Decoding Strategies

Tanmoy Chakraborty
Associate Professor, IIT Delhi
https://tanmoychak.com/

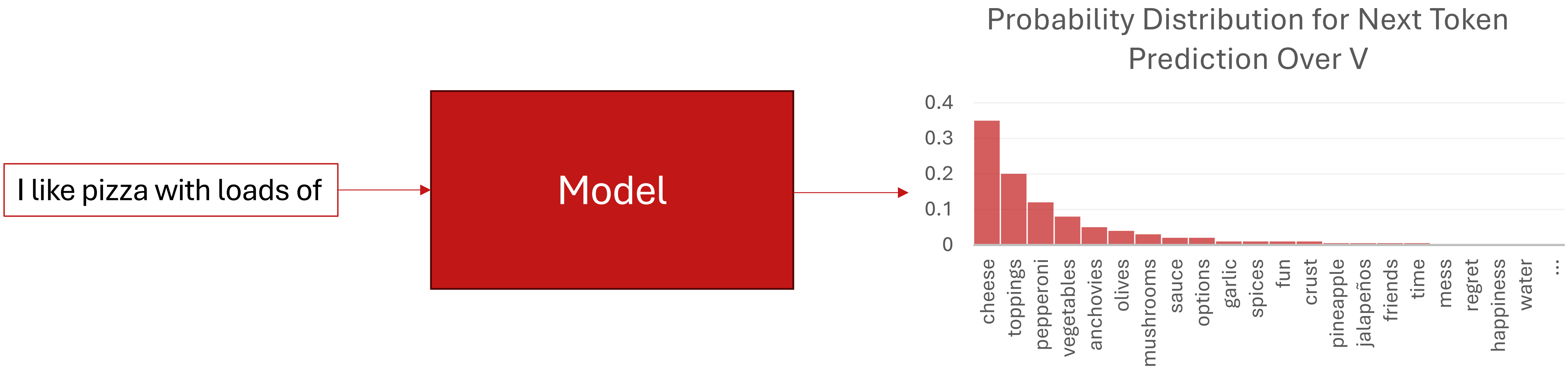
Introduction to Large Language Models

What is Decoding?

- Recall: In Seq2Seq models (like, NMT systems built with RNNs), we get a probability distribution over the tokens in vocabulary at each timestep.
    - We have $P(w_t | c, w_{<t})$ for $\forall\, w_t \in V$, where c is the input context/sentence/prompt, $w_{<t}$ are the tokens generated till now, and w_t is the token to be generated at the timestep t. V is the vocabulary.
- Decoding is the process of choosing the (next) token to generate based on the probability distribution over the vocabulary which the (language) model outputs.
- While generating the next token based on the output probability distribution, trade-off between two factors need to be handled by the decoding algorithms:

1. Quality (Generated responses should be coherent and accurate)
2. Diversity (Generated responses should be ‘creative’ and diverse, instead of being repetitive)

Recap



I like pizza with loads of

Model

Probability Distribution for Next Token Prediction Over V

Assume, we want to generate N subsequent tokens given the input context c (c is '*I like pizza with loads of*' in the above example).

In the previous lecture, we discussed:

1. Greedy Decoding (generate the token with the highest probability at each step)
2. Exhaustive Search Decoding (calculate the probabilities for all $|V|^N$ possible sequences; finally generate the most probable one)
3. Beam Search Decoding (retain only k most probable sequences at each step)

Deterministic vs. Stochastic Decoding Strategies

- The decoding methods which we discussed in the previous lecture – greedy decoding, exhaustive search and beam search – are all deterministic, i.e., *given the same model and the same input context, they always generate the same output sequence*.

- However, to ensure diversity of the generated responses we need stochastic decoding strategies.
  - The stochastic decoding strategies generate diverse responses even for the same model and same input context.

- In this lecture, we will look into three stochastic decoding strategies:
  1. Top-k Sampling
  2. Top-p / Nucleus Sampling
  3. Temperature Sampling

Random Sampling

- Before moving onto those three stochastic decoding strategies, let's first look at simple random sampling.
- If we want to generate a sequence of N words $w_1, \ldots, w_N$, then *random sampling* can be written formally as the following algorithm:

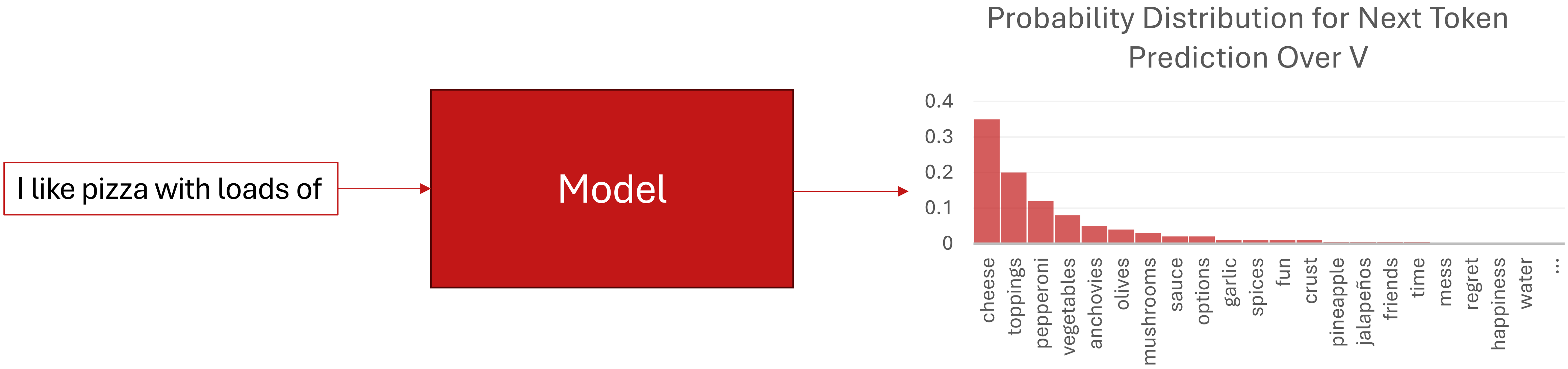
```
$i \leftarrow 1$
$w_i \sim P(w_i \mid c)$
while $w_i$ != EOS or $i<=N$:
        $i \leftarrow i + 1$
        $w_i \sim P(w_i \mid c, w_{<i})$
```

Intuitively, if say P*(cheese* | c='I like pizza with loads of'*)* = 0.35, then on giving c as input 100 times, on random sampling, 'cheese' is expected to be generated approximately 35 times as the next token, though the exact count may vary due to randomness.

Top-k Sampling

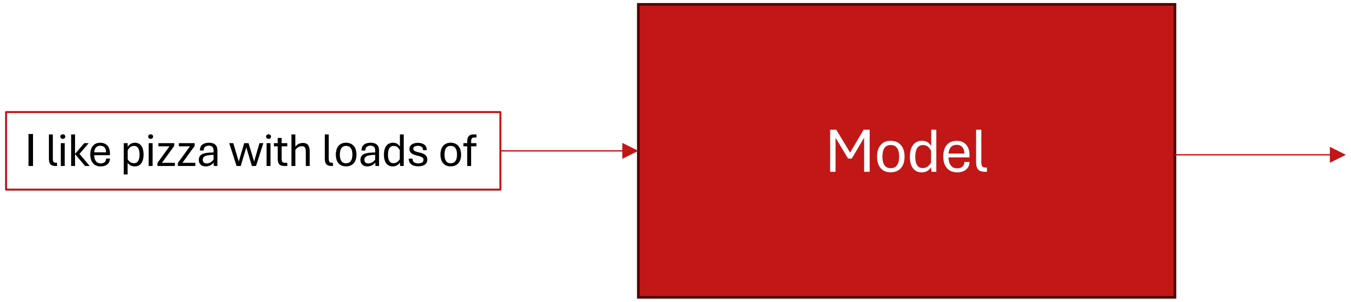
- In greedy decoding, the token with the highest probability was chosen at each step, making it deterministic. Top-k sampling is a stochastic generalization of greedy decoding.
- Top-k sampling involves the following steps:
  1. Choose in advance a number of tokens ***k***
  2. Given the probability $P(w_t | c, w_{<t})$ for all tokens in the vocabulary V, as generated by the model, sort the tokens by their likelihood, and throw away any token that is not one of the top ***k*** most probable tokens.
  3. Renormalize the scores of the ***k*** tokens to be a legitimate probability distribution.
  4. Randomly sample a token from within these remaining ***k*** most-probable tokens according to its probability.
- When k = 1, top-k sampling is identical to greedy decoding.

Top-k Sampling: Working Example

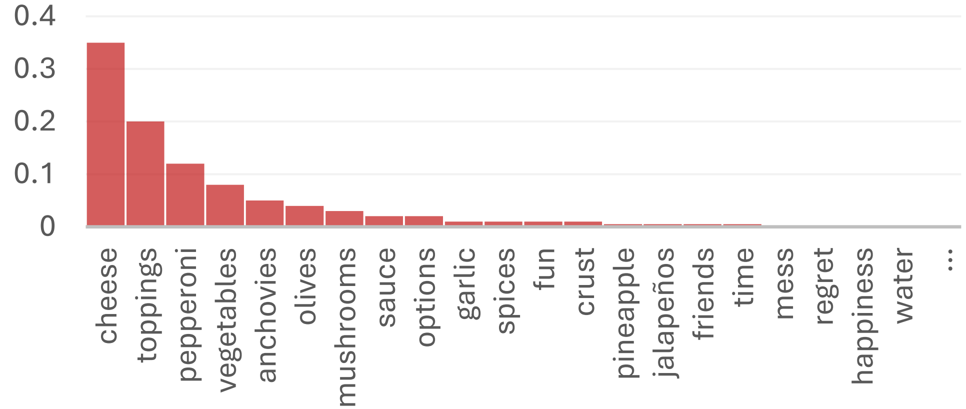


- Let's take **k=3**.
- Step-1: Sort the tokens by their likelihood, and throw away any token that is not one of the top ***k*** most probable tokens.
    - In our example, top-3 tokens with P(w|c) are: **cheese (0.35), toppings (0.20), pepperoni (0.12)**

Top-k Sampling: Working Example

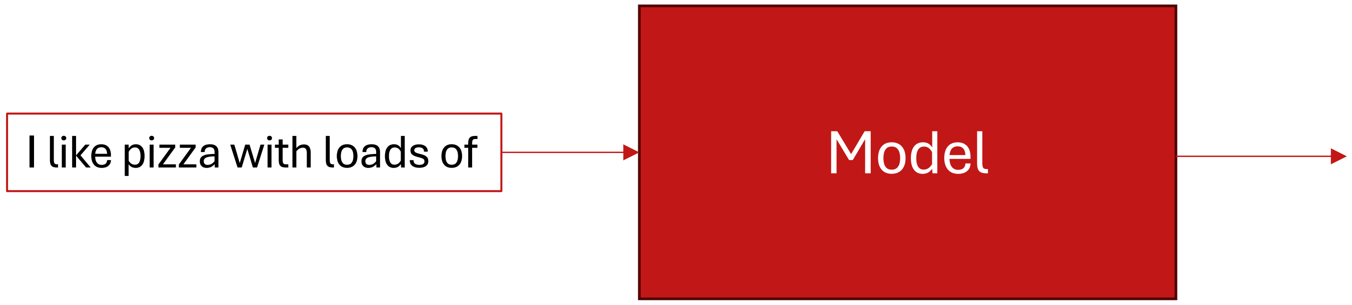

I like pizza with loads of

Model

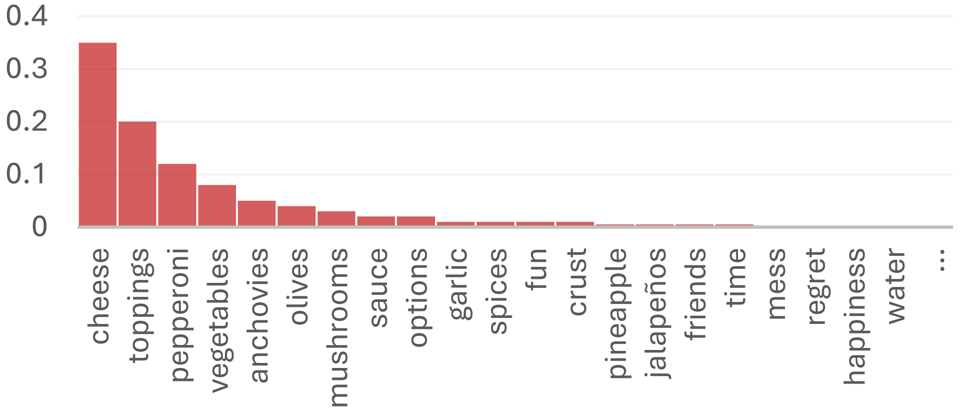

Probability Distribution for Next Token Prediction Over V

- <u>Step-2:</u> Renormalize the scores of the ***k*** tokens to be a legitimate probability distribution.
- After renormalization, $P_{renorm}(w|c) = \frac{P(w|c)}{\sum_{x \in top-k} P(x|c)}$.
  - Renormalized probabilities: **cheese (0.522), toppings (0.299), pepperoni (0.179)**

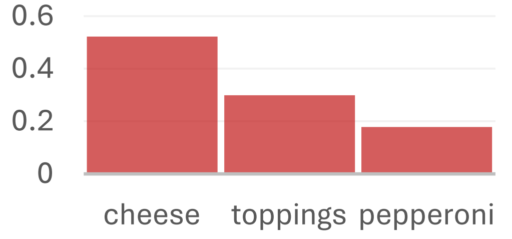
Top-k Sampling: Working Example



Probability Distribution for Next Token Prediction Over V



P_{renorm} after renormalization over top-3 tokens



- Step-3: Randomly sample a token from within these remaining ***k*** most-probable tokens according to its probability after renorm.
- For generating the next token, $w_t \sim P_{renorm}(w_t| c)$

Problem With Top-k Sampling

- In top-k sampling ***k*** is fixed, but the shape of the probability distribution over tokens differs in different contexts.
- For example, if we set k = 10:
  - Sometimes the top 10 tokens will be very likely and include most of the probability mass.
  - But other times the probability distribution will be flatter and the top 10 tokens will only include a small part of the probability mass.

Solution: Top-p / Nucleus Sampling

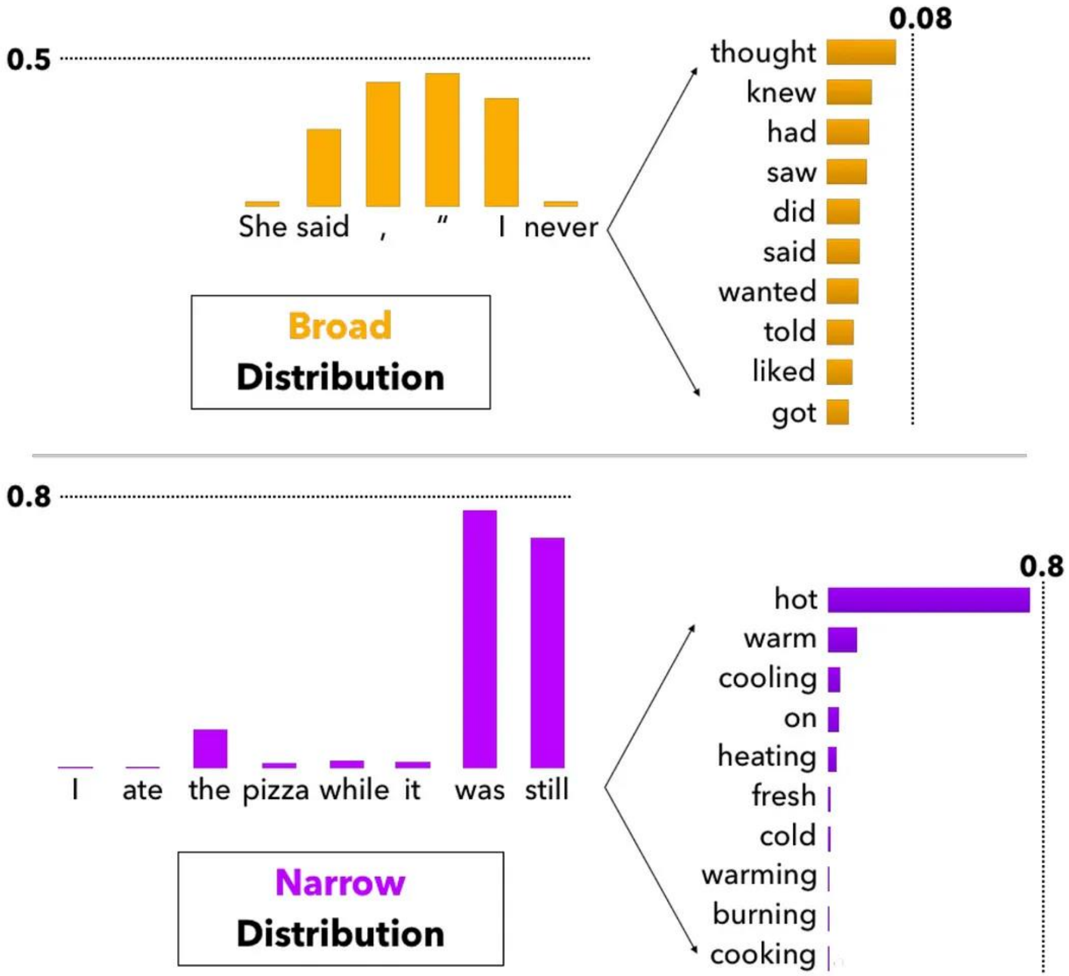


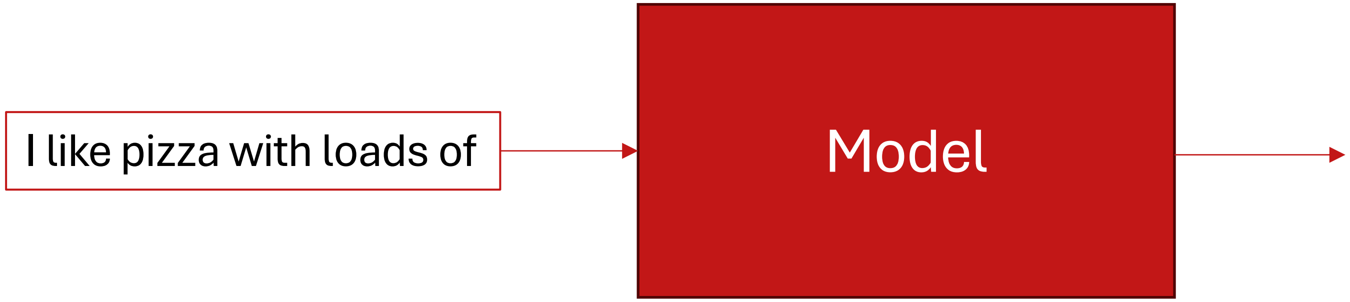
Image source: https://towardsdatascience.com/how-to-sample-from-language-models-682bceb97277

Top-p / Nucleus Sampling

- The goal of nucleus sampling is the same as top-k sampling, which is to truncate the distribution to remove the very unlikely tokens.
- Here, we keep the top ***p*** percent of the probability mass.
  - By measuring probability rather than the number of tokens, the hope is that the measure will be more robust in very different contexts, dynamically increasing and decreasing the pool of token candidates.
- Formally, given a distribution P(w_t | c, $w_{<t}$), the top-p vocabulary $\mathbf{V^{(p)}}$ is the smallest set of tokens such that:

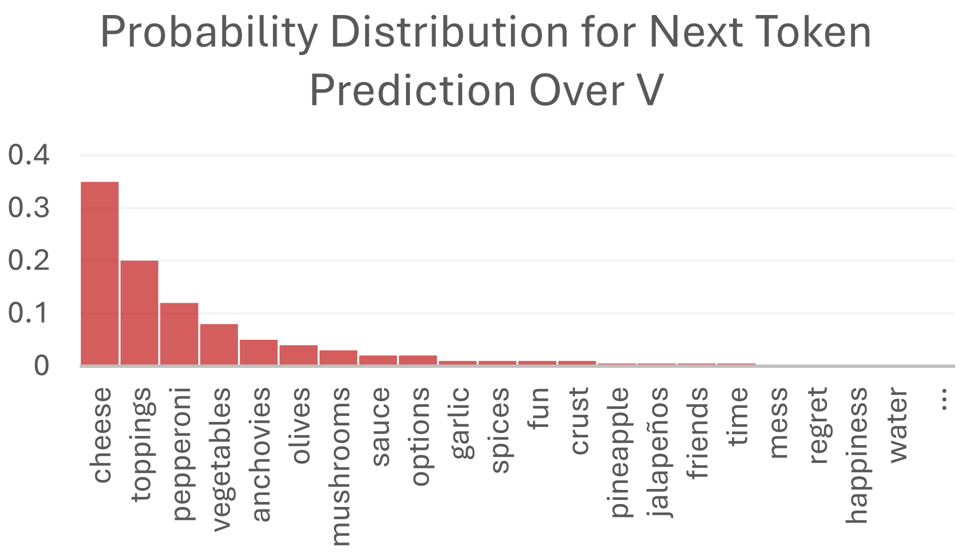
$$\sum_{w \in V^{(p)}} P(w|c, w_{<t}) \geq p$$

Top-p Sampling: Working Example



I like pizza with loads of

Model



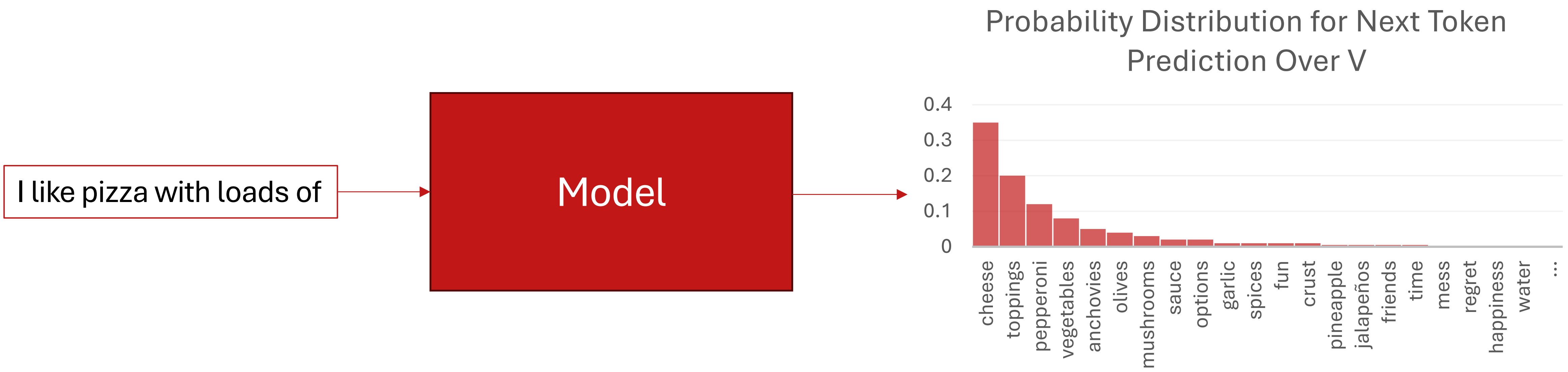
Token (w)	P(w|c)	Cumulative P
cheese	0.35	0.35
toppings	0.20	0.55
pepperoni	0.12	0.67
vegetables	0.08	0.75
anchovies	0.05	0.80

- Let's take **p=0.5**
- Step-1: Get the top-p vocabulary **V$^{(p)}$** , which is the smallest set of tokens such that: $\sum_{w \in V^{(p)}} P(w|c, w_{<t}) \geq p$

In our example, **V$^{(p)}$** consists of:

cheese (0.35), toppings (0.20)

Top-p Sampling: Working Example



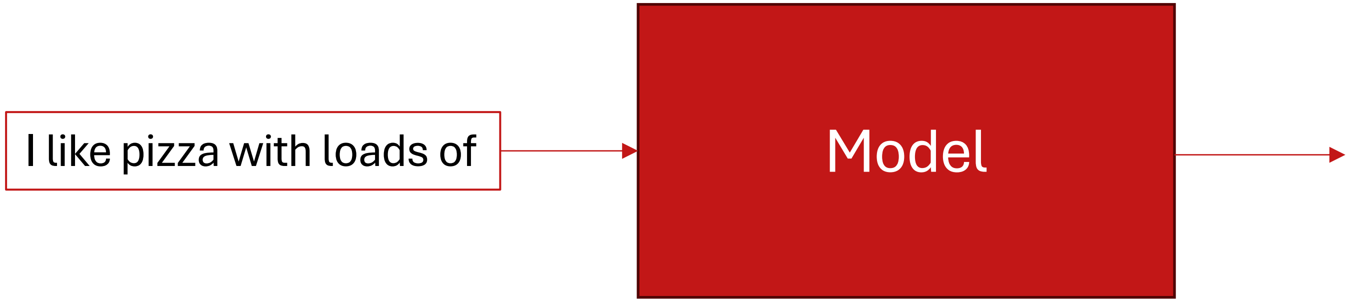
I like pizza with loads of

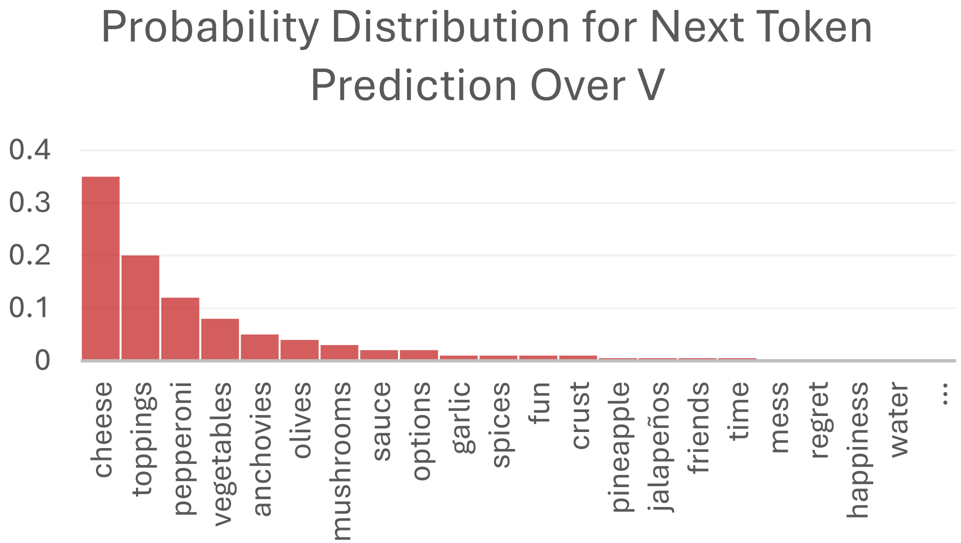
Model

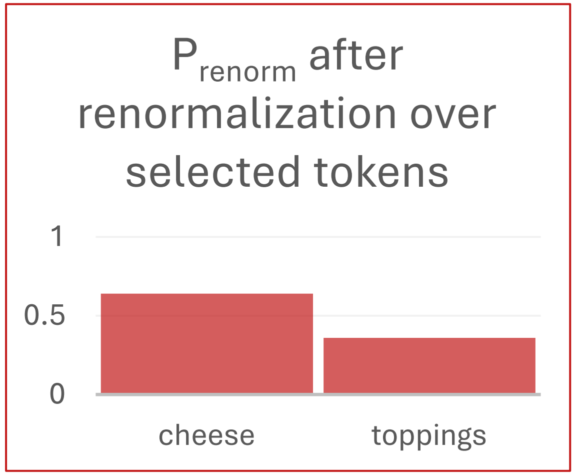
- <u>Step-2:</u> Renormalize the scores of the selected tokens (same procedure as top-k).

Renormalized probabilities: **cheese** ($\frac{0.35}{0.35+0.20} = 0.64$)**, toppings** ($\frac{0.20}{0.35+0.20} = 0.36$)

Top-p Sampling: Working Example







- Step-3: Randomly sample a token from within these selected tokens according to its probability after renormalization.
- For generating the next token, $w_t \sim P_{renorm}(w_t| c)$

Temperature Sampling

- In temperature sampling, we don’t truncate the distribution but instead reshape it.
- The **temperature parameter (τ)** is incorporated while computing ***softmax*** over the logits of the tokens (for next token prediction).

Normal Softmax

$$P(t_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

- $P(t_i)$: Probability assigned to token t_i
- z_i : Logit for token t_i

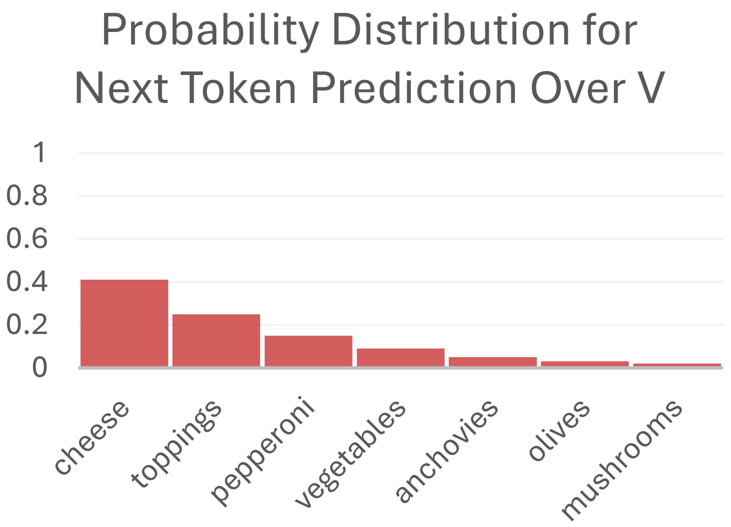
Softmax with Temperature

$$P(t_i) = \frac{\exp(z_i/\tau)}{\sum_j \exp(z_j/\tau)}$$

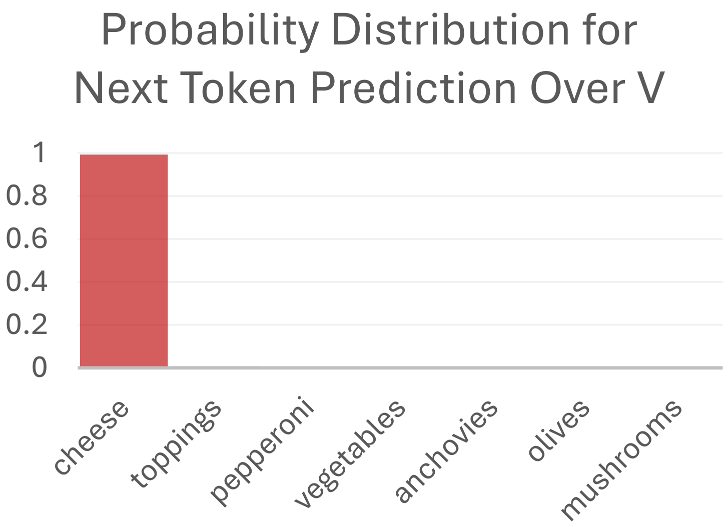
- τ : Temperature parameter
  - $\tau > 0$

Effect of Temperature

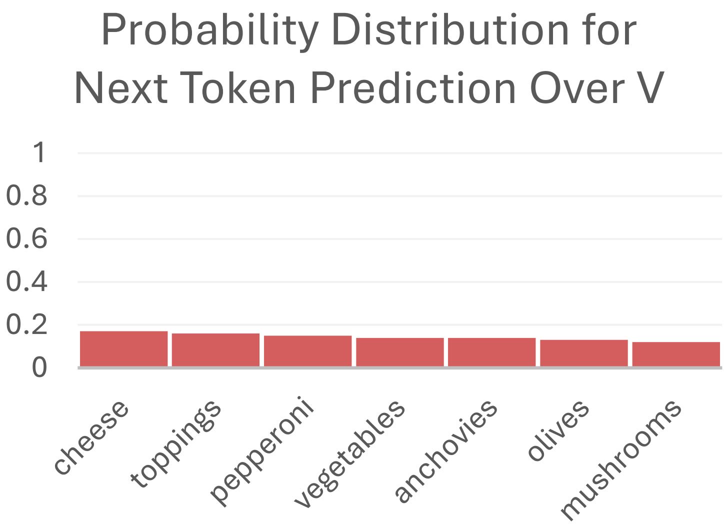
I like pizza with loads of



τ =1

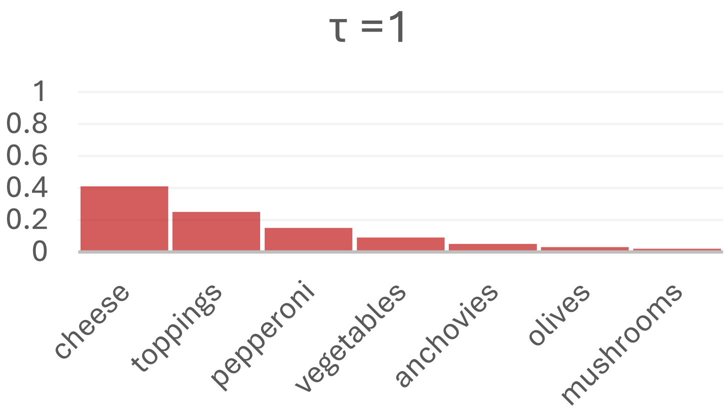


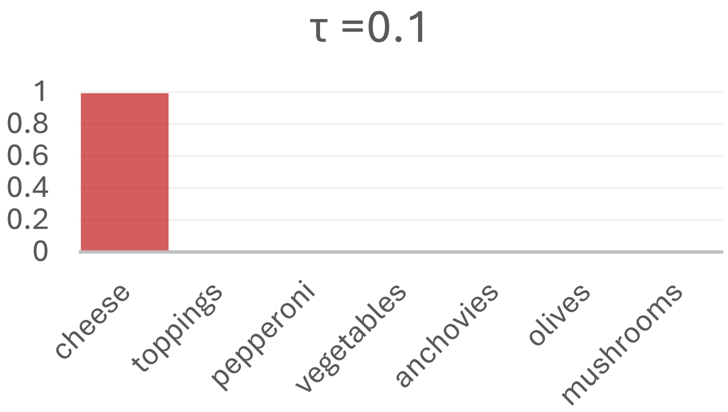
τ =0.1

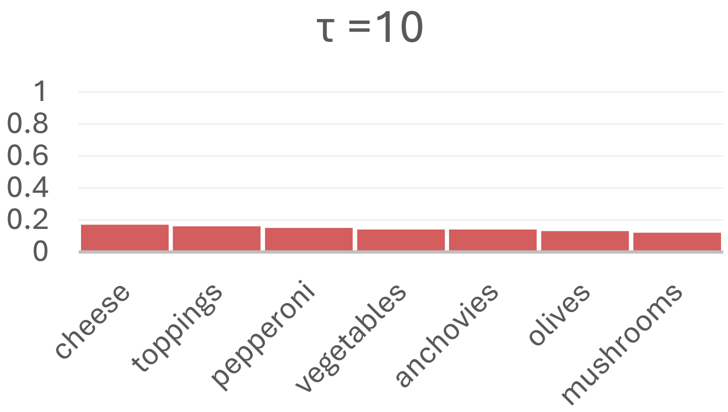


τ =10

Effect of Temperature







- A low temperature sharpens the probabilities, making the model confident and more deterministic.
  - As τ approaches 0 the probability of the most likely word approaches 1.
  - Lower temperatures make the model more “confident” which can be useful in applications like question answering.
- A high temperature flattens the probabilities, encouraging diversity and creativity in outputs but with a risk of incoherence.
  - Thus, higher temperatures make the model more “creative” which can be useful when generating prose, for example.