

The Art of Picking the Next Token

Amanvir Parhar

This is an interactive, scrollytelling-style work that thoroughly explains the concept of **sampling** in the context of large language models. Please watch **demo.mp4** for a video demo of this work in action, and follow the instructions (last section on this page) to launch and run the actual web experience.

Target audience: This work is intended for those who have interacted with LLMs like ChatGPT in some capacity, but would like to learn more about the technical details surrounding LLM-based text generation.

Expected time: 5-10 mins

Detailed description: This work explores how a single token is picked from the next-token probability distribution generated by a large language model (LLM). It begins by introducing the viewer to the technical basics of language models, which are a necessary prerequisite to understanding the content that follows. It then touches upon greedy decoding (arguably the simplest way to pick a token) and subsequently dives into sampling - what it is, how it's done, and why it's used in the first place. After covering the basics of sampling, multiple thorough explanations of key sampling parameters follow; the work explains how these parameters are used to modify the underlying next-token probability distribution before sampling. At the end, the viewer has a chance to generate their own next-token probability distributions by providing some text as input and feeding it to GPT-2 small (124M params). From there, they can then adjust sampling parameters to their liking, sample a token from this distribution, and autoregressively repeat this process. This interactive, inference-time experience helps them build a strong intuition for how LLMs generate text.

Instructions: cd into the the-art-of-picking-the-next-token directory; then, run `npm install && npm run dev` and open `http://localhost:5173` in a web browser. On the first run-through, the web experience will take some extra time to load, as the GPT-2 model weights will have to be downloaded. These weights are cached for subsequent runs, so the experience should load pretty quickly after the initial run.