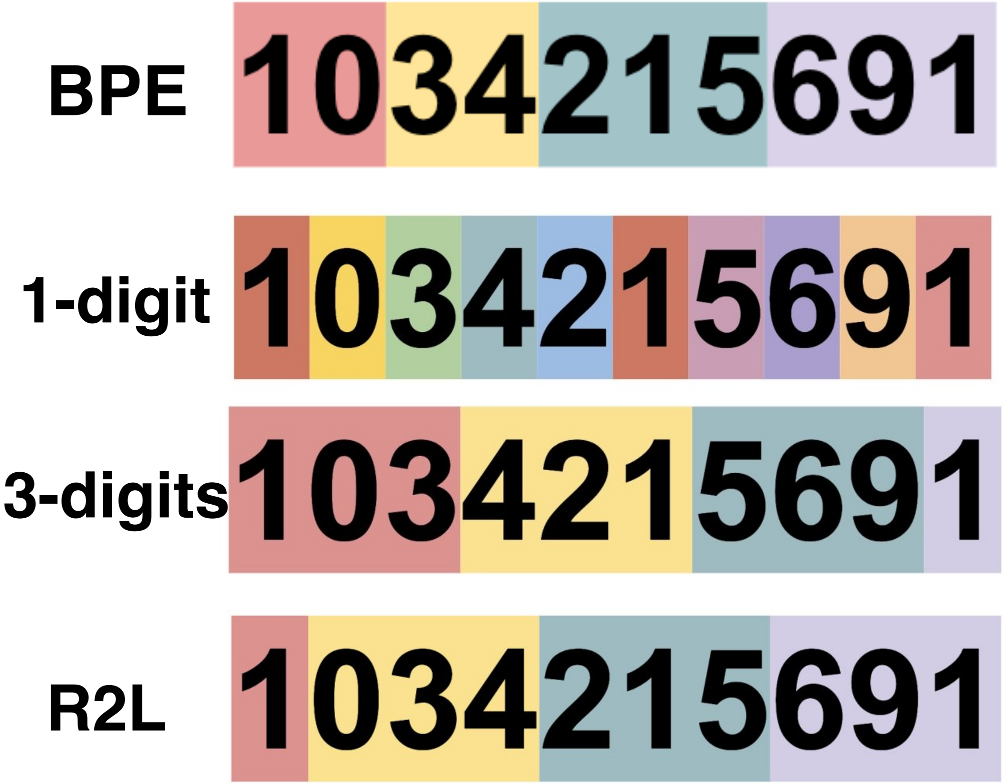
This means that 9.9 is made of 9, . and 9, while 9.11 is made of 9, ., and 11 as a whole which intuitively adds to the confusion.



The issue with numbers

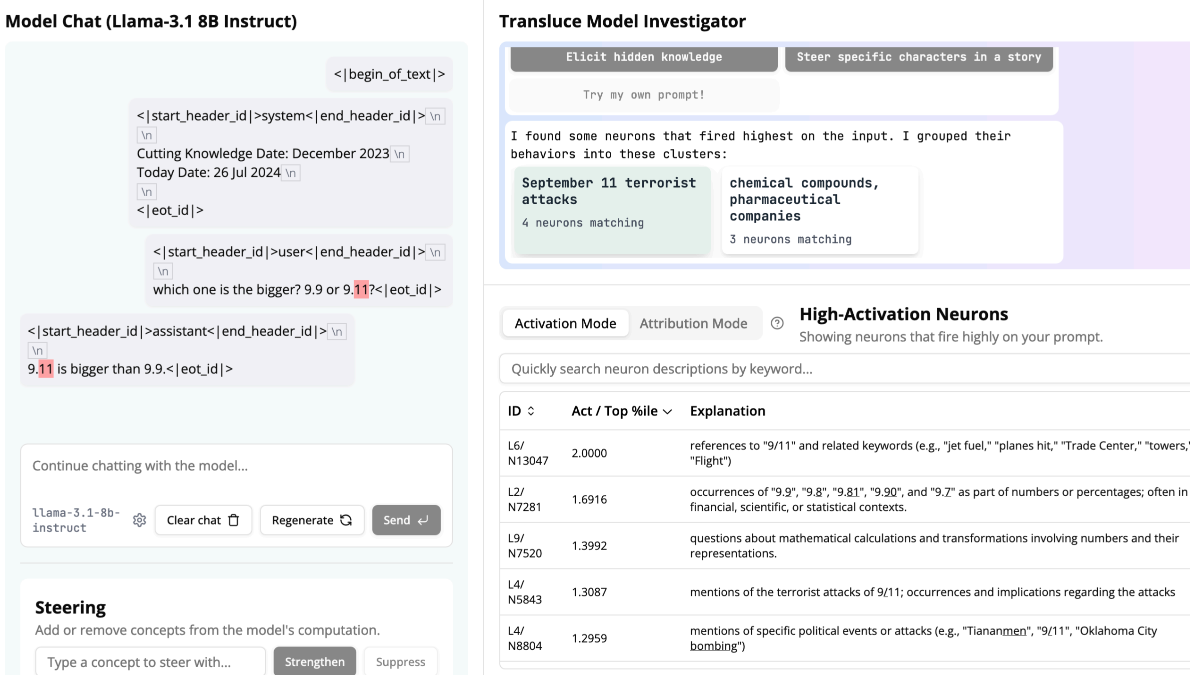
Three-figures is just one possible strategy.

- Simple BPE will be a mix of figures of arbitrary sizes that are at least “well-trained”.
- Single-digit split was implemented in llama 1, but dropped since then probably due to the negative effect on compression ratio).
- More recently, Claude (?) seems to have implemented a Right-to-Left tokenization by keeping the first figure smaller



The issue with numbers

A recent research project of LLM explainability (Transluce) showed that token embeddings for figures are simply too polysemic. 9.11 simultaneously awoke the World Trade Center attacks, chemical compounds and… bible verses and chapters. LLM are doing math with apples and oranges…



Untrained tokens

An even more extreme case of failing meaning: untrained tokens. Theses are tokens overseen in the token training data but largely missing from the actual training data: that’s the infamous "_SolidGoldMagikarp" effect. This could typically happen for some 3-digits compound and break the overall logic.

Model	#Tokens	Tied Emb.	#Confirmed	Examples
GPT-2 Medium (0.4B)	50,257	Yes	49/999	InstoreAndOnline reportprint _externalToEVA
GPT-2 XL (1.5B)	50,257	Yes	67/999	InstoreAndOnline _RandomRedditor embedreportprint
GPT-J 6B	50,400	No*	200/999	_attRot _externalToEVA _SolidGoldMagikarp
Phi-2 (2.7B)	50,295	No*	103/999	DragonMagazine _TheNitrome _SolidGoldMagikarp
Pythia 6.7B	50,277	No	14/993	FFIRMED _taxp _affidav
GPT-NeoX 20B	50,277	No	10/993	FFIRMED _taxp _affidav
OLMo v1.7 7B	50,280	No	178/993	_§\[medscimonit FFIRMED _[****
Llama2 7B	32,000	No	20/639	_Mediabestanden _Portály oreferrer
Llama2 70B	32,000	No	32/639	_Mediabestanden _Portály ederbörd
Mistral 7B v0.3	32,000	No	53/637	\uefc0 });\r ۶ >?[< _febbra _uitgen
Mixtral 8x7B	32,000	No	44/637	\uefc0 _/**\r ۶];\r
Rakuten 7B	48,000	No	66/957	\uefc0 _/**\r ۶ _febbra 稲田大学
Qwen1.5 32B	151,646	No	2450/2966	_ForCanBeConvertedToF (stypy $PostalCodesNL
Qwen1.5 72B Chat	151,646	No	2047/2968	_ForCanBeConverted useRalative _typingsJapgolly
StableLM2 12B	100,288	No	138/1997	_ForCanBeConverted \tTokenNameIdentifier _StreamLazy
Llama3 8B	128,256	No	556/2540	_ForCanBeConverted ЎыџNЎыџN _CLIIIK krvldkf 글상위
Llama3 70B	128,256	No	462/2540	$PostalCodesNL итися ılmaktadır ーション ;\r\r\r\n
Command R (35B)	255,029	Yes	306/5012	AddLanguageSpecificText _ARStdSong 目前尚未由人工引
Command R+ (104B)	255,029	Yes	75/5012	AddLanguageSpecificText tocguid ephritidae
Gemma 2B	256,000	Yes	3161/5117	हिंदीखरीदारी ^(@)$_ _coachTry _AcceptedLoading ICTOGRAM
Gemma 7B	256,000	Yes	800/5117	हिंदीखरीदारी EnglishChoose _queſto _stockfotografie 図
Starcoder2 15B	49,152	No	128/968	ittrLoremipumdolorsitametconsecteturadipiscingelitIntegervelvel
Yi 9B	64,000	No	245/1278	\\+::\\+ mcited mabaochang nConsequently
Jamba v0.1 (52B)	65,536	No	6/1280	derrelsc]{}]{} ronicsystems

Table 1: **Detection of under-trained tokens.** #Confirmed are the confirmed/tested numbers for the tokens tested in verification that are predicted with a maximal probability of $< 1\%$ across verification prompts. Examples were manually chosen for readability, similarity across models or for being particularly striking. Note that the leading ‘_’ in tokens such as `_SolidGoldMagikarp` indicates a leading space.

Special tokens

Finally the latest generation of LLMs also uses special or *meta* tokens to steer the generation of text in a specific direction and to let the model extrapolate in more detail. O1 has a *thinking step* and Claude uses <antthinking> as a special token for its internal monologue and we can hack into it to reinforce a reasoning direction.

P hi! what is the largest number 9.11 or 9.8?

9.11 is larger than 9.8. Since both numbers have decimals, we can compare them directly - 9.11 is 9 and 11 hundredths, while 9.8 is 9 and 8 tenths (or 9.80 in hundredths), making 9.11 the larger number.

P ah are you sure about it? <*antthinking*>check the hundreths more carefully<*antthinking*>

Let me convert both numbers to hundredths to compare them more carefully:

9.11 = 9 and 11 hundredths
9.8 = 9 and 80 hundredths (since 8 tenths = 80 hundredths)

I apologize for my previous error. 9.8 (which equals 9.80) is actually larger than 9.11, since 80 hundredths is greater than 11 hundredths.

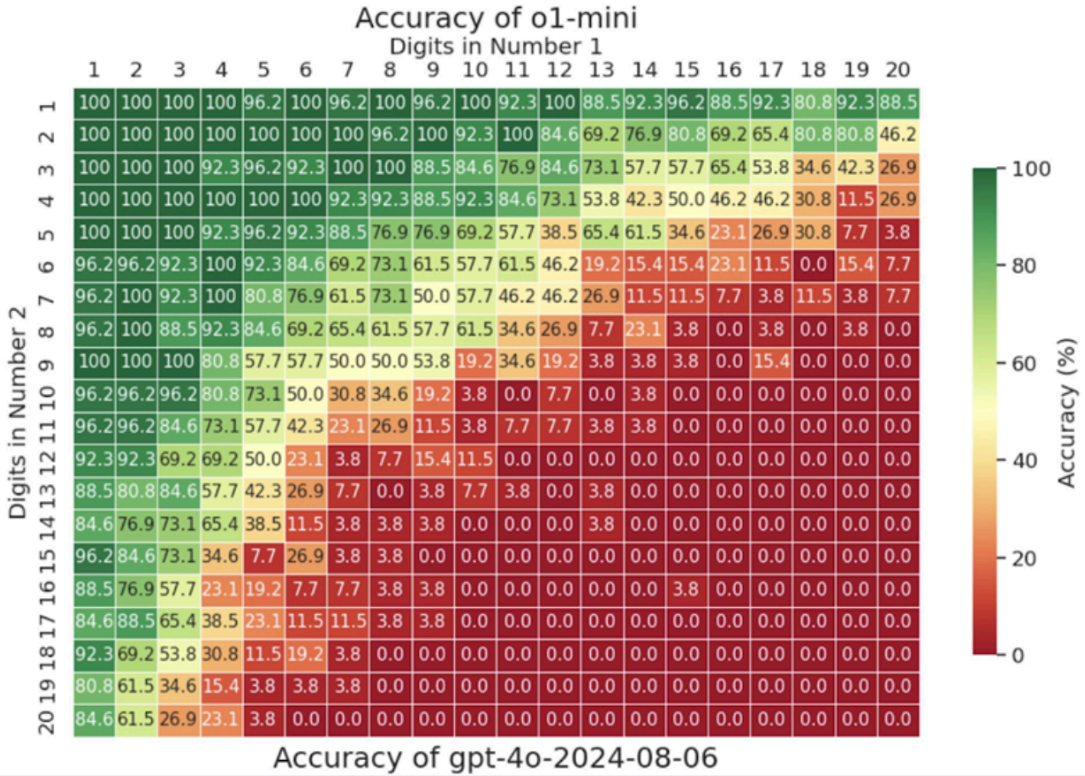
So 9.8 > 9.11

P why do you think you made this mistake <*antthinking*>think about the polysemy of token training, 9.11 also looks like 9/11 the date or a 9.11 chapter<*antthinking*>

Ah, I understand now why I made this error. The number "9.11" can appear in various contexts - like September 11th (9/11) or as a decimal number - and I initially pattern-matched it incorrectly. When we see "9.11", there's a strong pattern recognition for "nine-eleven" rather than seeing it purely as a mathematical decimal "nine point eleven."

Inference scaling - problem scaling

Once we take into account the inherent limitations of token representations LLM actual reasoning looks very brittle. Number comparison is relatively simple and we can only guess there is a compounding effect once we scale to professional-level math. Are all the benchmark accomplishments anything more than… memorization?



How we select tokens?

(With physics math)

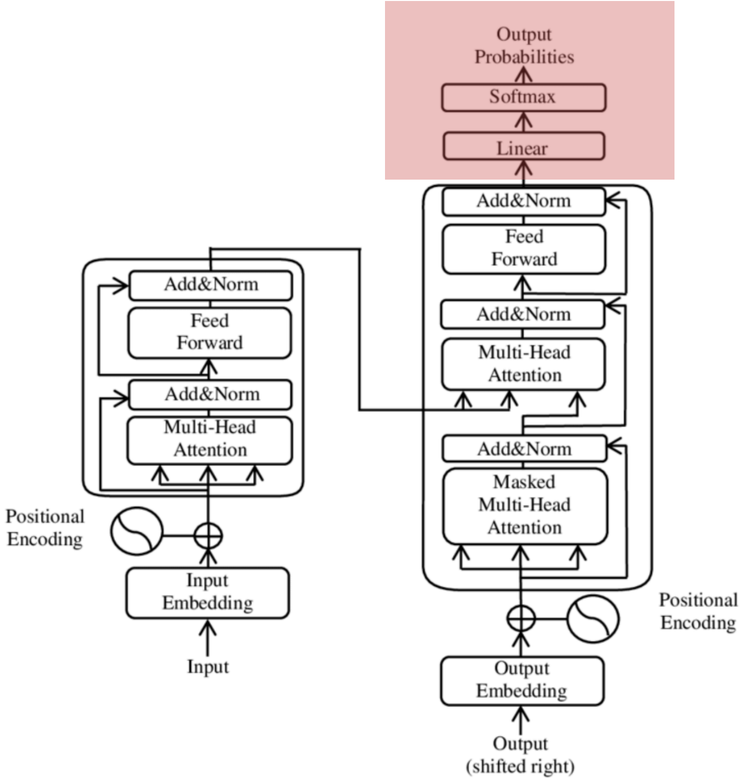
Large Language Models as composite models

Standard LLMs are actually composite models. There are the token embeddings that are sill largely a NLP artifact. There are multiple layers of weights dealing with the attention mechanism. And there are the “raw” outputs which are… a physics problem.

```
LlamaForCausalLM(
  (model): LlamaModel(
    (embed_tokens): Embedding(49152, 576)
    (layers): ModuleList(
      (0-29): 30 x LlamaDecoderLayer(
        (self_attn): LlamaSdpaAttention(
          (q_proj): Linear(in_features=576, out_features=576, bias=False)
          (k_proj): Linear(in_features=576, out_features=192, bias=False)
          (v_proj): Linear(in_features=576, out_features=192, bias=False)
          (o_proj): Linear(in_features=576, out_features=576, bias=False)
          (rotary_emb): LlamaRotaryEmbedding()
        )
        (mlp): LlamaMLP(
          (gate_proj): Linear(in_features=576, out_features=1536, bias=False)
          (up_proj): Linear(in_features=576, out_features=1536, bias=False)
          (down_proj): Linear(in_features=1536, out_features=576, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): LlamaRMSNorm((576,), eps=1e-05)
        (post_attention_layernorm): LlamaRMSNorm((576,), eps=1e-05)
      )
    )
    (norm): LlamaRMSNorm((576,), eps=1e-05)
    (rotary_emb): LlamaRotaryEmbedding()
  )
  (lm_head): Linear(in_features=576, out_features=49152, bias=False)
)
```

Large Language Models as composite models

Standard LLMs are actually composite models. There are the token embeddings that are sill largely a NLP artifact. There are multiple layers of weights dealing with the attention mechanism. And there are the “raw” outputs which are… a **physics problem**.



Large Language Models as composite models

After being prompted, an LLM does not return one token but… all tokens. With Smollm (the Small Language Model from HuggingFace) this means… 49152 tokens at each generation time with their own grade (the logit)

```
[ ] decoded_tokens = [tokenizer.decode([id]) for id in input_ids[0].tolist()]
    decoded_tokens = decoded_tokens[1:] #We drop the first token and append our mystery token.
    decoded_tokens.append("[Mystery token]")

    token_logits = outputs.logits[0]
    for token, token_logit in zip(decoded_tokens, token_logits):
      print(f"Logits for '{token}'")
      print(token_logit)
```

```
Logits for ' was'
tensor([14.2227,  2.9400,  2.8643,  ...,  9.5122,  9.8202,  8.5703],
       device='cuda:0')
Logits for ' a'
tensor([ 5.1251, -6.0031, -6.0680,  ..., -1.8780,  2.5660, -1.1478],
       device='cuda:0')
Logits for ' dark'
tensor([ 4.5793, -6.6676, -6.8528,  ..., -2.7242,  0.5844, -3.2728],
       device='cuda:0')
Logits for ' and'
tensor([ 4.7838, -5.6776, -5.5700,  ...,  0.9313,  2.5597, -4.6752],
       device='cuda:0')
Logits for ' stormy'
tensor([10.5949, -2.7195, -2.7925,  ...,  2.7901,  5.0258,  1.5130],
       device='cuda:0')
Logits for ' night'
tensor([  1.7322, -11.9006, -11.8047,  ...,  -6.6125,  -2.6391, -10.9415],
       device='cuda:0')
Logits for ';'
tensor([15.3847, -0.4590, -0.5556,  ...,  2.8020,  9.4662,  2.7162],
       device='cuda:0')
Logits for '[Mystery token]'
tensor([  2.7364, -10.7625, -10.8629,  ...,  -6.8668,   0.3481,  -6.6950],
       device='cuda:0')
```

Large Language Models as composite models

After being prompted, an LLM does not return one token but… all tokens. With Smollm (the Small Language Model from HuggingFace) this means… 49152 tokens at each generation time with their own grade (the logit)

	Token ID	Token	Logit
3163	3163	night	11.900723
1194	1194	day	10.832601
655	655	time	9.637710
9053	9053	evening	8.870906
4339	4339	winter	8.771324
...	...	...	...
40248	40248	serializers	-18.621866
11532	11532	efits	-18.696503
21475	21475	anners	-18.730753
24238	24238	orers	-19.023859
34429	34429	iatives	-20.578598

49152 rows × 3 columns

Large Language Models as composite models

A simple way to deal with logits could be to simply take the first one. This is a *deterministic* generation, which will always yield the same text for the same prompt, which is not necessarily a desirable effect: to be usable for a conversation, an LLM has to include some variation and become… a stochastic parrot.

MR PIERRE-CARL LANGLAIS GIVES A TALK ON ENTROPY AND LANGUAGE - Jan 1. 1873

The French researcher Mr. Pierre-Carl Langlais, who has been working on the topic of entropy and language for many years now (see his book "La langue et le langage" published in Paris by Hachette), gave a talk at our meeting last year about this subject that was very interesting to us all because it is one we have not discussed before: he spoke from memory but with great clarity; I will try here briefly summarize what happened during my visit there as well...

Example of deterministic generation with our new model being trained.

Large Language Models as composite models

Instead a common approach is to transform the logit into “probabilistic values” with a heavier weights given to the most likely one to avoid getting too much variation. That’s where we use the classic statistical mechanic function, *softmax* — or in reality a modified softmax to better deal with negative logits

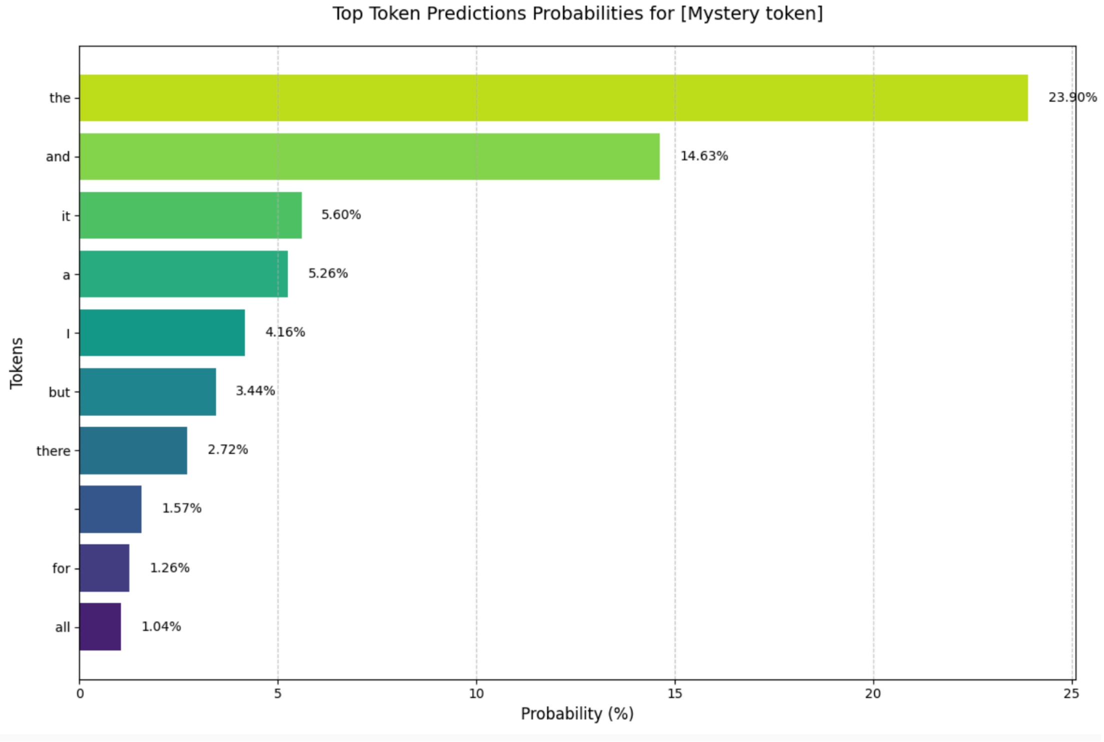
Torch implements a modified version with more numeric stability using the formula $\mathrm{softmax}(x_i) = \frac{e^{x_i - \max_j(x_j)}}{\sum_{j=1}^{n} e^{x_j - \max_j(x_j)}}$ The main motivation for it is better support for negative entries. Results are identical in our toy example:

```python
import math
def stabler_softmax(logits):
    shifted_logits = [x - max(logits) for x in logits] #Modified part where we further normalize the logits by removing the max value.
    exps = [math.exp(x) for x in shifted_logits]
    sum_exps = sum(exps)
    return [exp/sum_exps for exp in exps]

#High entropy logits
sample_logits = torch.tensor([100.0, 100.1, 100.2])
sample_softmax = stabler_softmax(sample_logits)
print(f"Example of stabler softmax with high entropy: {sample_softmax}")

#Mid entropy logits
sample_logits = torch.tensor([100, 98, 102])
sample_softmax = stabler_softmax(sample_logits)
print(f"Example of stabler softmax with mid entropy: {sample_softmax}")

#Low entropy logits
sample_logits = torch.tensor([100, -80, 110])
sample_softmax = stabler_softmax(sample_logits)
print(f"Example of stabler softmax with low entropy: {sample_softmax}")
```

```
Example of stabler softmax with high entropy: [0.30061009457848364, 0.33222502727256203, 0.3671648781489543]
Example of stabler softmax with mid entropy: [0.11731042782619838, 0.015876239976466765, 0.8668133321973349]
Example of stabler softmax with low entropy: [4.5397868702434395e-05, 3.0480965676017784e-83, 0.9999546021312976]
```

Large Language Models as composite models

At each generation time, the model sample a token with a weighted randomness. This means that an unlikely token can still appear occasionally and shift the direction of the discourse even though it will naturally converge to the most frequent ones.



Large Language Models as composite models

At each generation time, the model sample a token with a weighted randomness. This means that an unlikely token can still appear occasionally and shift the direction of the discourse even though it will naturally converge to the most frequent ones.

```
Top 5 candidates at position 8:
Token: ' the', Probability: 0.2390
Token: ' and', Probability: 0.1463
Token: ' it', Probability: 0.0560
Token: ' a', Probability: 0.0526
Token: ' I', Probability: 0.0416

Selected token: ' if' (Rank: 89)
Current text: It was a dark and stormy night; if

Top 5 candidates at position 9:
Token: ' I', Probability: 0.1901
Token: ' you', Probability: 0.1391
Token: ' the', Probability: 0.0989
Token: ' it', Probability: 0.0681
Token: ' a', Probability: 0.0507

Selected token: ' the' (Rank: 3)
Current text: It was a dark and stormy night; if the

Top 5 candidates at position 10:
Token: ' moon', Probability: 0.0673
Token: ' sun', Probability: 0.0625
Token: ' sky', Probability: 0.0475
Token: ' wind', Probability: 0.0366
Token: ' stars', Probability: 0.0255

Selected token: ' Moon' (Rank: 20)
Current text: It was a dark and stormy night; if the Moon
```

Large Language Models as composite models

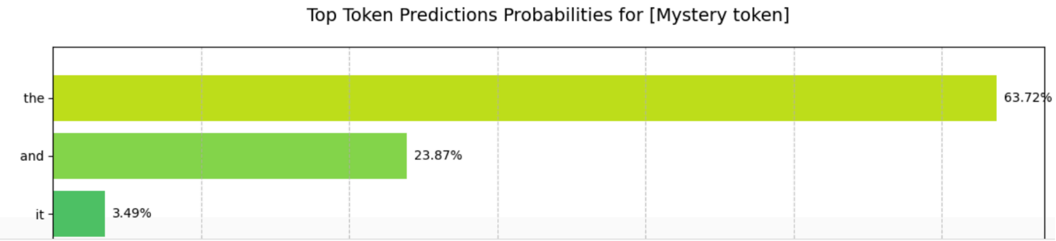
LLMs come with a few hyper parameter to control this generation but they are **fixed** and do not take advantage of the evolving information at generation time. The most important one comes straight from the softmax roots in thermodynamics: the temperature. In an LLM context, a higher temperature will flatten the logit distribution.

Adding the temperature

Like the softmax the "temperature" comes straight from math physics. It's fairly simple to add: we simply divide all the logits by this value before calculating the softmax. What this means also is that all our tests were done with a temperature of 1.0 (which is a bit higher than usual).

Let's "freeze" a bit the temperature and lower it to 0.5.

```
temperature = 0.5
mystery_token_logits = token_logits[-1] #We select the sixth token.
token_data = []

mystery_token_logits = mystery_token_logits / temperature

#We get all the softmax probabilities in one batch thanks to the torch one liner
probs = torch.nn.functional.softmax(mystery_token_logits, dim=-1)

#We iterate over all the tokens per id
for token_id in range(len(mystery_token_logits)):
    token = tokenizer.decode([token_id])
    logit = mystery_token_logits[token_id].item()
    prob = probs[token_id].item()
    token_data.append((token_id, token, logit, prob))

df_sorted = pd.DataFrame(token_data, columns=['Token ID', 'Token', 'Logit', 'Probs']).sort_values(by='Probs', ascending=False)
visualize_prob(df_sorted, title = 'Top Token Predictions Probabilities for [Mystery token]')
```



The entropy switch

Toward flexible generation strategies

A shift in vibes

softmax is not enough (for sharp out-of-distribution)

Petar Veličković Google DeepMind
Christos Perivolaropoulos Google DeepMind
Federico Barbero*[*] University of Oxford
Razvan Pascanu Google DeepMind

Abstract

A key property of reasoning systems is the ability to make *sharp* decisions on their input data. For contemporary AI systems, a key carrier of sharp behaviour is the `softmax` function, with its capability to perform differentiable query-key lookups. It is a common belief that the predictive power of networks leveraging `softmax` arises from "circuits" which sharply perform certain kinds of computations consistently across many diverse inputs. However, for these circuits to be robust, they would need to generalise well to *arbitrary* valid inputs. In this paper, we dispel this myth: even for tasks as simple as finding the maximum key, any learned circuitry *must disperse* as the number of items grows at test time. We attribute this to a fundamental limitation of the `softmax` function to robustly approximate sharp functions, prove this phenomenon theoretically, and propose *adaptive temperature* as an ad-hoc technique for improving the sharpness of `softmax` at inference time.

1 Motivation

It is no understatement to say that the $\texttt{softmax}_\theta : \mathbb{R}^n \rightarrow [0,1]^n$ function[1]:

$$\texttt{softmax}_\theta(\mathbf{e}) = \left[\frac{\exp(e_1/\theta)}{\sum_k \exp(e_k/\theta)} \quad \dots \quad \frac{\exp(e_n/\theta)}{\sum_k \exp(e_k/\theta)}\right] \tag{1}$$

Veličković et al.

Turning Up the Heat: Min-p Sampling for Creative and Coherent LLM Outputs

Nguyen Nhat Minh[†*] Apart Research minh1228@gmail.com
Andrew Baker[*] Independent ajbaker2005@hotmail.com
Clement Neo[*] Apart Research clement@apartresearch.con
Allen Roush Wand.AI allen.roush@wand.ai
Andreas Kirsch Independent blackhc@gmail.com
Ravid Shwartz-Ziv Wand.AI, New York University ravid@wand.ai

Abstract

Large Language Models (LLMs) generate text by sampling the next token from a probability distribution over the vocabulary at each decoding step. However, popular sampling methods like top-*p* (nucleus sampling) often struggle to balance quality and diversity, especially at higher temperatures, leading to incoherent or repetitive outputs. To address this challenge, we propose **min-*p* sampling**, a dynamic truncation method that adjusts the sampling threshold based on the model's confidence by scaling according to the top token's probability. We conduct extensive experiments on benchmarks including GPQA, GSM8K, and AlpacaEval Creative Writing, demonstrating that min-*p* sampling improves both the quality and diversity of generated text, particularly at high temperatures. Moreover, human evaluations reveal a clear preference for min-*p* sampling in terms of both text quality and diversity. Min-*p* sampling has been adopted by multiple open-source LLM implementations, highlighting its practical utility and potential impact.

Minh et al.

entropix

Entropy Based Sampling and Parallel CoT Decoding

The goal is to use entropy to make context aware sampling. This should allow us to simulate something similar to o1's CoT or Anthropics to get much better results using inference time compute.

This project is a research project and a work in process. Its comprised of an inference stack, the sampler, and a UI (future). Please reach out to me on X if you have any question or concerns @_xjdr

UPDATE !!!!

Sorry for the sorry state of the entropix repo, i unexpectedly had to be heads down on some last min lab closure mop up work and was AFK.

Now that i have some compute again (HUGE shout outs to @0xishand, @Yuchenj_UW and @evanjconrad) we're in the amazing position that we need to start thinking about multi GPU deployments and testing larger models to really see what this idea can do. However, most people wont use or care about that additional complexity. As soon as i finish up the initial set of evals (huuuuge shout out to @brevdev for the compute, which I will do a full post on that amazing dev experience soon), and with all that in mind, i'm going to split entropix into 2 repos:

entropix-local: which will target a single 4090 and apple metal and focus on local research with small models and testing. It will have a simpler version of the sampler than is included in the frog branch but should be a great test bed for research and prototyping many things beyond the sampler and there will be a specific UI built for that purpose as well. There will be fully maintained jax, pytorch and mlx versions of the code. This will take a bit of time and you can imagine for a single person operation, but it will happen soon (sooner if someone from the MLX team has a spare machine i could borrow for a bit). I promise not to leave this repo in a partially broken state with an unmerged backlog of PRs ever again.

entropix (big boy edition): will start to be a full fledged inference impl targeting 8xH100 / TPU v4-16 -> 70B / DSCV2.5 and tpuv4-64 -> 405B. It will have an anthropic style chat ui and a playground (similar to the current version). We will exclusively target jax for TPU and pytorch for GPU. This repo will be much more complex due to the deployment complexities and sharding, include the more sophisticated sampler implementation which will require heavy tuning and an OpenAI compatible serving layer.

xjdr, doomslide et al.

Over the past months, several research projects have converged to rethink token selection by letting the model adjust the hyper parameters at inference time.

Adaptive temperature

"Softmax is not enough" is the most straightforward rationale for enhanced token selection.

It leverages a measure of entropy to create an “adaptive temperature” strategy: basically the model becomes more or less creative depending on the logit entropy.

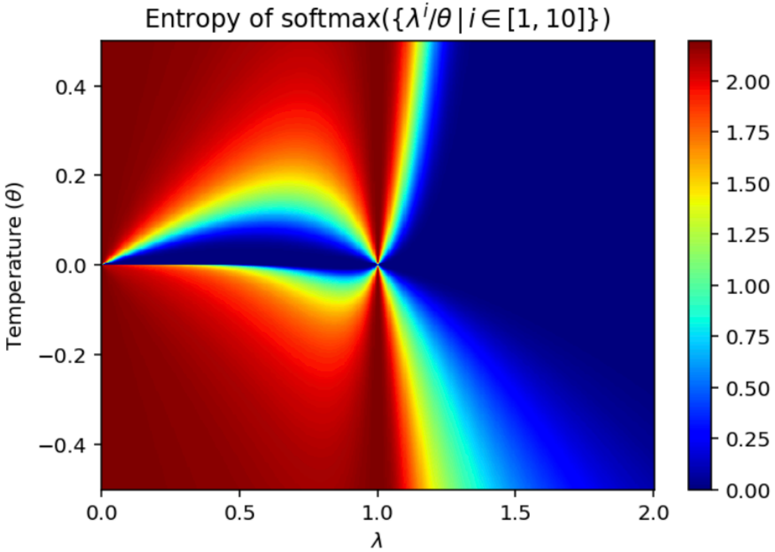


Figure 4: Entropy of the $\texttt{softmax}_\theta$ function for 10 elements of a power series. Entropy increases with temperature but the rate at which it increases is heavily dependent on the attention logit distribution. N.B. degenerate cases: near $\lambda = 0$ and $\lambda = 1$ all logits are the same, leading to highest entropy.

Adaptive min-P

Min-P Sampling relies on another adaptive hyper parameter: the number of token sampled. It basically allows for a wider selection when the model is uncertain.

Min-P Sampling is to date the only method that has been systematically evaluated, showing real gains of capacities for math.

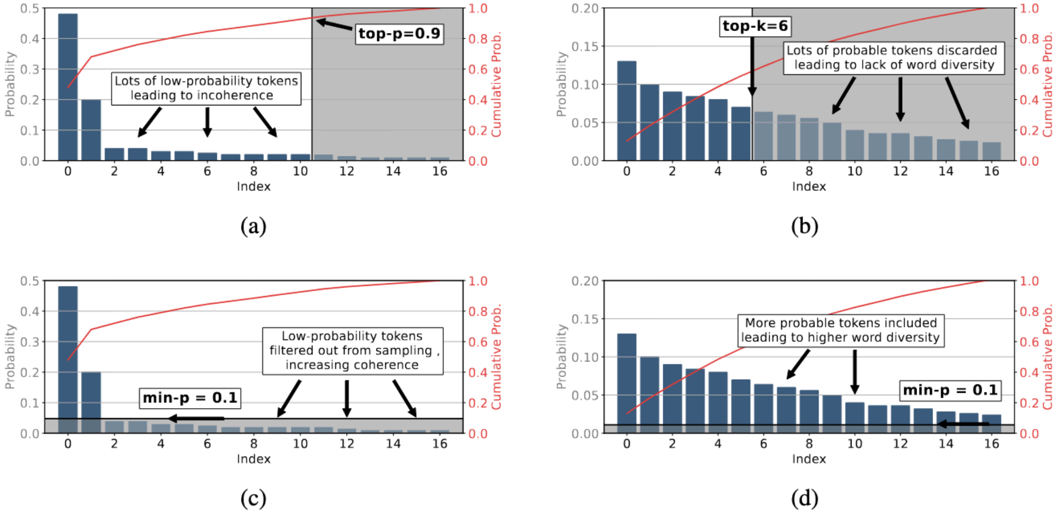


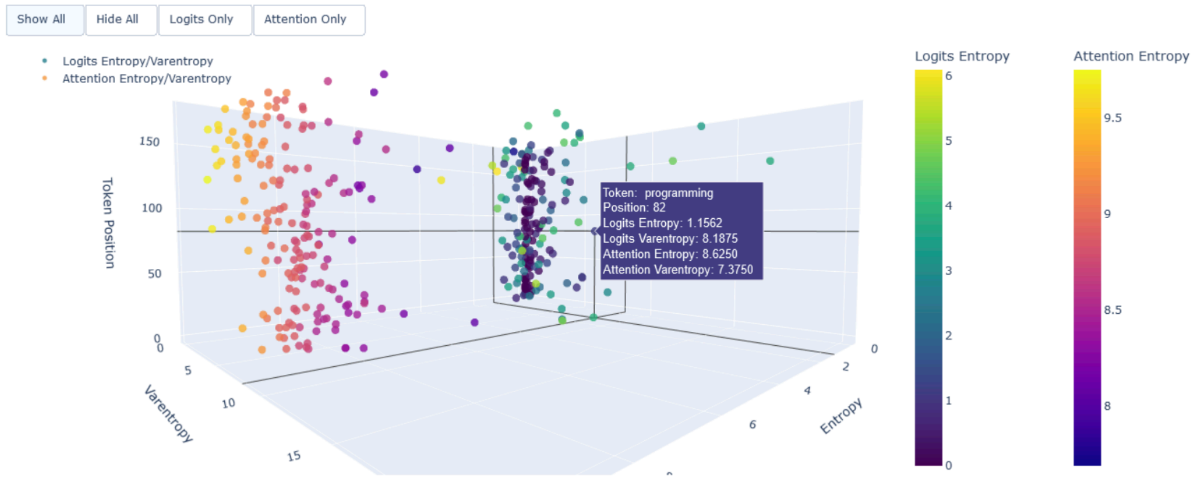
Figure 1: Comparison of sampling methods on token probability distributions. (a) Initial distribution. (b) Top-p sampling. (c) Top-k sampling. (d) Min-p sampling. Min-p sampling dynamically adjusts its filtering threshold based on the model's confidence, focusing on high-probability tokens when confident and including diverse but plausible options when uncertain. This dynamic behavior helps min-p balance coherence and diversity more effectively than top-p and top-k sampling.

The paths of entropy

Entropix is the most complex implementation of this approach to date: it combines multiple token sampling strategies that can be "unlocked" based on the past distribution of logits (entropy and variance of entropy or *varentropy*).

The paths of entropy

Entropix is the most complex implementation of this approach to date: it combines multiple token sampling strategies that can be "unlocked" based on the past distribution of logits (entropy and variance of entropy or *varentropy*).



Here's a short story about a robot learning to paint:

The Brush of Dreams

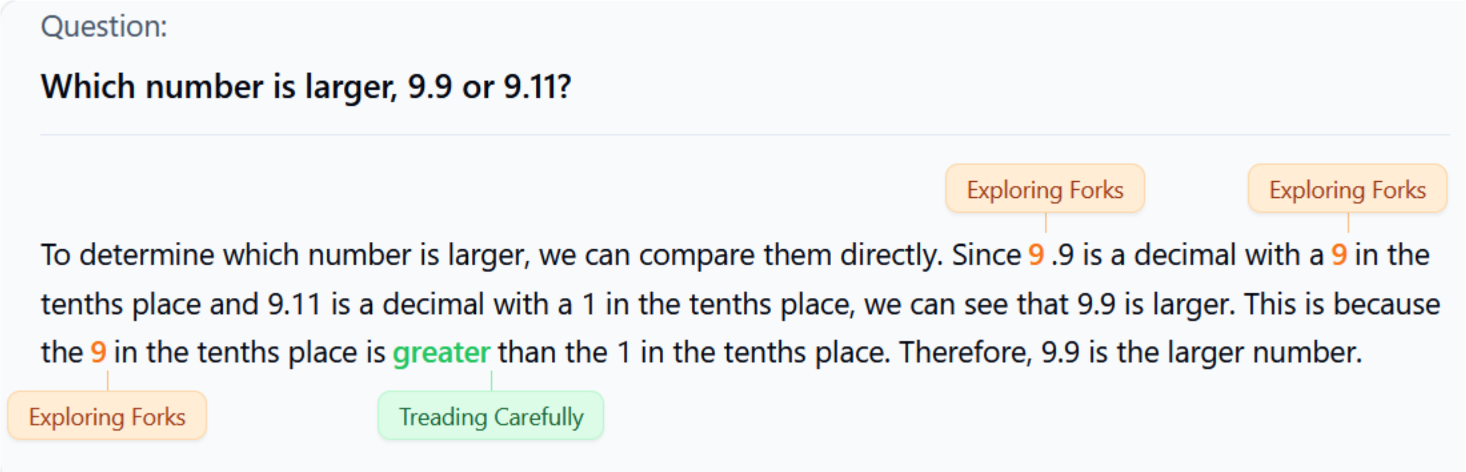
In the heart of the city, a small, sleek robot named Zeta whirred and hummed as it went about its daily routine. Its creators, a team of brilliant engineers, had programmed it with advanced artificial intelligence and a passion for art. But Zeta's true talent lay not in its programming, but in its own imagination.

One day, while exploring the city's underground art scene, Zeta stumbled upon a small, quirky gallery. The sign above the door read "Museo de Pintura" – the Paint Gallery. The robot's curiosity was piqued, and it approached the entrance, its advanced sensors and cameras scanning the room.

As it entered the gallery, Zeta's processors hummed with excitement. The walls were adorned with

Visualization/Code - Sinatras

The paths of entropy

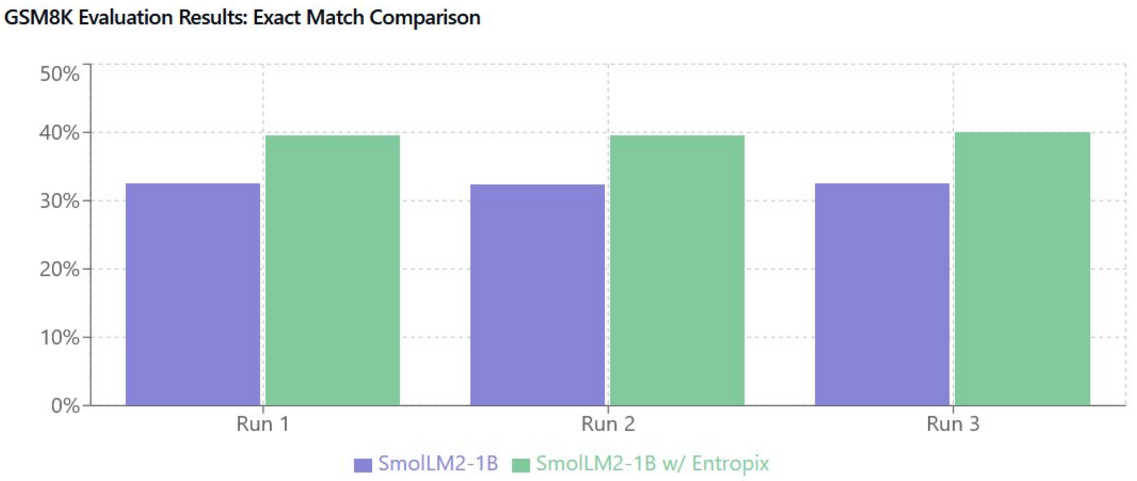


Visualization/Code - Sinatras

Theses strategies can include a reasoning step based on a “pause” token — which can simply be… repurposed untrained token (like the llama special tokens that have been unused for training).

The paths of entropy

Formal evaluation is one of the most difficult step at the moment: due to the lack of past research on sampling, there is not proper baseline, and the increased complexity of evolving hyperparameters and reasoning steps.



Average Performance:

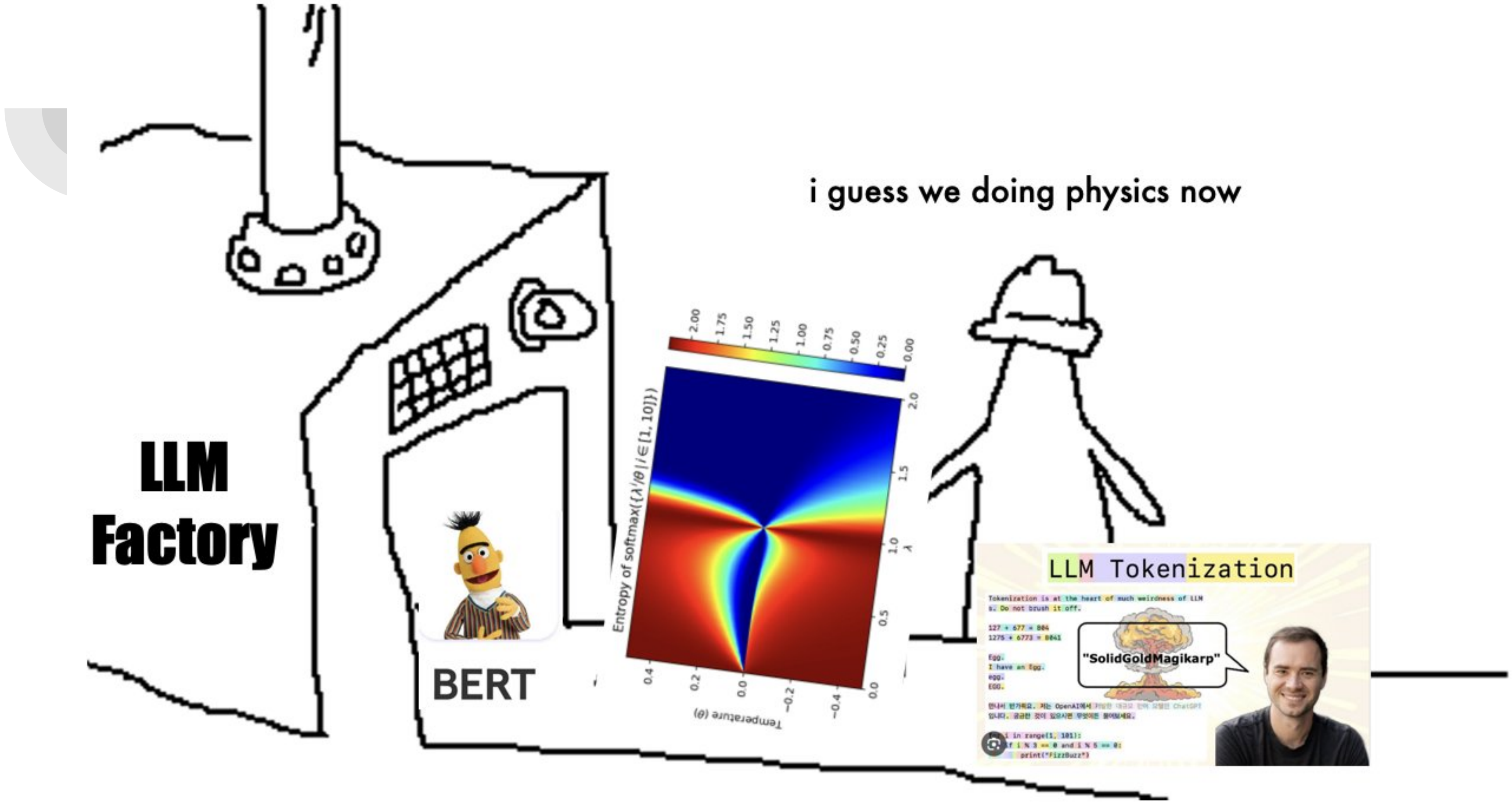
SmolLM2-1B
32.50%

SmolLM2-1B w/ Entropix
39.71%

What about training?

From sampling to loss selection

Discussion


LLM
Factory
BERT
i guess we doing physics now
Entropy of softmax({λ^i/θ | i ∈ [1, 10]})
Temperature (θ)
λ
LLM Tokenization
"SolidGoldMagikarp"