**LLM generation methods**

Great! Let’s go through the **main LLM generation methods** one by one — such as **greedy decoding, beam search, top-k sampling, top-p sampling (nucleus), and reinforcement learning** — with **simple explanations** and **examples** for each.

## **🧠 1. Greedy Decoding**

**What it does:** It always picks the word with the highest probability at each step — the "most likely" next word.

**Pros:** Fast and simple.  
 **Cons:** Often produces dull or repetitive text.

**Example:** Prompt: *"The weather today is"*

Greedy decoding might output:  
 **"The weather today is sunny and warm."** (It's the safest, most probable output.)

## **🧠 2. Beam Search**

**What it does:** It keeps multiple “paths” (beams) at each step and selects the most promising one in the end. It’s like testing different versions of a sentence in parallel.

**Beam width**: how many paths are explored at once.  
 More width = more options checked = slower but better.

**Pros:** More fluent and coherent than greedy.  
 **Cons:** May still miss creative or accurate outputs; can favor generic phrases.

**Example:** Prompt: *"Artificial intelligence is"*

Beam search might try:

* "Artificial intelligence is changing the world."
* "Artificial intelligence is the future of technology."
* "Artificial intelligence is a growing field."

It picks the best full sentence from these options.

## **🧠 3. Top-k Sampling**

**What it does:** Instead of always choosing the top word, it looks at the top **k most probable** next words, then **randomly selects** one.

**Pros:** Adds creativity and variation.  
 **Cons:** Can be less coherent if k is too high.

**Example (k=3):** Prompt: *"The cat jumped on the"*

Top-k may choose between:

* "sofa"
* "table"
* "bed"

Randomly picks one — maybe “bed” this time.

## **🧠 4. Top-p Sampling (Nucleus Sampling)**

**What it does:** Looks at a group of next words whose **total probability adds up to p** (like 0.9 or 90%). Then randomly selects from them.

**Pros:** Smarter randomness — balances coherence and creativity.  
 **Cons:** Can still go off-track in long texts.

**Example (p=0.9):** Prompt: *"He opened the mysterious"*

Top-p may include words like:

* "box"
* "door"
* "letter"  
   (Which together make up 90% of the likely next words.)

Randomly picks one — like “door”.

## **🧠 5. Reinforcement Learning (like RLHF – Reinforcement Learning from Human Feedback)**

**What it does:** After generating many outputs, the model is trained to prefer **those that humans liked better** (e.g., more helpful, polite, or safe).

**Pros:** Improves quality and reduces harmful or biased output.  
 **Cons:** May still hallucinate if human feedback wasn’t perfect.

**Example:** If a model used to say:  
 *"Vaccines cause X problem"* — and humans flagged that as false,

Over time, the model will **learn to avoid** such answers and say:  
 *"Scientific studies show vaccines are safe."*

## **🔍 How They Cause Hallucinations**

| **Method** | **Why Hallucinations Happen** |
| --- | --- |
| Greedy Decoding | Too rigid — picks the safest word, might miss context. |
| Beam Search | Over-optimizes fluency, prefers generic over accurate. |
| Top-k Sampling | Adds randomness — can choose less likely (wrong) words. |
| Top-p Sampling | Same — can sacrifice accuracy for creativity. |
| Reinforcement Learning | Depends on training feedback — if humans missed errors, model keeps making them. |

If you want, I can also make a **table or chart** to compare all methods visually or explain how they are used differently in research vs production models.