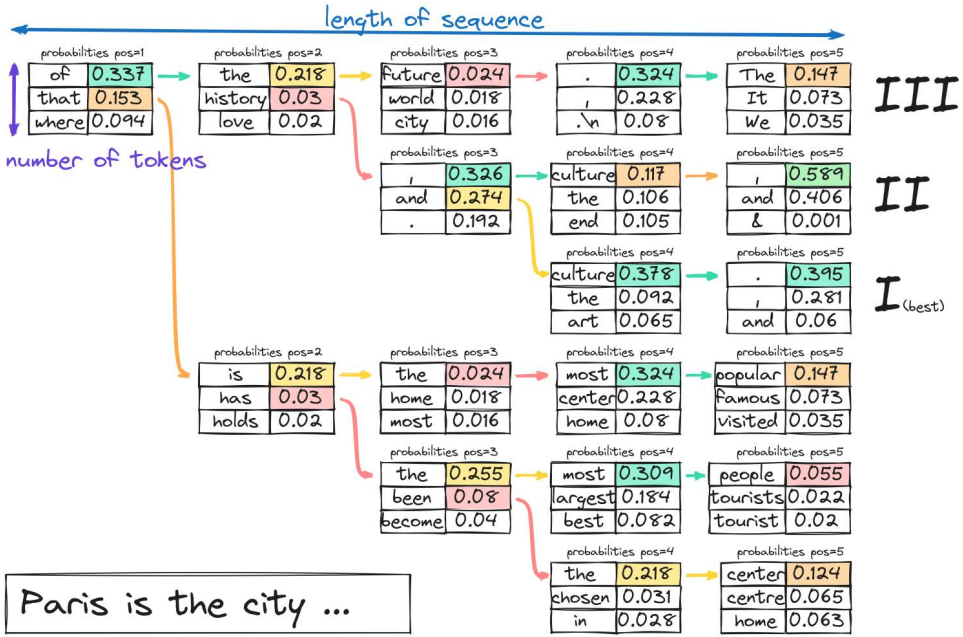
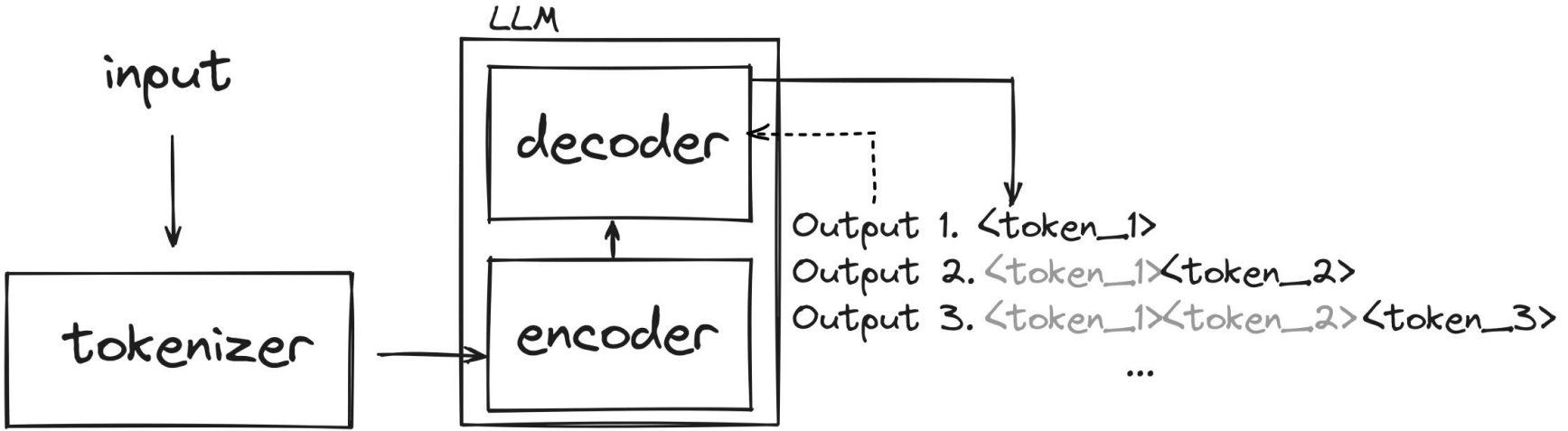
How is text generated?



Dr. Ivan Reznikov, Principal Data Scientist

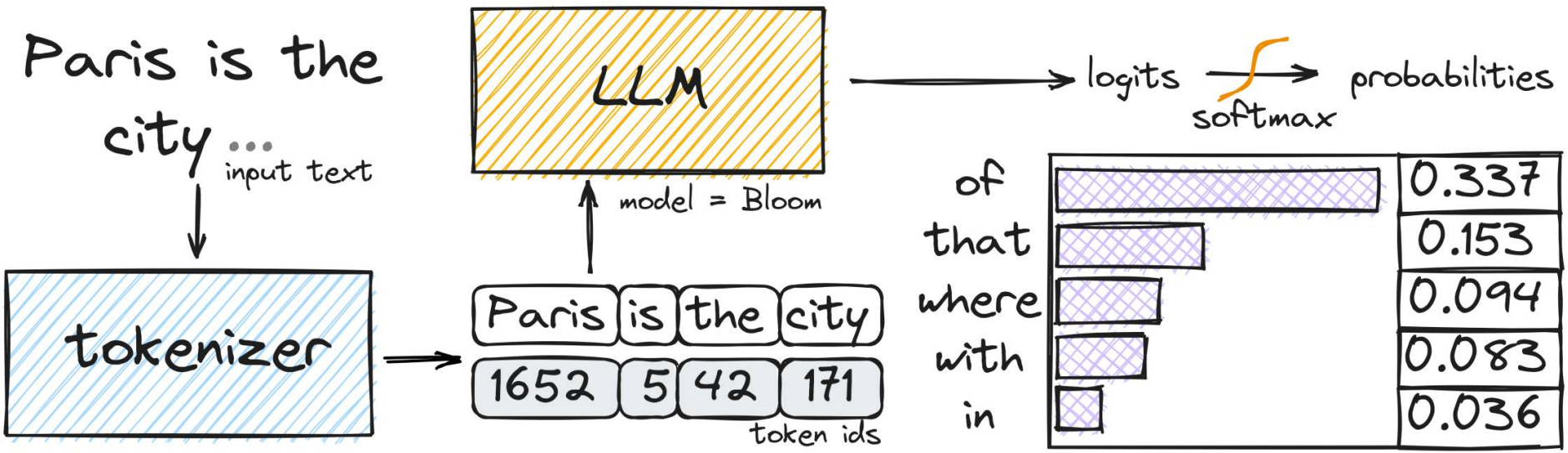
Three main steps for text generation

- The input text is passed to a tokenizer that generates unique numerical representation of every token.
- The tokenized input text is passed to the Encoder part of the pre-trained model. The Encoder processes the input and generates a feature representation that encodes inputs meaning and context.
- The Decoder takes the feature representation from the Encoder and starts generating new text based on that context token by token.



Decoding the outputs

Say, we want to generate the continuation of the phrase "*Paris is the city ...*".
The Encoder (we'll be using Bloom-560m model) sends logits for all the tokens we have (if you don't know what logits are — consider them as scores) that can be converted, using softmax function, to probabilities of the token being selected for generation.

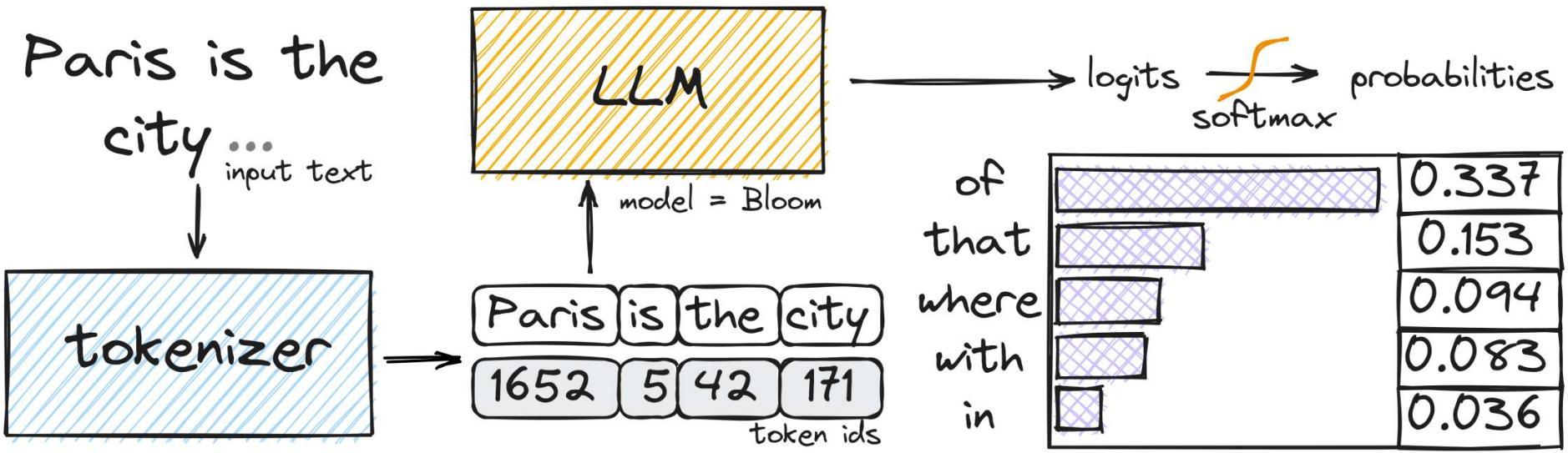


Possible following tokens

Paris is the city **of** love.

- Paris is the city **that** never sleeps.
- Paris is the city **where** art and culture flourish.
- Paris is the city **with** iconic landmarks.
- Paris is the city **in** which history has a unique charm.

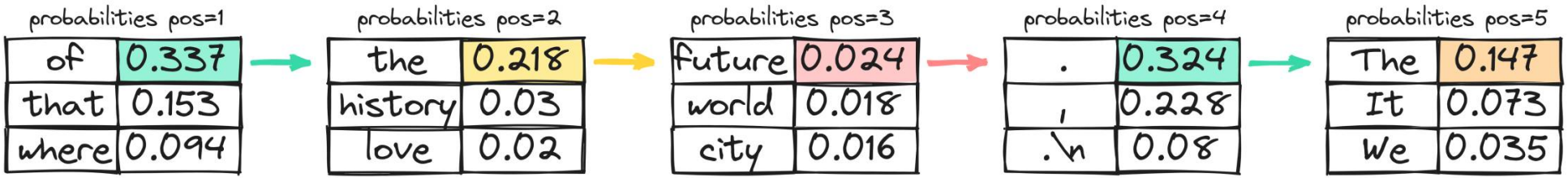
How to choose the next token?



1. Greedy Sampling

In the greedy strategy, the model always chooses the token it believes is the most probable at each step — it doesn't consider other possibilities or explore different options. The model selects the token with the highest probability and continues generating text based on the selected choice.

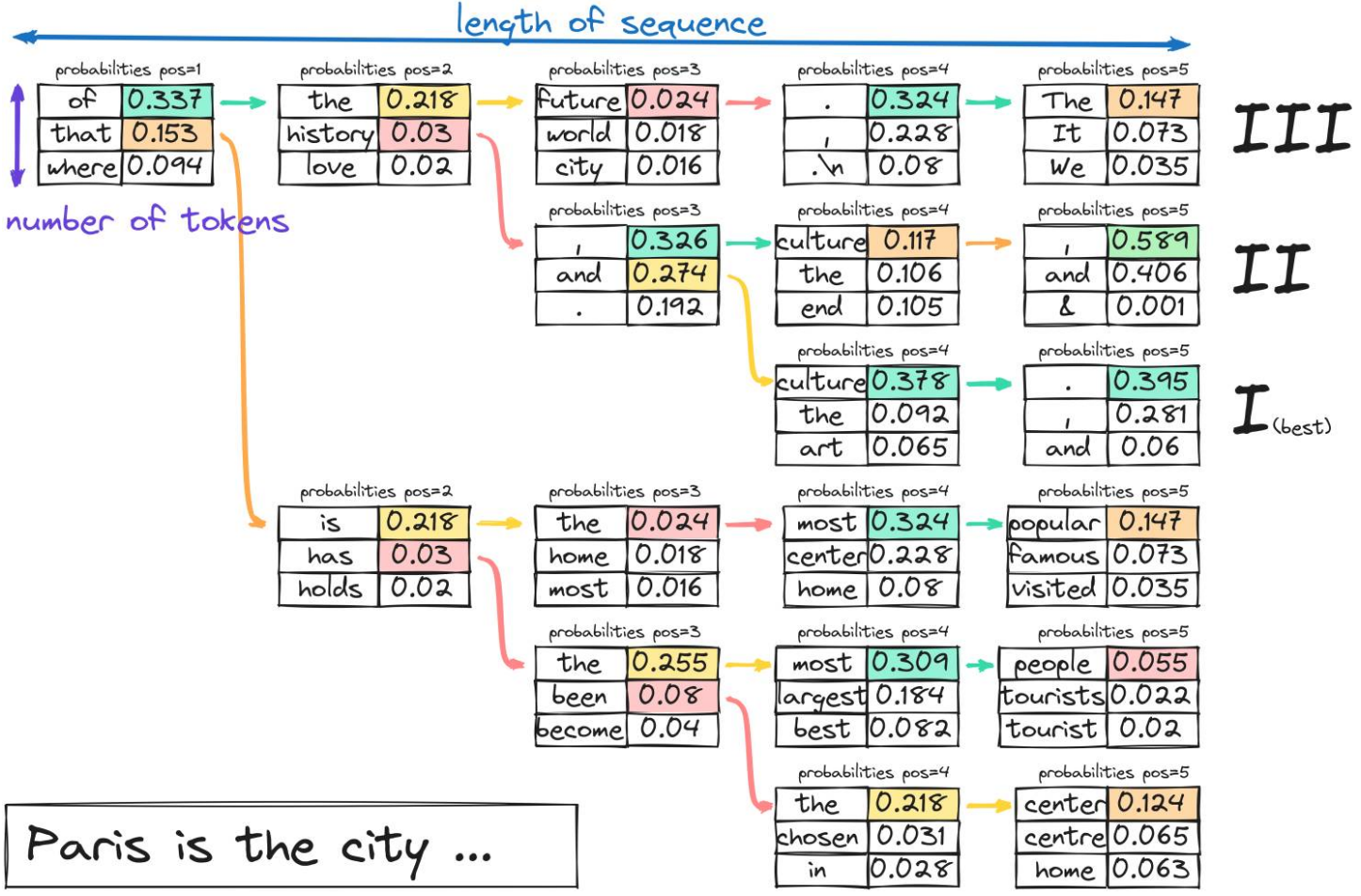
Generated output: Paris is the city of the future. The



2. Beam search

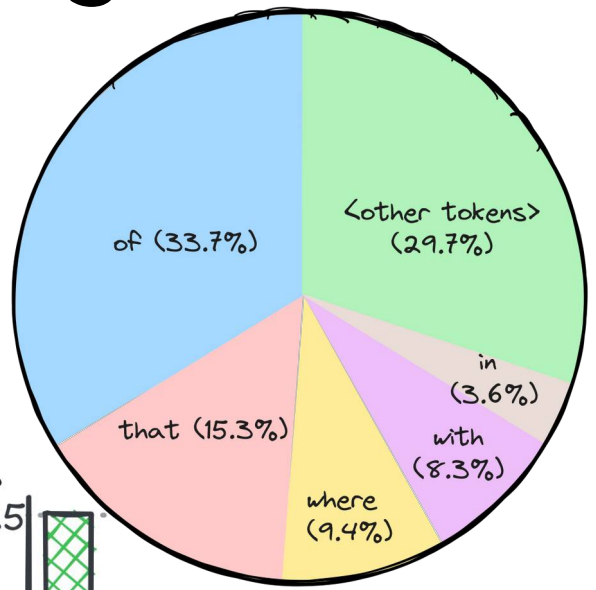
In beam search, instead of just considering the most likely token at each step, the model considers a set of the top "k" most probable tokens. This set of k tokens is called a "beam."

Generated output: Paris is the city of history and culture.



3. Normal Random Sampling

In normal random sampling you select the next word by choosing a random value and mapping it to the token got picket. Imagine it as spinning a wheel, where the area of each token is defined by its probability. The higher the probability — the more chances the token would get selected.

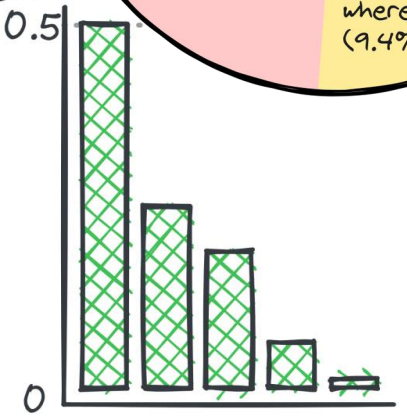


	logits
token A	-1.806
token B	-2.499
token C	-2.786
token D	-3.885
token E	-5.494

softmax →

probabilities
0.494
0.247
0.185
0.062
0.012

$$softmax(x_i) = \frac{e^{x_i}}{\sum_i e^{x_i}}$$

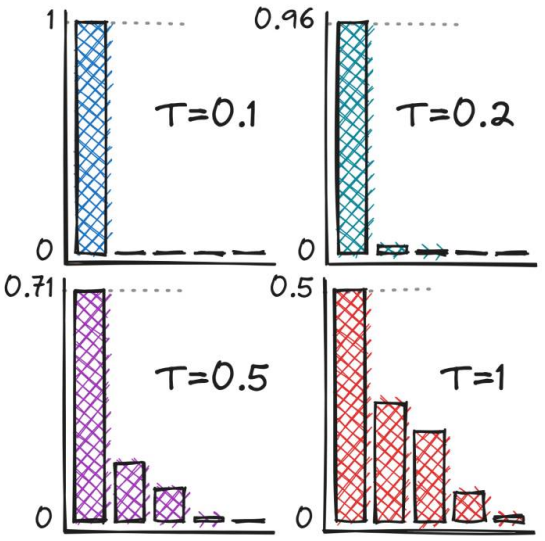


4. Random Sampling + Temperature

We've been using the softmax function to convert logits to probabilities. We now introduce temperature — a hyperparameter that affects the randomness of the text generation. The difference with the classic softmax is in the denominator - we divide by T. Higher values of temperature (e.g., 1.0) make the output more diverse, while lower values (e.g., 0.1) make it more focused and deterministic. In fact, T = 1 will lead to the softmax function we used initially.

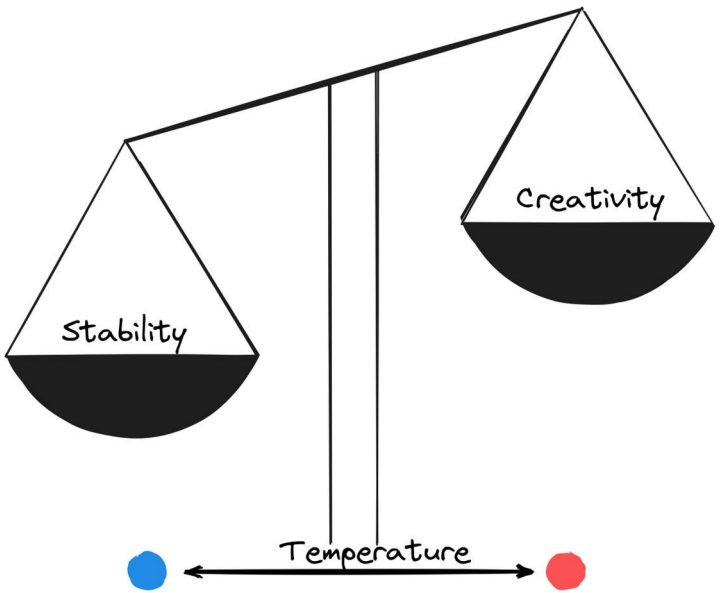
	logits
token A	-1.806
token B	-2.499
token C	-2.786
token D	-3.885
token E	-5.494

$$softmax(x_i) = \frac{e^{x_i/T}}{\sum_i e^{x_i/T}}$$



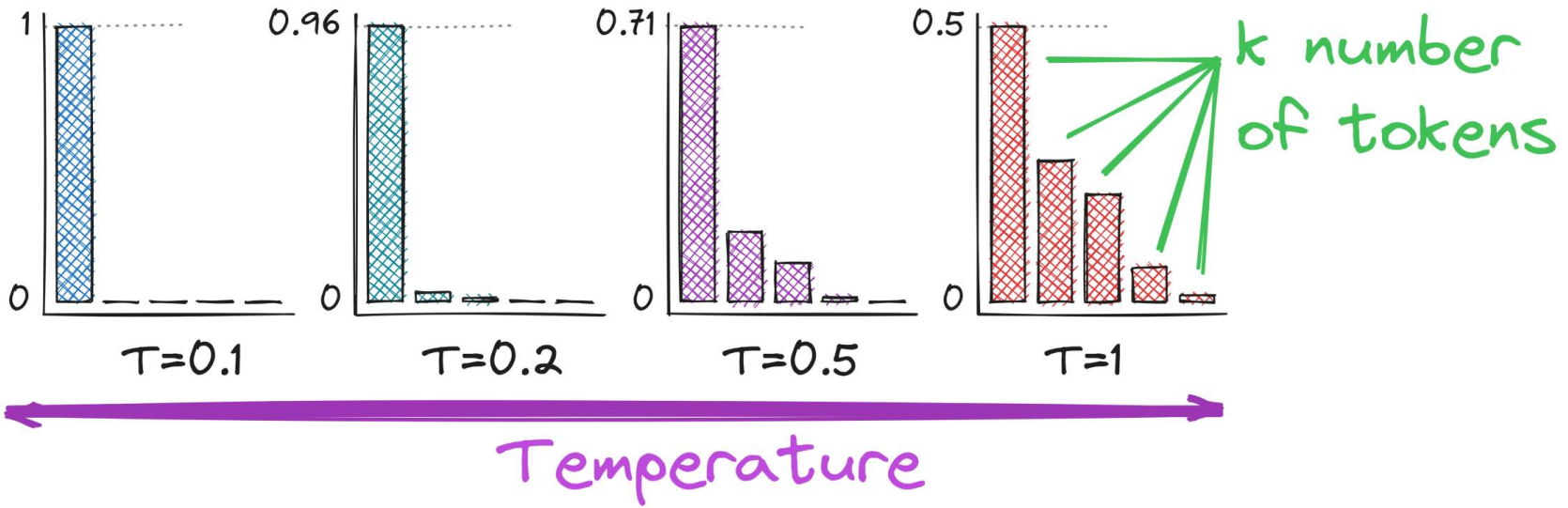
Temperature

Temperature controls the randomness of token selection during decodingHigher temperature boosts creativity, whereas lower temperature is more about coherence and structure. While embracing creativity allows for fascinating linguistic adventures, tempering it with stability ensures the elegance of the generated text.



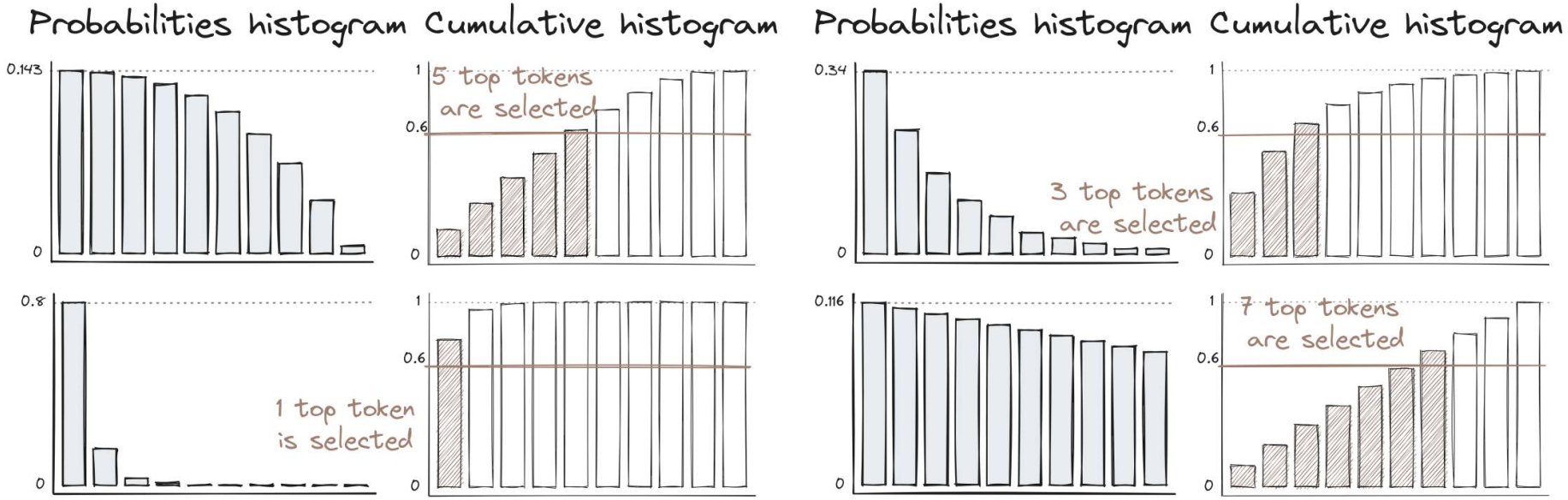
5. Top-k sampling

We can now shift probabilities with temperature. Another enhancement is to use top-k tokens rather than all of them. This will increase the stability of the text generation, not decreasing creativity too much. Basically, it's now random sampling with temperature for only top k tokens.



6. Top-p or nucleus sampling

Nucleus sampling is designed to address some limitations of different sampling techniques. Instead of specifying a fixed number of "k" tokens to consider, a probability threshold "p" is used. This threshold represents the cumulative probability that you want to include in the sampling. The model calculates the probabilities of all possible tokens at each step and then sorts them in descending order.



Summary

Method	Description	Diversity	Performance	Quality of Output
Greedy Search	Selects the most probable token at each step.	Low	High	May lack diversity
Beam Search	Considers top-K probable tokens in parallel.	Moderate	Medium-High	Improved diversity
Random Sampling	Randomly selects tokens based on probabilities.	High	Medium	May lack coherence
Random Sampling with Temperature	Adds randomness based on temperature.	High	Medium	More controlled randomness
Top-K Sampling	Selects top-K probable tokens randomly.	High	Medium	Improved diversity
Nucleus Sampling	Selects tokens until cumulative probability ≤ p.	High	Medium-High	Improved coherence