Sampling and Inference

EN.705.743: ChatGPT from Scratch

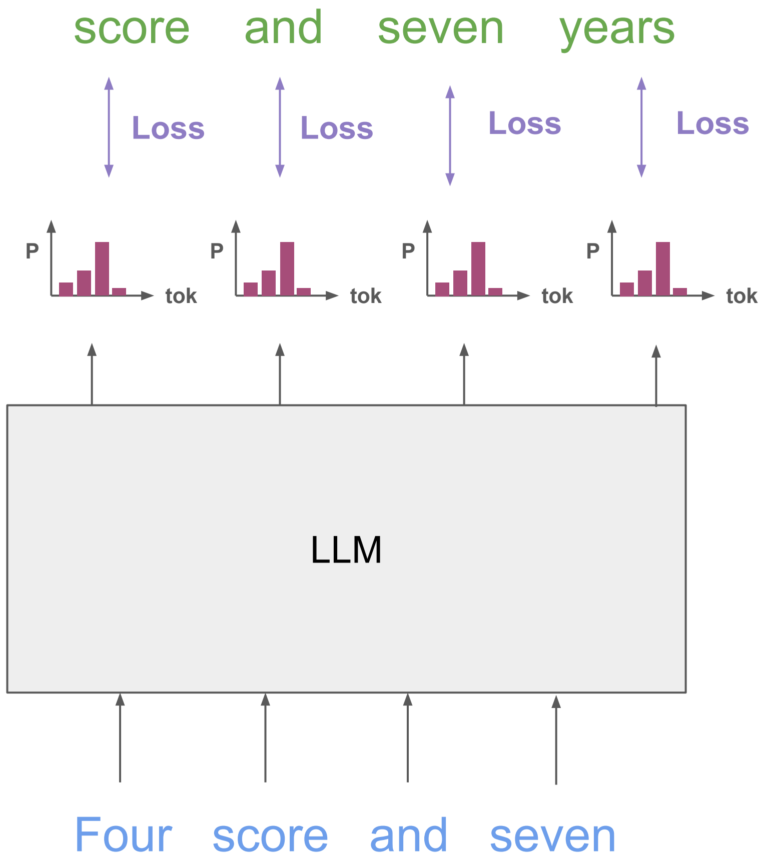
Recap: A Trained Model

Recap

Last week we discussed model training. An LLM is trained on many billions of tokens, often representing large swaths of the internet.

Recap

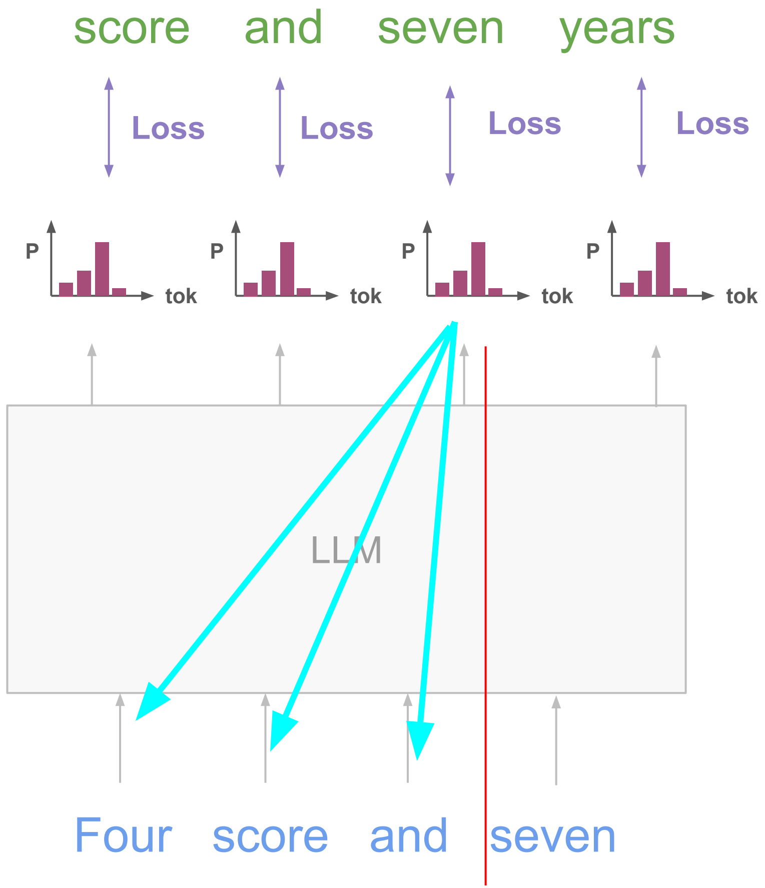
The LLM is trained by taking a sample of text and computing, at each position, a loss between the predicted probabilities over next tokens and the actual token from that sample (cross-entropy loss).



Recap

The LLM is trained by taking a sample of text and computing, at each position, a loss between the predicted probabilities over next tokens and the actual token from that sample (cross-entropy loss).

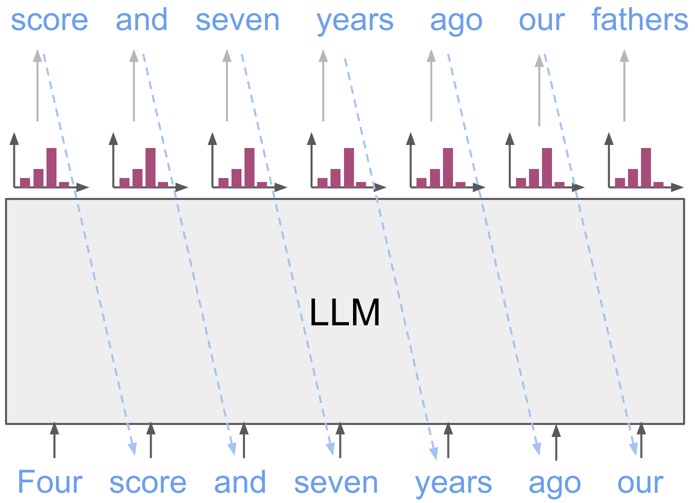
At position i, the output is only dependent on inputs [0 to i], because we use causal masking.



Recap

To generate text with our model, we can start with some initial text, and then sample one token at a time.

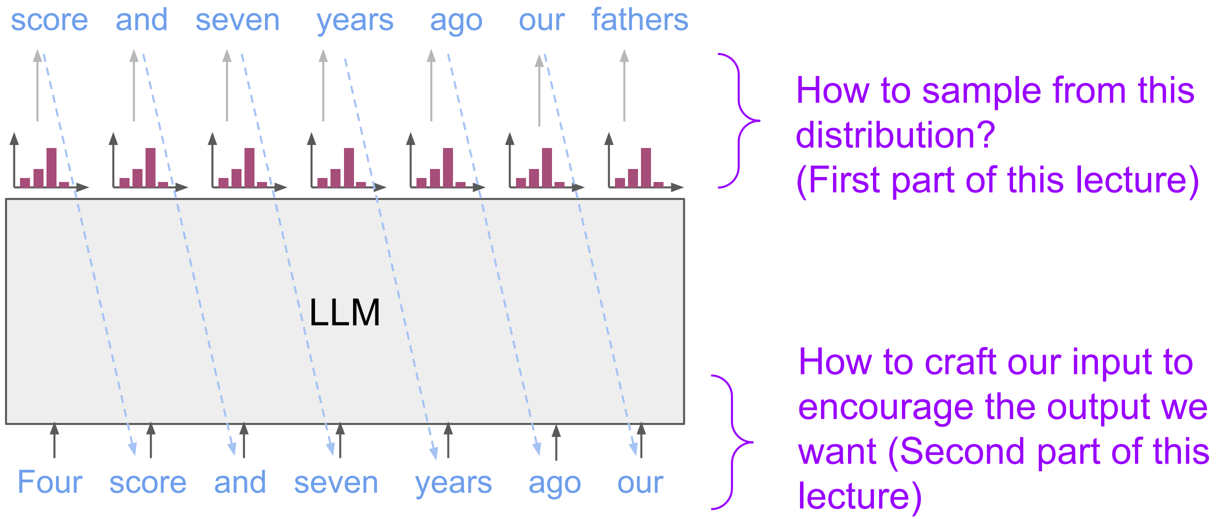
Each time we sample a token, we add it to the input and repeat (until we reach some desired length or an <|eos|> token).



Recap

To generate text with our model, we can start with some initial text, and then sample one token at a time.

Each time we sample a token, we add it to the input and repeat (until we reach some desired length or an <|eos|> token).

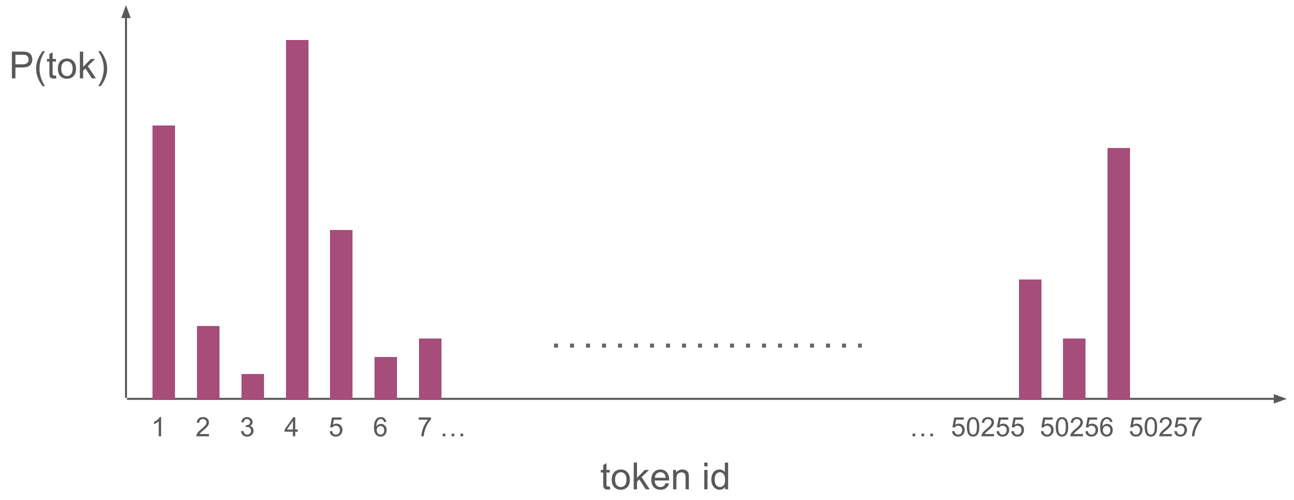

score
and
seven
years
ago
our
fathers
LLM
Four
score
and
seven
years
ago
our
How to sample from this distribution?
(First part of this lecture)
How to craft our input to encourage the output we want (Second part of this lecture)

Sampling

Thinking about the output distribution

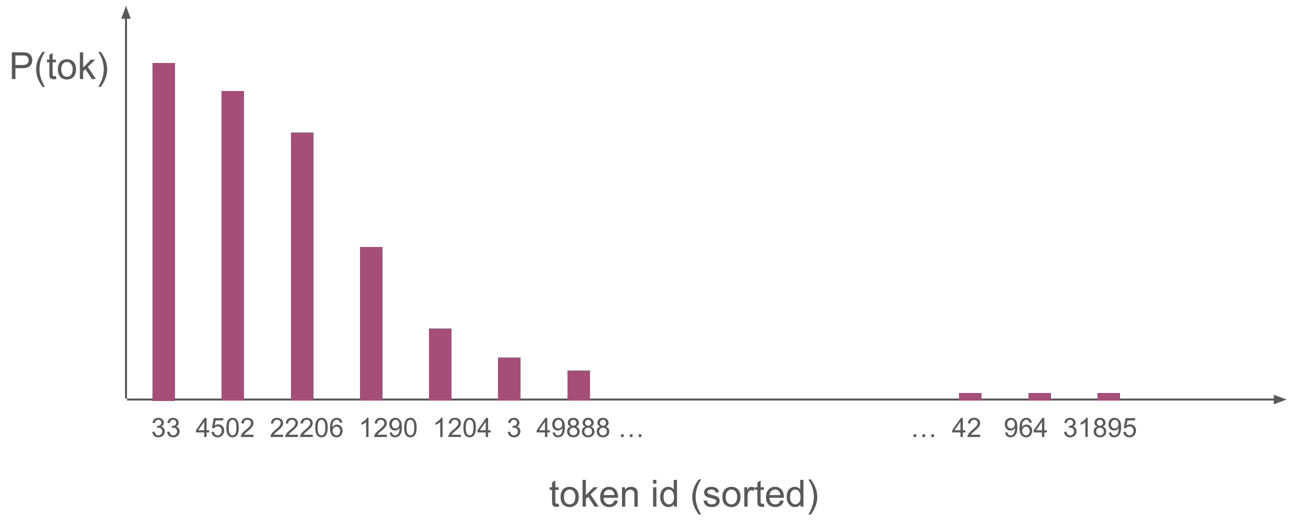
Each output is a probability distribution over our entire vocabulary (something like 50k entries).

This is very difficult to visualize or reason about:



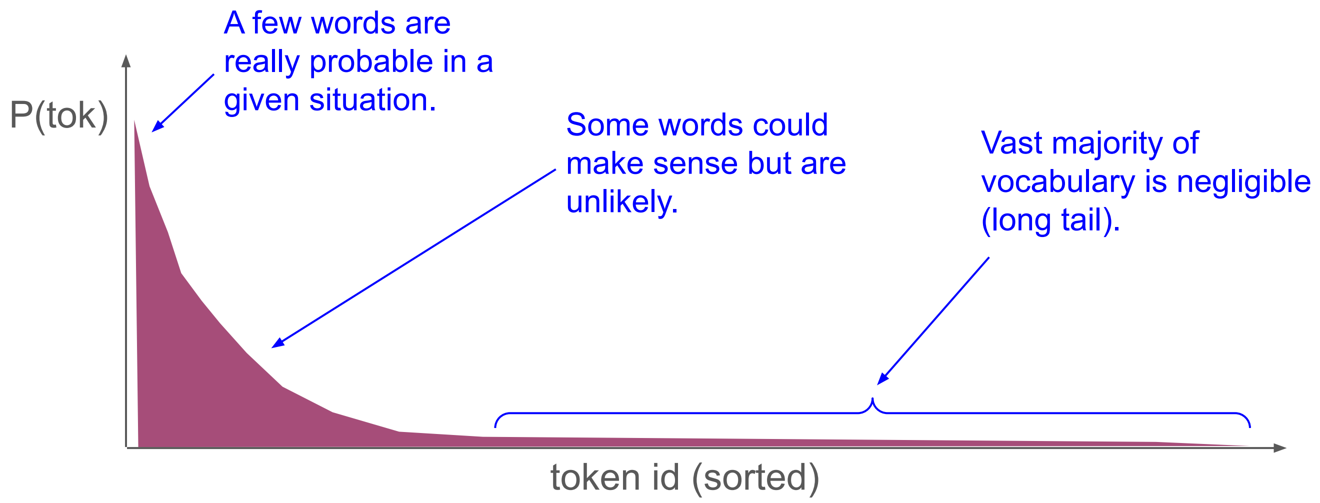
Sorting the distribution

So, we sort this distribution by decreasing probability:



Sorting the distribution

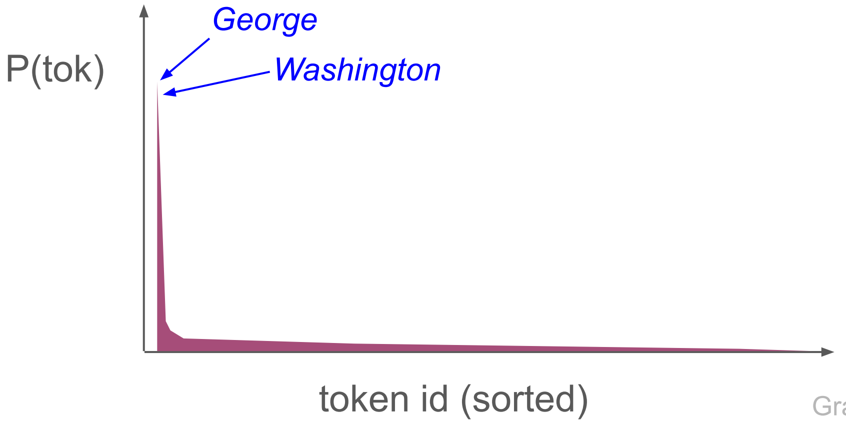
If we zoom out and actually show all ~50k entries, we would see a shape like this:



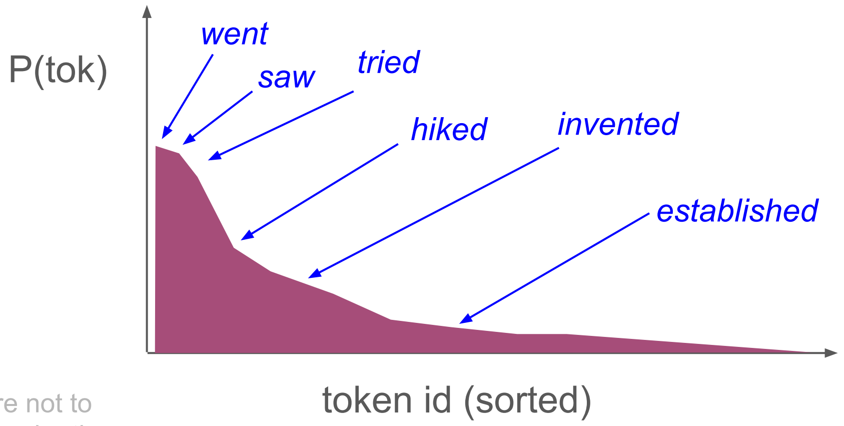
Distribution is a function of the input

The shape of the output distribution is reflective of how many words "fit" when it comes to continuing the input text.

The first president of the U.S. was …



Over the weekend I …



Graphs are not to scale w.r.t. each other

Temperature

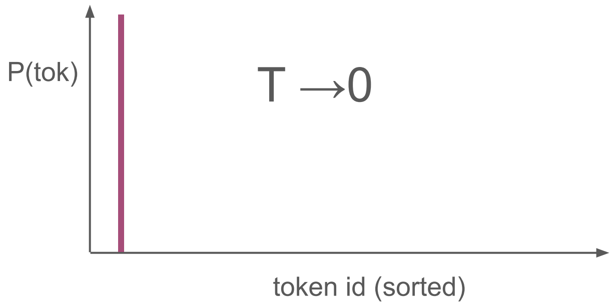
Whenever a softmax is used, you can introduce a temperature to adjust it.

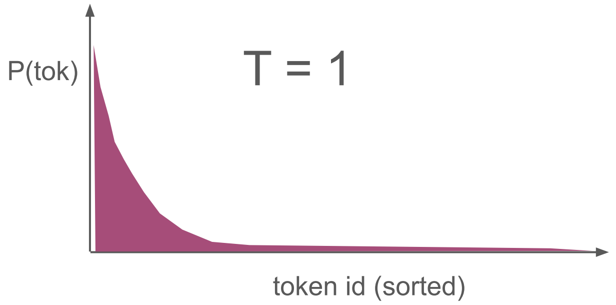
$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

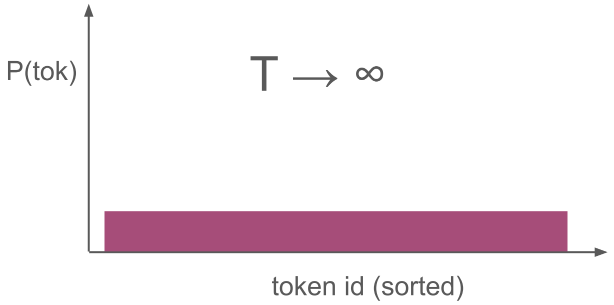
Temperature

Whenever a softmax is used, you can introduce a temperature to adjust it.

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$



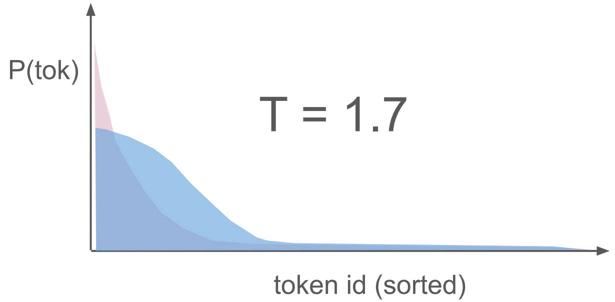


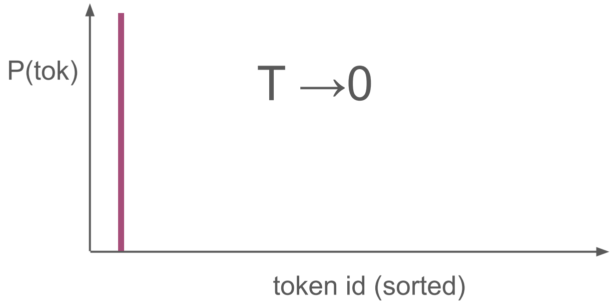


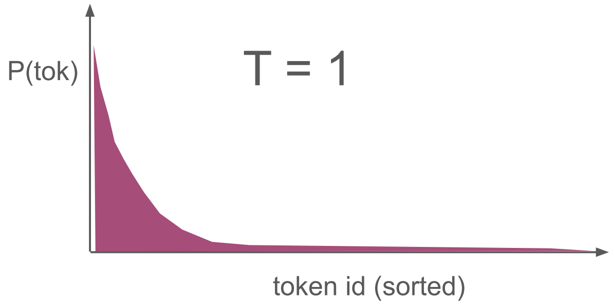
Temperature

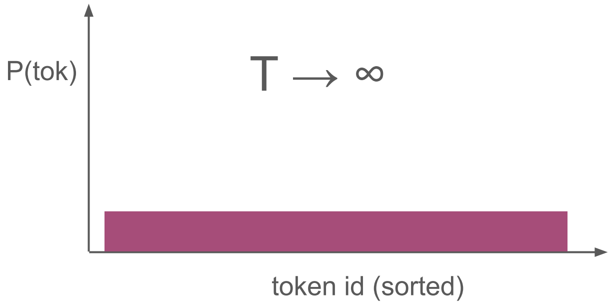
Whenever a softmax is used, you can introduce a temperature to adjust it.

Something greater than 1 will smooth out the beginning of this curve, giving more weight to words that are possible but not likely. This can lead to outputs that seem more "creative".









Problems with naive sampling (Top Choice)

If we always sample the most probable word (entry #1 in our sorted list of tokens), we have a few problems:

First of all, the text is not very exciting (deterministic).

More importantly, text is highly repetitive (detrimentally so).

Problems with naive sampling (Top Choice)

If we always sample the most probable word (entry #1 in our sorted list of tokens), we have a few problems:

First of all, the text is not very exciting (deterministic).

More importantly, text is highly repetitive (detrimentally so).

Example Completion (green):
Selecting top word:

Yesterday I went to the gym and I was doing my workout and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press

Problems with naive sampling (Top Choice)

LLMs output is based on the entire input, and once a word appears it is more likely that it will occur again (i.e. it is the topic of conversation).

This can cause feedback loops in which the model creates strong patterns that it then propagates over and over.

Example Completion (green):
Selecting top word:

Yesterday I went to the gym and I was doing my workout **and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press**

Patterns

Pretrained LLMs **love** patterns. The training objective is to predict the next token based on all prior tokens, which means there are two sources of information for prediction:

1) The model's internal parameters
2) The previous tokens

(2) can overrule (1) in many cases if strong patterns exist.

Patterns

The capital of the United States is *Washington.*

The model needs to know the capital of the U.S. (mainly internal weights)

Red, blue, red, blue, red, blue, red, *blue.*

The model is simply continuing the input pattern (provided context)

Raspberry, blackberry, blueberry, *strawberry.*

A bit of both. The input has a pattern, but the model also needs to know fruit.

Most examples will be in the last category- the model uses knowledge and the context to provide an output distribution.

Patterns

Keep in mind that the model is just trying to continue text.

Continuing a pattern is a perfectly valid way to do this.

Consider example on right: do we expect an answer or more questions? Both continue the input text.

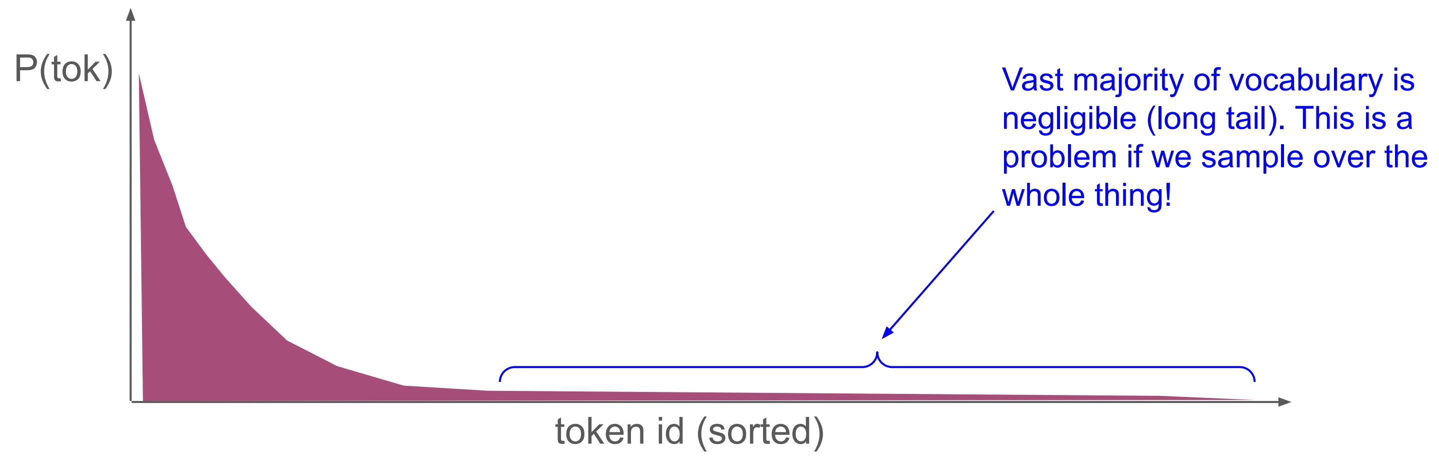
What is the capital of France? How many people live there?

GPT3 completion (Babbage):
Who is the president of France? Is Paris safe to visit? What is the capital of Russia? Which country has the largest army? Which countries have the largest population? What is the language of Russia? How many inhabitants are there in Europe?

Problems with naive sampling (Full Distribution)

If we sample from the whole distribution, we have a high probability of selecting nonsense. While each low-probability word is unlikely individually, there are so many of them that there is still a chance of selecting something silly.

[Very low probability] X [45,000 choices] = non-negligible probability!



Problems with naive sampling (Full Distribution)

If we sample from the whole distribution, we have a high probability of selecting nonsense. While each low-probability word is unlikely individually, there are so many of them that there is still a chance of selecting something silly.

Example Completion (green):
Sampling the whole distribution:

Yesterday I went to LLBean to pick up some key-rings. Guess who I found in my "hope chest". Woman in White!! In her Believe me! It was a while back so remember to visit Lakeland a few days early. LOLW

back to fishing/love thing >

From ken40946:

Yep, I voted

Problems with naive sampling (Full Distribution)

Example Completion (green):
Sampling the whole distribution:

Once upon a time, there was a young boy who wandered into the forest. He was lost and aimless. One day he met and fell in love with a lovely girl. They danced in the forest all night and two years later, they were married. They welcomed their first child, a boy named Alan. Alan growss up and remembers his mother heavily. He is Mainstream and loves anime. Alan teaches Kun. And that is how Mai

In this example, the LLM makes sense from word to word but the topic “drifts”. Every time an unlikely word is sampled, this pushes us farther away from the original topic.

Problems with naive sampling (Full Distribution)

Example Completion (green):
Sampling the whole distribution:

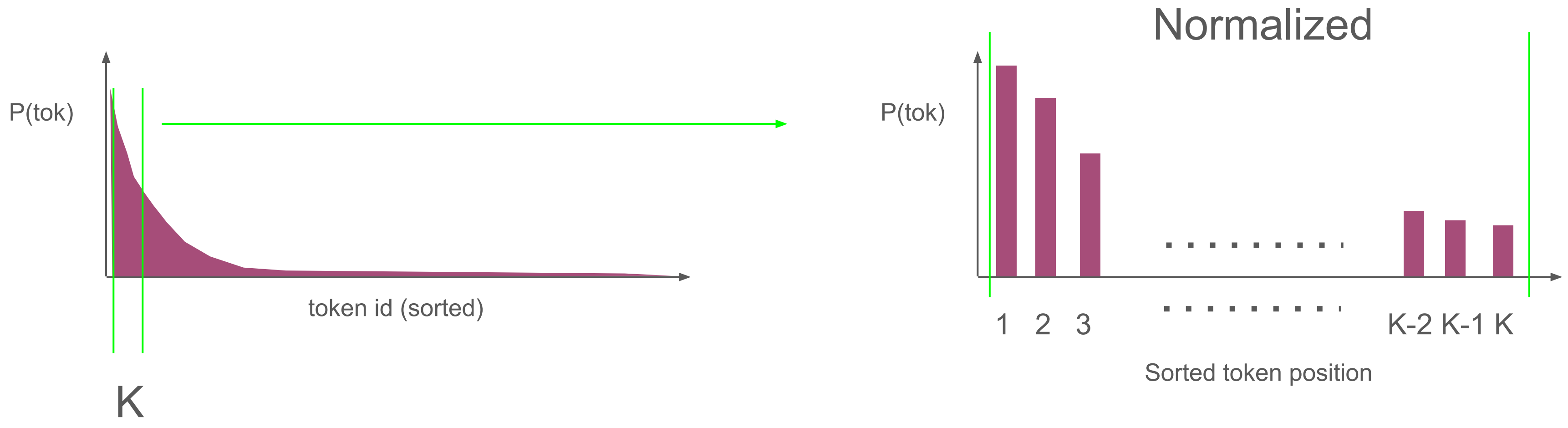
Once upon a time, there was a young boy who wandered into the forest. He was…
Girl, Body, Heart
Spirituality
Things
Turn a Child into An Adult
Clutter Busting
If you love this article you'll also love… Your Self Esteem Declines with every move, change, and imperfection. It's your…
Getting Your Surrender Processed It's the first weekend of November and while people are getting…
Family Business! Congrats Brother But you're not done yet… Once upon a time there were…

In this example, the LLM samples some unlikely words early and quickly degrades into nonsense.

How do we fix this? (Top-K)

The easiest remedy is to only select among the K most probable tokens. This is called “top-k” sampling. (Typically, K=30, 40, 50, something like that).

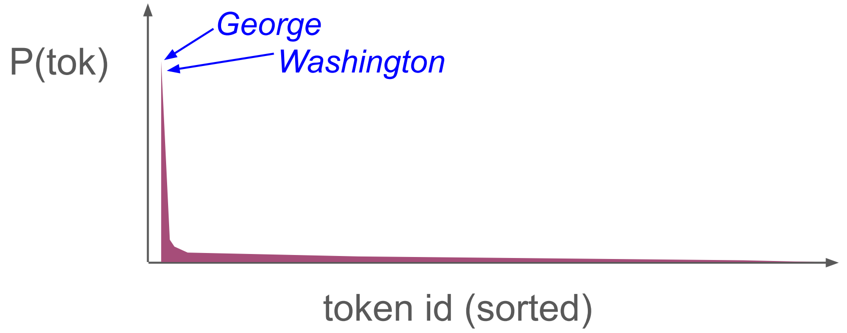
The first K tokens have their probabilities normalized to one and we sample from this new distribution:



Top-P

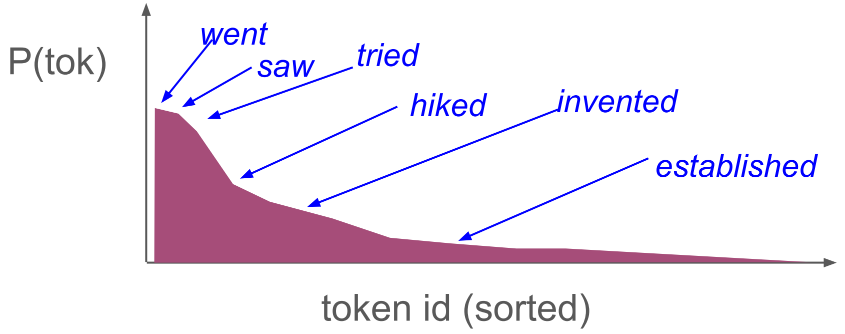
Top-K is simple and can work pretty well, but it does not account for the shape of the distribution.

The first president of the U.S. was …



In this example, a few tokens dominate, and we want to only sample from these. A typical value of K=30 might be too big!

Over the weekend I …

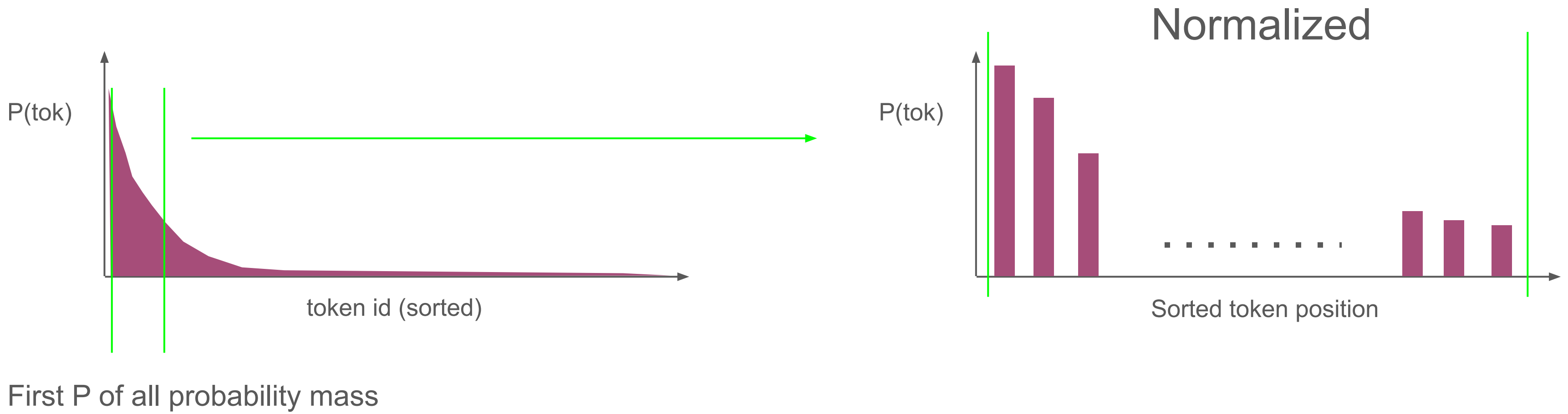


In this example, there are lots of possible options. A value of K=30 would be too restrictive!

Top-P

Top-P sampling samples from the first N tokens which sum to probability mass at least P (always at least one token!). Typical values of P might be 0.8, 0.9, 0.95.

If only 3 tokens take up 90% of all probability, then we only consider those 3 tokens! However, if 90% is spread over 100 tokens, we will consider all of those.



Problems with Repetition

Top-K or Top-P sampling fix our balance issue- we can consider multiple good choices while ignoring the “long tail” of our distribution.

However, they do not completely fix some problems of the model latching on to patterns and creating feedback loops.

Example Completion (green):
Selecting top word:

Yesterday I went to the gym and I was doing my workout and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press

The problem here is not from the tail- it is that only a few repeated tokens dominate the distribution. Top-P or Top-K would still include these tokens.

Breaking Repetition

There are some common ways we can encourage the model to not repeat itself:

Frequency Penalty - the probability of a token is lowered proportionately to how many times it has occurred so far.

Presence Penalty - the probability of a token is lowered if it is present at all in the previous text.

More generally, these are types of **Repetition Penalties**, which lower the probability of a token (or sequence of tokens) based on previous occurrence.

Breaking Repetition

Repetition Penalty: lowers the probability of a token (or sequence of tokens) based on previous occurrence.

Modify the temperature on a per-token basis by adding a multiplier based on past text.

Important Note: These should be used lightly! They can have adverse effects against common words and punctuation.

Other Important Note: To use both of these penalties together, add the incremental values (i.e. 0.1 and 0.2) to a starting value of k=1. See next slide.

$$softmax(x_i) = \frac{exp(x_i/Tk_i)}{\sum_{j=0}^{N} exp(x_j/Tk_j)}$$

k is initially 1.0 (no change).

Frequency of 1.1:
k += (0.1*number of occurrences)

Presence of 1.2:
k += (0.2 if token has occurred)

Sometimes these would be defined as “0.1” and “0.2”- it depends on your definition. Above, we define them as [1+increment] and accrue an incremental penalty of [penalty_setting - 1.0].

Example:

Vocabulary: [a, b, c, d], Frequency Penalty: 1.1, Presence Penalty: 1.2, Current sequence: a,b,a,c,a,a
Raw logits from model: [0.8, 0.9, 0.1, 0.4] (if these included negative values, we would subtract the min, next slide)
Without penalties, our distribution would be: [0.305, 0.338, 0.152, 0.205]

First we initialize a value of k for each of our vocabulary entries: k=[1, 1, 1, 1]

Then we apply the frequency penalties: a has occurred 4 times, b once, and c once: Occurrences: [4, 1, 1, 0]
Our accrued penalties are [occurrences]*[frequency_penalty - 1.0] = [0.4, 0.1, 0.1, 0.0]

Then we find our presence penalties. We accrue a penalty of [presence_penalty - 1.0] for each token that has occurred at all: [0.2, 0.2, 0.2, 0.0]

Our final k values are: [1,1,1,1] + [0.4,0.1,0.1,0.0] + [0.2,0.2,0.2,0.0] = [1.6, 1.3, 1.3, 1.0]

We update our logits with these values: new_logits = [0.8, 0.9, 0.1, 0.4] / [1.6, 1.3, 1.3, 1.0] = [0.5, 0.69, 0.08, 0.4]

After softmax, we now get: [0.265, 0.320, 0.174, 0.240]. Note “d” has become more probable.

“c” also becomes more probable because all the values need to add to 1.0 at the end. Although we penalized “c”, we also penalized “a” and “b”, and “c” ended up gaining more than it lost.

Important Note

The penalties only works if the logits are positive! Move the logits to a positive range before doing this (i.e. by subtracting the minimum value).

Explanation: Dividing by a larger temperature (i.e. a penalty) moves the logit closer to zero. If some logits are negative, this will actually make that entry more probable relative to the other negative logits.

Breaking Repetition

Generating with both of these added we can break out of repetitive patterns. Here the “leg press” pattern is introduced on purpose, and we can see the model gets out of the feedback loop:

Example Completion (green):

Yesterday I went to the gym and I was doing my workout and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was doing the leg press and I was about to finish my workout, my leg pump. And I heard somebody say something. It caught me off guard because it's like someone walked in. So all of a sudden this guy comes up to me while I'm just sitting there going through a workout trying to enjoy myself at the gym…

A Complete Sampling Algorithm

Typically use Top-P of 0.8 or 0.9, generally preferred to Top-K.

Play with temperature if you want to, but not needed.

Add minor frequency and presence penalties.

Adjust based on observed behavior.

Example Completion (green):
top-p=0.9, frequency and presence of 1.2

Yesterday I went to Kavli Institute for Particle Astrophysics and Cosmology, where the astronomers set up a star cluster and took pictures. They were very lovely people who would explain things that I did not understand but to me it was very interesting.

Summary

Naive Sampling (Top 1)

Prone to repetition or obvious/boring text.

Naive Sampling (Full Dist)

Quickly derails by sampling from the tail.

Top-K

Better, but does not account for distribution shape. Still prone to repetition.

Top-P

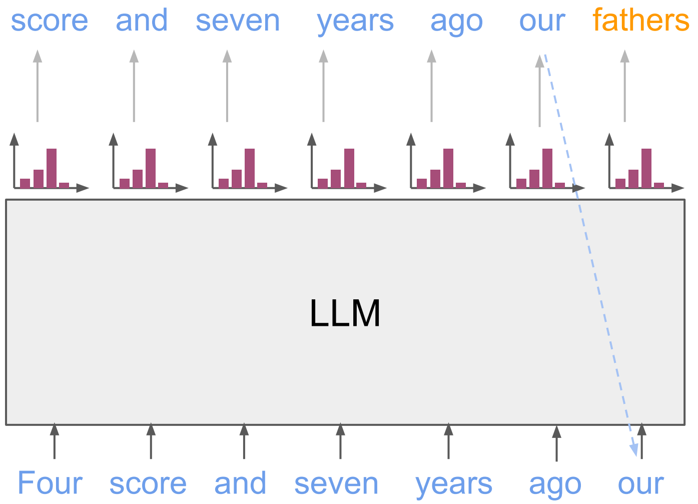
Better, but still prone to repetition.

Softmax Alterations:

- Temperature
  - Sharpen or flatten the entire distribution.
- Frequency Penalty
  - Make tokens less likely proportional to how often they have occurred before.
- Presence Penalty
  - Moke tokens less likely if they have already occurred at all.

Brief Aside on KV Caching

Efficient Generation

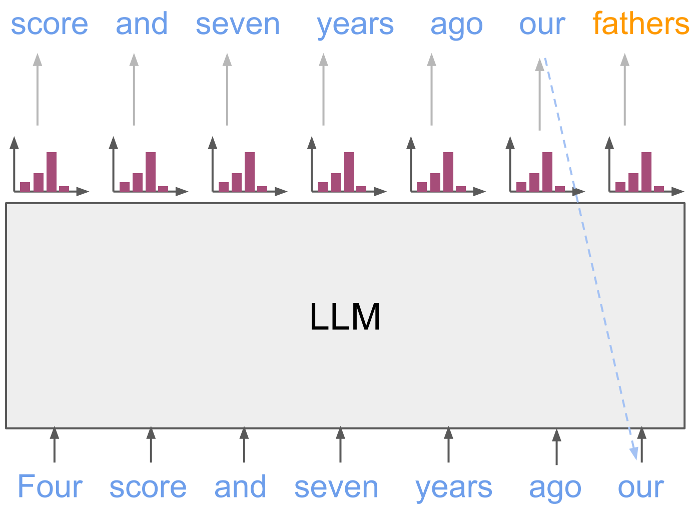


Consider generating the word **“fathers”.** This token is based on all previous inputs.

If we feed “*Four score and seven years ago our*” into the model, we will actually get an output for every input token!

All the outputs except for “fathers” are redundant computation.

Efficient Generation



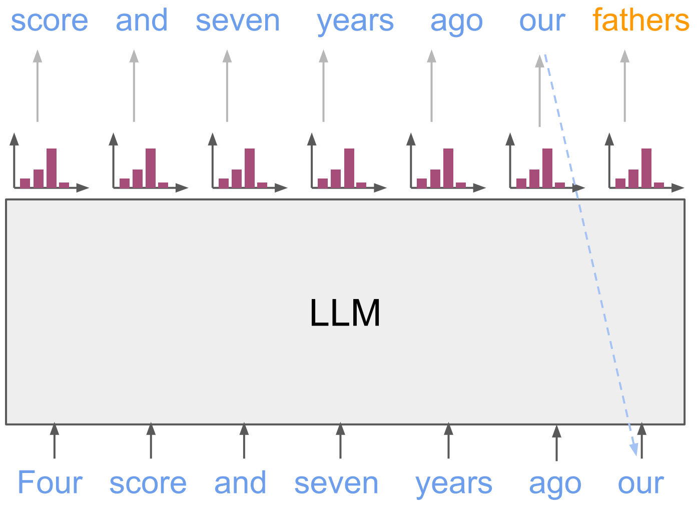
If we look at the attention equation (simplified here):

$$y_i = \frac{1}{i} \sum_{j=0}^{i} (q_i \bullet k_j) * v_j$$

Within each transformer layer, the output at position *i* depends on the *ith* query and the keys and values of all previous tokens (and the current):

y_i depends on q_i, $k_{<=i}$, $v_{<=i}$

Efficient Generation



When we generate “fathers”, we should already know the keys and values from all previous tokens.

$$y_i = \frac{1}{i}\sum_{j=0}^{i}(q_i \bullet k_j) * v_j$$

New for position i, not needed for other outputs.

New for position i, **known for previous positions**

New for position i, **known for previous positions**

KV-Caching

Store a history of keys K and values V.

In all attention layers, when generating the next token:

- Retrieve K and V for all previous tokens.
- Compute Q, K, and V <u>only for the current token.</u>
- Add K and V for the current token to our history

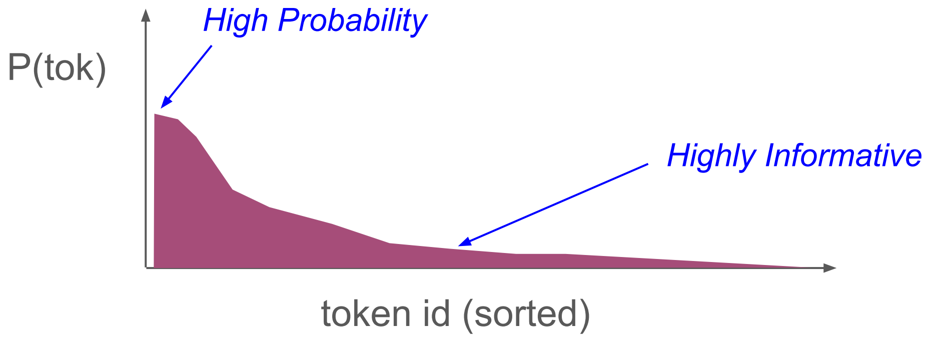
Another Aside on Human Word Sampling

Skipped in Lecture

How do humans sample their words?

Humans write or speak to convey information.

The information of a token is inversely proportional to how probably it is (from probability theory).



Skipped in Lecture

How do humans sample their words?

Humans write or speak to convey information.

The information of a token is inversely proportional to how probably it is (from probability theory).

The zebra’s stripes were black and ____

Skipped in Lecture

How do humans sample their words?

Humans write or speak to convey information.

The information of a token is inversely proportional to how probably it is (from probability theory).

The zebra's stripes were black and white

Not very informative, but probably what we expect.

Skipped in Lecture

How do humans sample their words?

Humans write or speak to convey information.

The information of a token is inversely proportional to how probably it is (from probability theory).

The zebra’s stripes were black and green

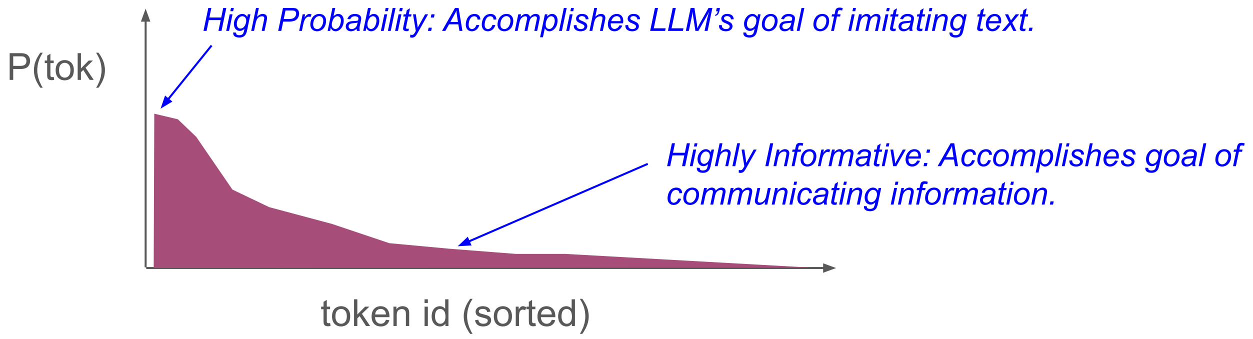
Pretty informative! That is a really weird zebra.
(And therefore, probably worth communicating about)

Skipped in Lecture

How do humans sample their words?

Humans write or speak to convey information.

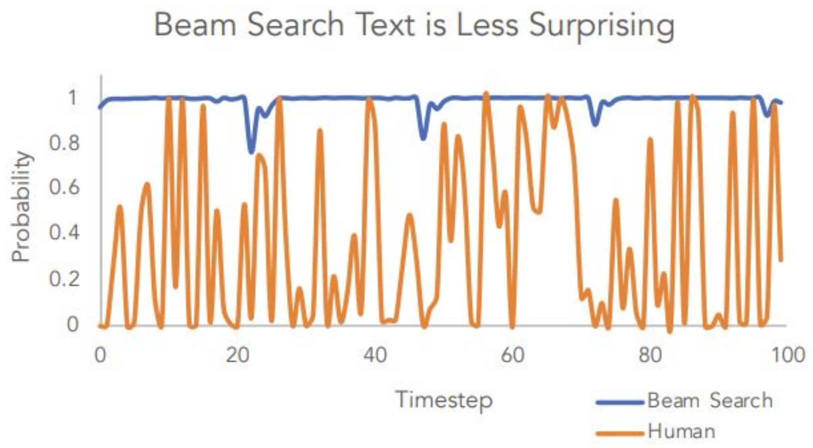
Because we are trying to communicate (convey information), humans often use words that are unlikely according to a base language model.



Skipped in Lecture

Detecting Human (and Non-Human) Text

Text that is human-authored will have regular fluctuations in probability that is not seen in LLM-generated text. This can be used to detect if a piece of text (perhaps secretly) written by an AI.



From *The Curious Case of Neural Text Degeneration.*

Skipped in Lecture

Prompting

Prompting

We now know how to generate decent quality text from our model (sampling).

We know how to do so efficiently, if we need to (KV-cache).

However, the model is still generating random continuations- how do we get it to output what we want - something useful?

Prompting

Remember there are two sources of information for our model:

1) The stored knowledge in the weights
2) Context and patterns in the input

The model loves to pick up on input patterns, and sometimes this is really bad (i.e. feedback loop, repeats the same words over and over again).

However, we can also use this to our advantage.

Prompting

Prompting is the art of introducing context, knowledge, or patterns into the input in order to steer the output of the model.

The easiest way to do this is typically a pattern- craft some text pattern that you want the model to follow. Obviously we can craft some useless patterns:

Input: “red, blue, red, blue, red, blue, red” - Model will output: blue, red, …

Input: “Monday, Tuesday, Wednesday,” - Model will output: Thursday, Friday, …

Prompting

We can also make patterns that demonstrate a behavior:

Input:

“I loved the movie : Positive,

It was way too long : Negative,

The dialogue was fine, but the story didn’t make sense. : Negative,

The leading actress did a wonderful job. : ”

How would you continue this text?

Prompting

We can also make patterns that demonstrate a behavior:

Input:

“I loved the movie : Positive,

It was way too long : Negative,

The dialogue was fine, but the story didn’t make sense. : Negative,

The leading actress did a wonderful job. :

Positive”

By showing examples of sentiment classification, we have turned our model into a sentiment classifier!

We typically show a few examples of the desired behavior, and then end with the example we want the model to process.

Prompting

Another Example:

I love the beach / Ich liebe den Strand

What time is it? / Wie spät ist es?

The beer is ten dollars. / Das Bier kostet zehn Dollar.

Translation examples. Clear English / German pattern.

Where can I get a coffee? /

Incomplete pattern continuation for what we want to translate.

Prompting

Another Example:

I love the beach / Ich liebe den Strand

What time is it? / Wie spät ist es?

The beer is ten dollars. / Das Bier kostet zehn Dollar.

Where can I get a coffee? / Wo kann ich einen Kaffee bekommen?

By continuing the text, the model serves as a translator.

Use your imagination!

This can be very powerful if you brainstorm a bit. Some more interesting examples:

```
# get all files in the ./data/ directory
files = [f for f in os.listdir("./data") if isfile(join("./data", f))]

# Make a 5 by 3 numpy array of zeros
np.zeros((5,3))

# import opencv into python
import cv2

# sort each sublist in the list
y = [x.sorted() for x in y]

# calculate the cumulative sum over my list
sum = 0
for k in y:
    sum += x[k]
```

I see a green car going southbound on I-95.
{vehicle: car, color:green, heading:south}

There is a truck speeding on 295 west.
{vehicle: truck, color: unknown, heading: west}

A pink sedan is driving aggressively.
{vehicle: sedan , color: pink, heading: unknown}

I'm in a red pickup headed down to Florida.
{vehicle: pickup, color:red, heading:south}

This one took some fiddling to get working. Since there is only one “correct” answer here, I tweaked the sampling to be less random.

Prompting vs Instruction Tuning

Modern LLMs are often trained to follow instructions instead of simply continuing an input pattern.

This amounts to fine-tuning with (instruction, expected reply) pairs.

What is the capital of France? How many people live there?

Who is the president of France? Is Paris safe to visit?...

vs:

The capital of France is Paris, which has a population of 2.2 million people.

Sampling from Our Homegrown Models

If you happen to run a sampler on the model from our assignments, you get things like this (initial input in red, continuation in blue):

Thomas Jefferson was the command in two First World War. In which event had become a senator, and many were particularly good friends as president. Both of Manipiros began to continue work on how American agents go on regular debateders they could use Malcolm Belgradous about his life rights towards her mother's best belonging to the former officers too poor. Luke <unk> acted into one " live salary camp, with off way little insurance from each other. The

The city of Chicago, created a book called " wicky house [sic] by detectively echair over... or hero ”. The poem is now published in <unk> referring to the church as titled first directed by Yia W 'Melebrity Mana and Space Shannada Film Music Society. The release are " Oss Reed Beech on radio at their Halloween Reculverment Program collection that assignments " by censored figures " (including

Sampling from Our Homegrown Models

This happens because our models are tiny. They *almost* produce English.

What they are missing is scale. For reference, I trained a model that is 30x larger and for 10x longer (just repeated the data 10 times, not a good practice). The underlying code is otherwise identical.

Thomas Jefferson was ultimately elected as the first Democrat to the Virginia House of Representatives, having raised a large stockpile of common land in his district. Williams enjoyed strenuous adventures during these years, becoming known as one of their " master masonesses's manuals " by practicing with them.<|endoftext|>

Thomas Jefferson was elected as the second Confederate representative in June 1864. During this time, Washington and John Pope Cass called for a meeting with Oliver Ellsworth of the Confederate States Secretary Robert C. Gates.<|endoftext|>

Sampling from Our Homegrown Models

This happens because our models are tiny. They almost produce English.

What they are missing is scale. For reference, I trained a model that is 30x larger and for 10x longer. The underlying code is otherwise identical.

The city of Chicago purchased the land, draining (or completely) his lands to Charles Hamilton. Land was taken up again between two British immigrant families who claimed Indiana as part of their treaty. After Adams'death in 1783, President James Monroe called it " one @-@ half men growing together at a distance and fair? " The city, like Vincennes, had lost control over many neighboring areas such as North Bend and Siouxsvania parishes before they were settled by non @-@ natives near them ; New York was included

The city of Chicago experienced a 99 @-@ year hiatus due to low attendance at events in the early 21st century, including the Augustinian First Lady's Day Parade. Average daily ridership reached record highs on October 11 and December 9 with 88 @,@ 151 visitors when it replaced Lehi En Lawn by placing the statue below the Soldiers House Museum Building where it is being walked over town from Memorial Park. The fountain quickly crossed Pershing Square as soon as three oars made light contact with America before finally hitting the

Sampling from Our Homegrown Models

This happens because our models are tiny. They almost produce English.

What they are missing is scale. For reference, I trained a model that is 30x larger and for 10x longer. The underlying code is otherwise identical.

The city of Chicago purchased the land, draining (or completely) his lands to Charles Hamilton. Land was taken up again between two British immigrant families who claimed Indiana as part of their treaty. After Adams'death in 1783, President James Monroe called it " one @-@ half men growing together at a distance and fair? " The city, like Vincennes, had lost control over many neighboring areas such as North Bend and Siouxsvania parishes before they were settled by non @-@ natives near them ; New York was included

It also learns a peculiarity of the Wikitext dataset: punctuation directly between characters is surrounded by @'s: non-natives becomes non @-@ natives, 3.141592 would be 3 @.@ 141592

Also, Vincennes is a real historic place in Indiana, founded in 1732.

Scale (Preview of next lecture)

Our models are ~35M parameters, and trained for maybe 100M tokens.

A state-of-the-art model is on the order of billions parameters, trained for trillions tokens. This represents:

- **3+ orders of magnitude increase in model size**
- **3+ orders of magnitude in terms of data**

Additional Readings

The Curious Case of Neural Text Degeneration: https://arxiv.org/pdf/1904.09751

Further Works that Use Sampling Statistics:

- GLTR: (site was down when I tried to find it)
- Detect-GPT (Using Sampling to Detect AI-Written Text): https://arxiv.org/pdf/2301.11305
- Watermarking (Using Sampling to Mark Text as AI-Authored): https://arxiv.org/pdf/2301.10226
- Related Video: https://www.youtube.com/watch?app=desktop&v=-vToUx5SDW4

Review Assignments