LLM Token Prediction Explorer: Step-by-Step Instructions

Part 1: Setting Up the Notebook

1. Go to the workshop notebook at colab.research.google.com/ drive/1t5sU84hi8rFTOWUrELDvZlioEulJqy7v.
2. Save a copy to your own Google Drive by going to File → Save a copy in Drive. A new tab will open with your personal copy of the notebook.
3. In your copy, change the runtime to GPU. Go to Runtime → Change runtime type. In the dropdown under "Hardware accelerator," select T4 GPU, then click Save.
4. Run the first cell by clicking the play button to its left. This cell installs the required software libraries and takes about one minute to complete.
5. Run the second, third, and fourth cells in order. The second cell downloads and loads the language model (TinyLlama-1.1B). The third cell defines the prediction function. The fourth cell runs a quick test to confirm the model is working — you should see a list of predicted words printed in the output.
6. Run the fifth cell. This cell starts the API server. When it finishes, it will print a public URL in the output area that looks something like:

> Running on public URL: https://abc123def456.gradio.live

Copy this URL. You will paste it into the interface in the next section. The cell will show a green checkmark, but the server continues running in the background — do not close the Colab tab.

Part 2: Opening the Interface

7. Go to the Token Prediction Explorer interface at simulation-and-society.org/workshops/ ai_futures/token-predictor-4b.html.
8. At the top of the page, paste the Gradio URL you copied from the notebook into the field labeled "Paste Gradio public URL" and click Connect. The status should read " Connected."

Part 3: Using the Next-Token Predictor

9. Click the hamburger menu ( ) in the top left corner of the interface. Select **Next-Token Predictor** from the dropdown menu.
10. The screen is divided into two side-by-side columns, each with its own set of controls: Temperature, Top-K, Likelihood Rank, and System Prompt. Both columns share a single prompt field at the top.
11. Type an incomplete sentence into the prompt field, such as:

> The most important thing about education is

candidates.

13. To compare how different parameter settings affect the predictions, adjust the controls in one column while leaving the other unchanged. For example, lower the Temperature in the left column to 0.1 and leave the right column at 1.0, then click Predict → again. The two columns will show how the same prompt produces different probability distributions under different settings.

Temperature controls how sharply the model discriminates between candidates. Low values concentrate probability on the top candidate. High values spread probability more evenly across all candidates.

Top-K sets a hard cutoff on the number of candidates. With Top-K set to 5, only the top 5 words are considered. Everything else is eliminated regardless of its probability.

Likelihood Rank shifts which part of the model's ranked list you see. At 100 (the default), you see the most probable words. Lower values show less likely candidates — words the model considers possible but improbable.

System Prompt lets you prepend an instruction to the model before your prompt. Typing an instruction here (such as "Respond as a scientist") changes the probability distribution because the model now generates predictions in the context of that instruction.

Part 4: Using the Full Sentence Predictor

14. Click the hamburger menu ( ) and select **Full Sentence Predictor**
15. Type an incomplete sentence into the prompt field, such as:

> A university should

16. Adjust the controls below the prompt field. **Tokens to Generate** sets how many words the model will produce. **Temperature** **Top-K**, and **System Prompt** work the same way as in the Next-Token Predictor.
17. Click **Generate →**. The model will generate words one at a time, appearing in the Generated Text area. Each word is color-coded by the model's confidence: darker green means higher confidence, orange means lower confidence.
18. Below the generated text, the **Token Log** shows each generated word with its probability, displayed as a bar chart. This log records the model's confidence at every step of the generation.

Clicking a Word

This panel shows the same kind of ranked list you saw in the Next-Token Predictor, but now embedded at a specific position within a generated sentence.

Branching

20. In the alternatives panel, click on a different word. The interface will branch the sentence: it keeps everything up to the word you clicked, substitutes the alternative you chose, and generates the rest of the sentence forward from that new starting point. The original sentence remains visible above, with the branch point marked, and the new sentence appears below it.
21. You can create multiple branches from different positions in the sentence. Each branch shows how a different word choice at a single step produces a different sentence from that point forward.

Troubleshooting

Error" when clicking Predict or Generate. Make sure the Gradio URL in the API field includes https:// and does not have a trailing slash. Make sure the fifth cell in the notebook has finished running and the Colab tab is still open.

No predictions appear. Check the Colab notebook for error messages in the cell output. Make sure you ran all five cells in order, starting from the first.

The interface says "Predicting…" or "Generating…" for a long time. The model runs on the Colab GPU. If the notebook has been idle for a while, it may need a moment to respond. If it does not return after 30 seconds, check that the Colab tab is still open and the runtime has not disconnected.

The Gradio URL stopped working. Colab sessions time out after a period of inactivity (typically 30–90 minutes). If this happens, re-run the fifth cell in the notebook to generate a new URL, then paste the new URL into the interface and reconnect.

Words appear garbled or include special characters. The model uses sub-word tokenization, so some generated tokens may include fragments like " " (a space marker) or partial words. This is expected behavior — it reflects how the model processes language internally.