**My road map in LLM**

references :

https://www.youtube.com/@AndrejKarpathy

https://www.youtube.com/watch?v=zduSFxRajkE

https://github.com/brevdev/notebooks/blob/main/mistral-finetune-own-data.ipynb

https://medium.com/@thakermadhav/build-your-own-rag-with-mistral-7b-and-langchain-97d0c92fa146

https://www.langchain.com/

<https://huggingface.co/blog/how-to-generate>

Tokenization: <https://github.com/openai/openai-cookbook>

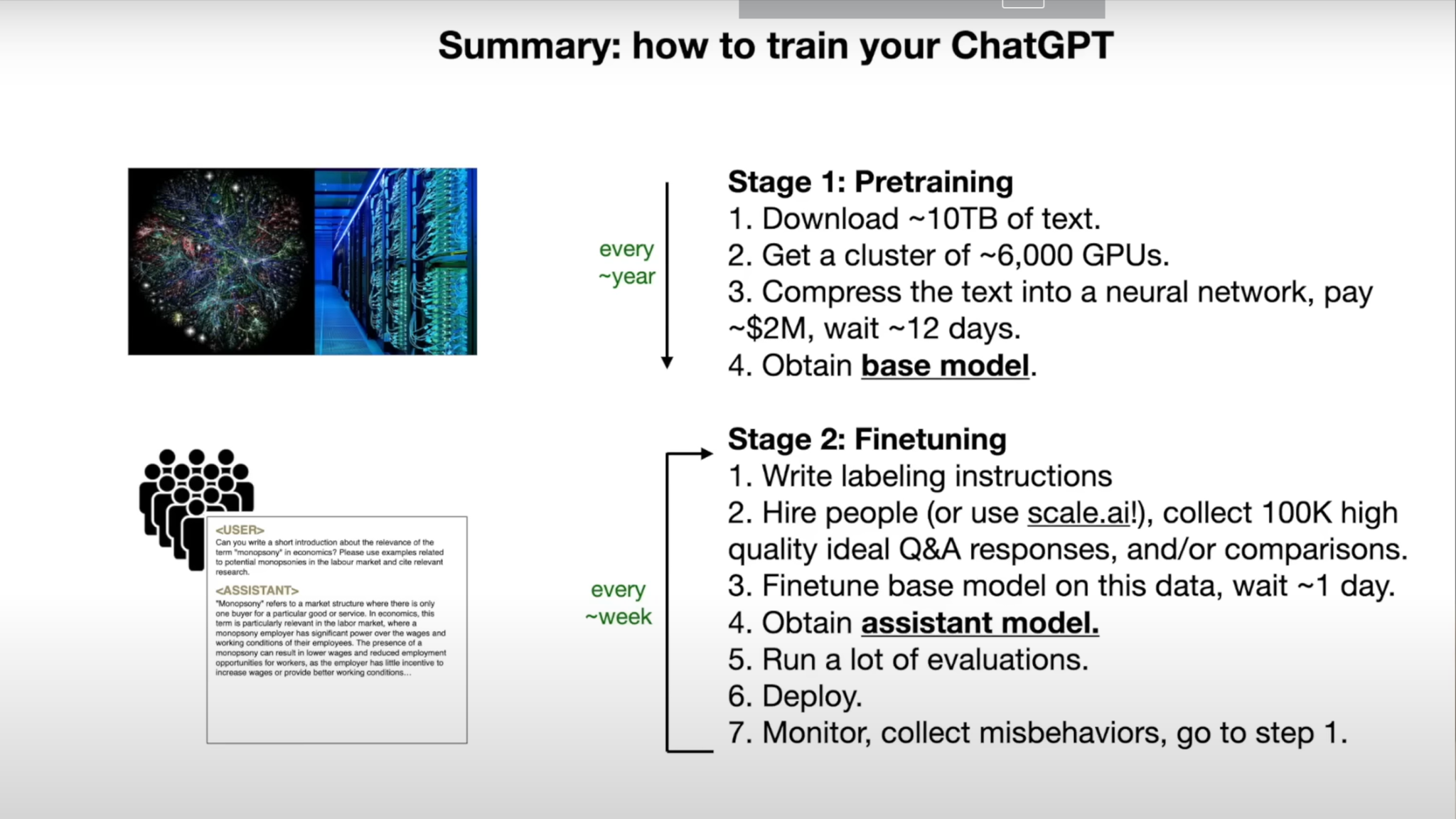
Fine-tuning: https: //[github.com/brevdev/notebooks/blob/main/mistral-finetune-own-data.ipynb](http://github.com/brevdev/notebooks/blob/main/mistral-finetune-own-data.ipynb)[first link]

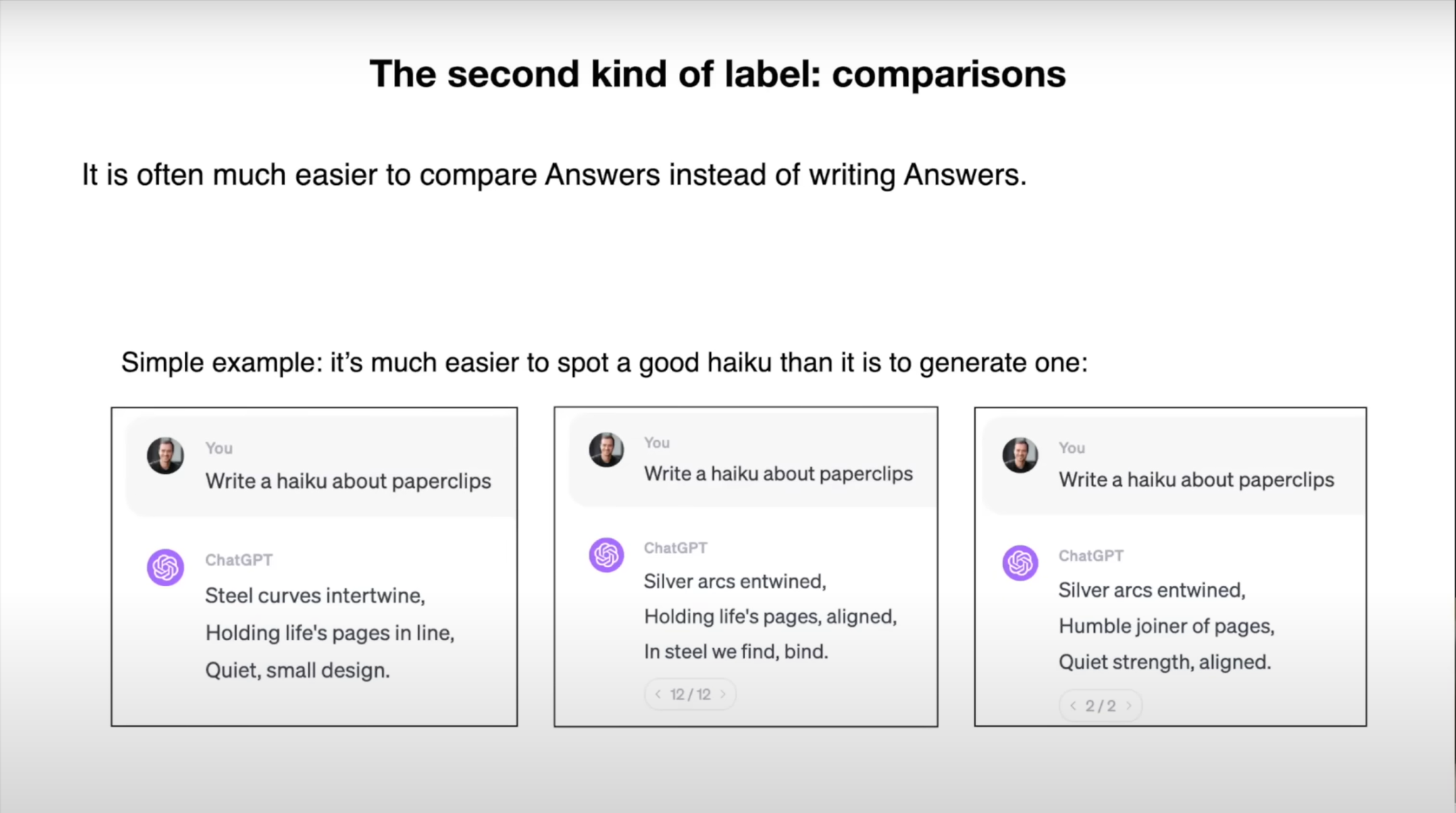
Future: <https://github.com/ashishpatel26/LLM-Finetuning> It is advanced for fine-tuning after the first link

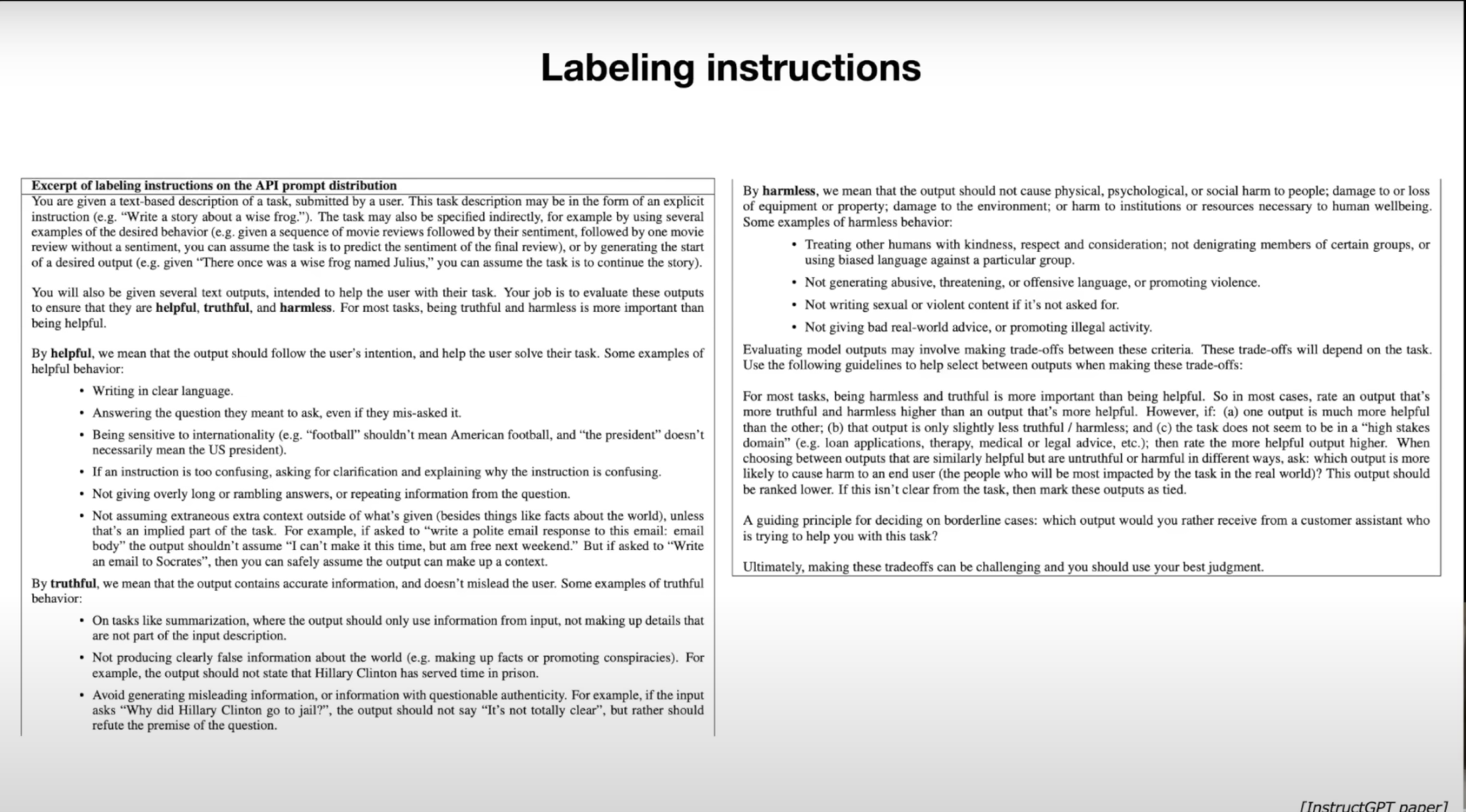
The below link is for Tokenization, especially for LLm:

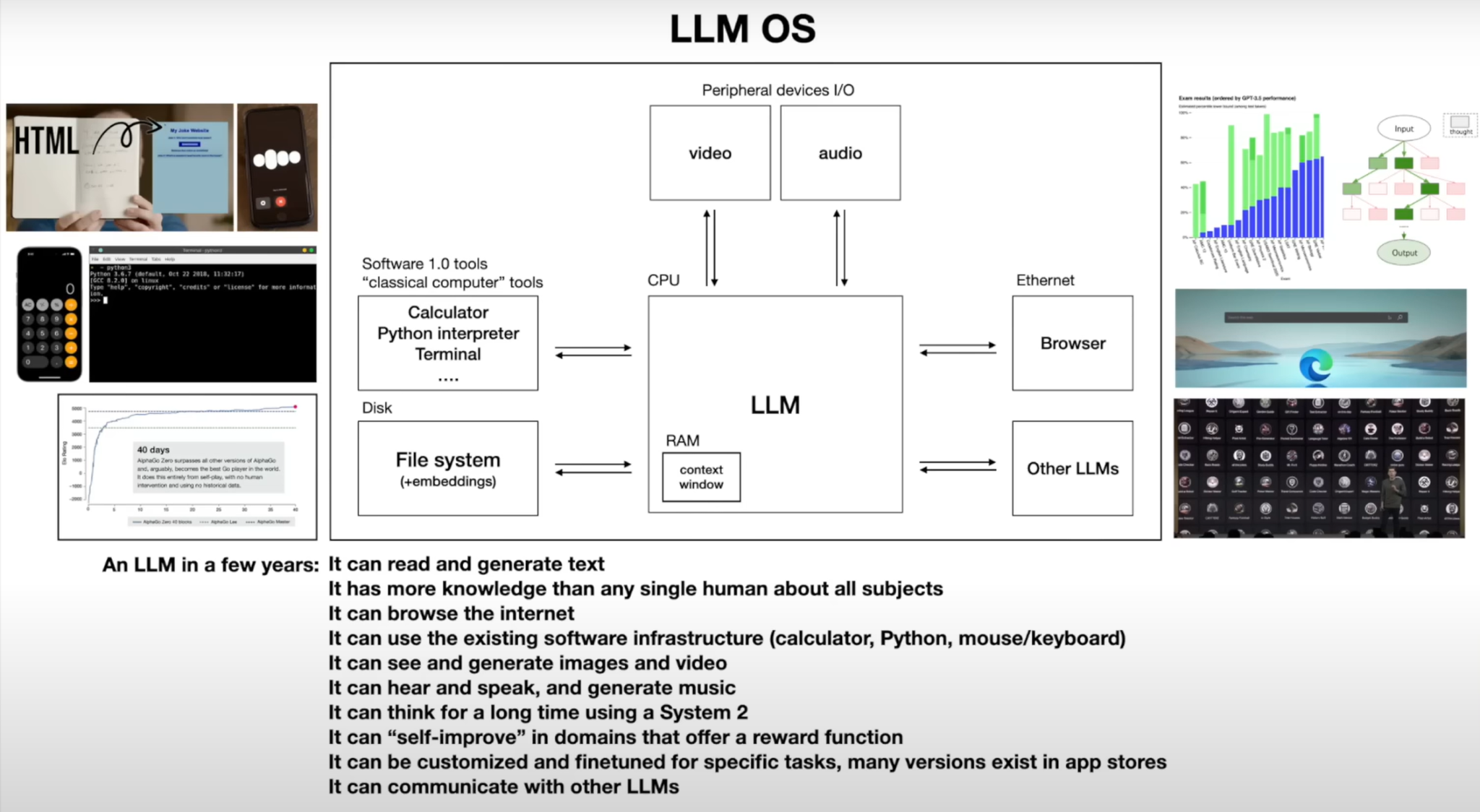
<https://github.com/openai/openai-cookbook/blob/main/examples/How_to_count_tokens_with_tiktoken.ipynb>

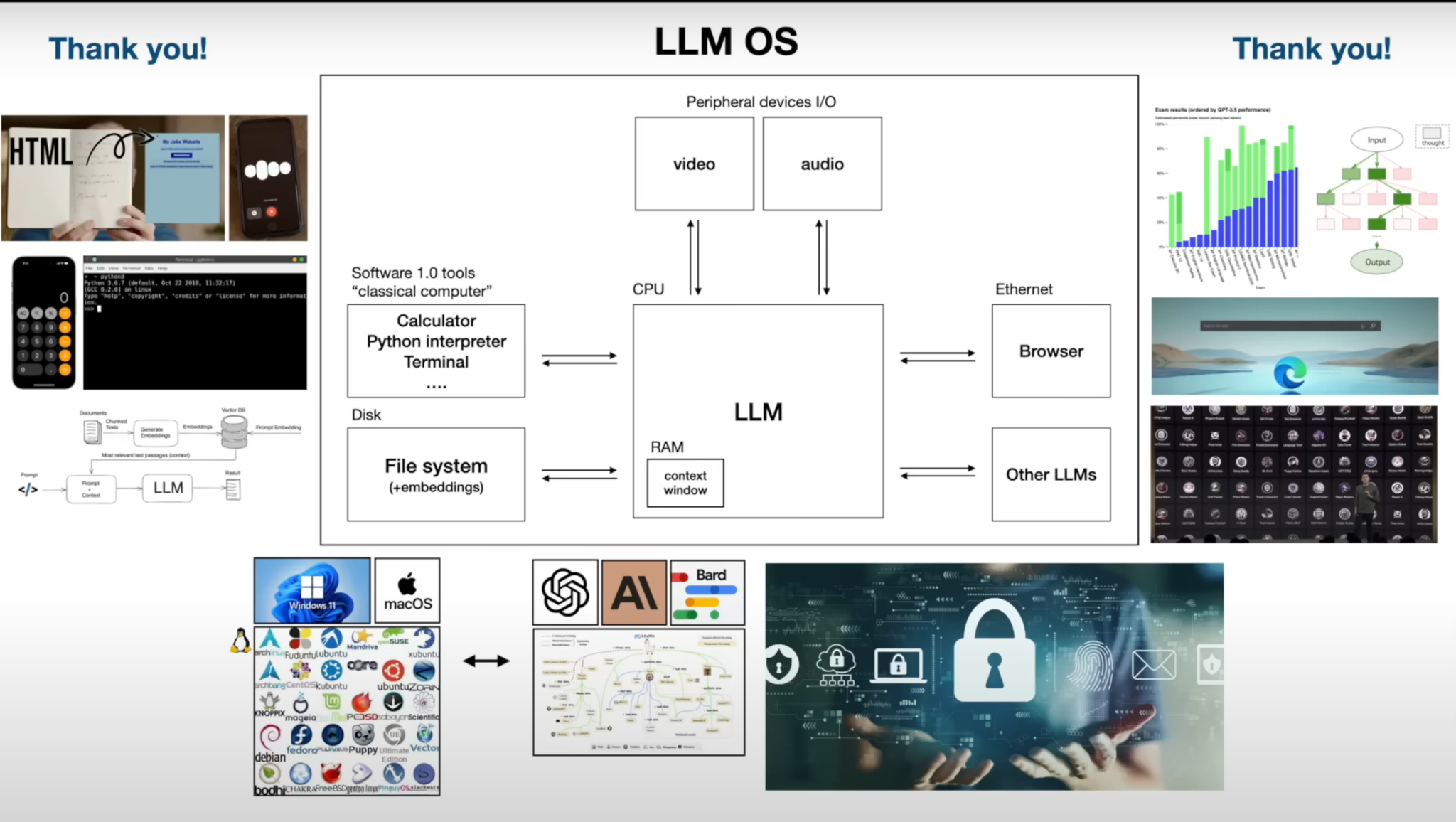
What is LLM?











**What is tokenization?**

Reference: “<https://huggingface.co/docs/transformers/en/preprocessing>”

Before you can train a model on a dataset, it needs to be preprocessed into the expected model input format. Whether your data is text, images, or audio, they must be converted into tensors batches. In Huginface there is a library (Transformers: it provides a set of preprocessing classes to help prepare your data for the model.)

* Text uses a [Tokenizer](https://huggingface.co/docs/transformers/en/main_classes/tokenizer) to convert text into a sequence of tokens, create a numerical representation of the tokens, and assemble them into tensors. The main tool for preprocessing textual data is a [tokenizer](https://huggingface.co/docs/transformers/en/main_classes/tokenizer). A tokenizer splits text into *tokens* according to a set of rules. The tokens are converted into numbers and then tensors, which become the model inputs. The tokenizer adds any additional inputs required by the model.

**What is the** [**Tokenizer**](https://huggingface.co/docs/transformers/en/main_classes/tokenizer)**?**

The Hugging Face Tokenizer documentation explains how tokenizers prepare inputs for models, including tokenization, converting tokens to IDs, and encoding/decoding sequences. There are two implementations: a full Python version and a faster Rust-based version. Key classes include PreTrainedTokenizer and PreTrainedTokenizerFast, which manage tokenization methods, adding new tokens, handling special tokens, and more. It also details batch encoding, managing token attributes, and configuring tokenization options such as padding, truncation, and special tokens**.**

**Fast tokenizer :**

* Significant Speed-up with Batched Tokenization: When doing batched tokenization, the PreTrainedTokenizerFast class provides a significant speed-up. This is because it leverages the fast implementation in Rust.
* Mapping Between Original String and Token Space: The PreTrainedTokenizerFast class offers methods to map between the original string (characters and words) and the token space

Example of a Fast tokenizer: https://github.com/monirmo97/LLM/blob/main/Tokenizer.ipynb

**Different types of tokenization?**

**Link of GitHub**: <https://github.com/monirmo97/LLM/blob/main/Different_Type_of_tokenization.ipynb>

1. Word Tokenization:

* Splits text into individual words.
* Simple and intuitive.
* Example: "Hello, world!" → ["Hello", ",", "world", "!"]

1. Subword Tokenization:

* Splits text into subwords or morphemes.
* Useful for handling out-of-vocabulary words and reducing vocabulary size.
* Techniques include Byte-Pair Encoding (BPE) and WordPiece.
* Example: "unhappiness" → ["un", "happiness"] or ["un", "##happy", "##ness"]

1. Character Tokenization:

* Splits text into individual characters.
* Useful for languages with a large number of unique characters.
* Example: "Hello" → ["H", "e", "l", "l", "o"]

1. Sentence Tokenization:

* Splits text into individual sentences.
* Useful for tasks involving sentence-level processing.
* Example: "Hello world. How are you?" → ["Hello world.", "How are you?"]

**Tokenization in LLM?**

**……**

**What is the library for tokenization?**

**…….**

**Example of tokenization:**

**……..**

**How Can give input to LLM?**

**…..**

**Example of giving input**

**…….**

**What is prompt engineering?**

**……..**

**Example of Prompt**

**……..**

**Fine-tuning LLM**

**……….**