Prompt Engineering

Class for ARC 101

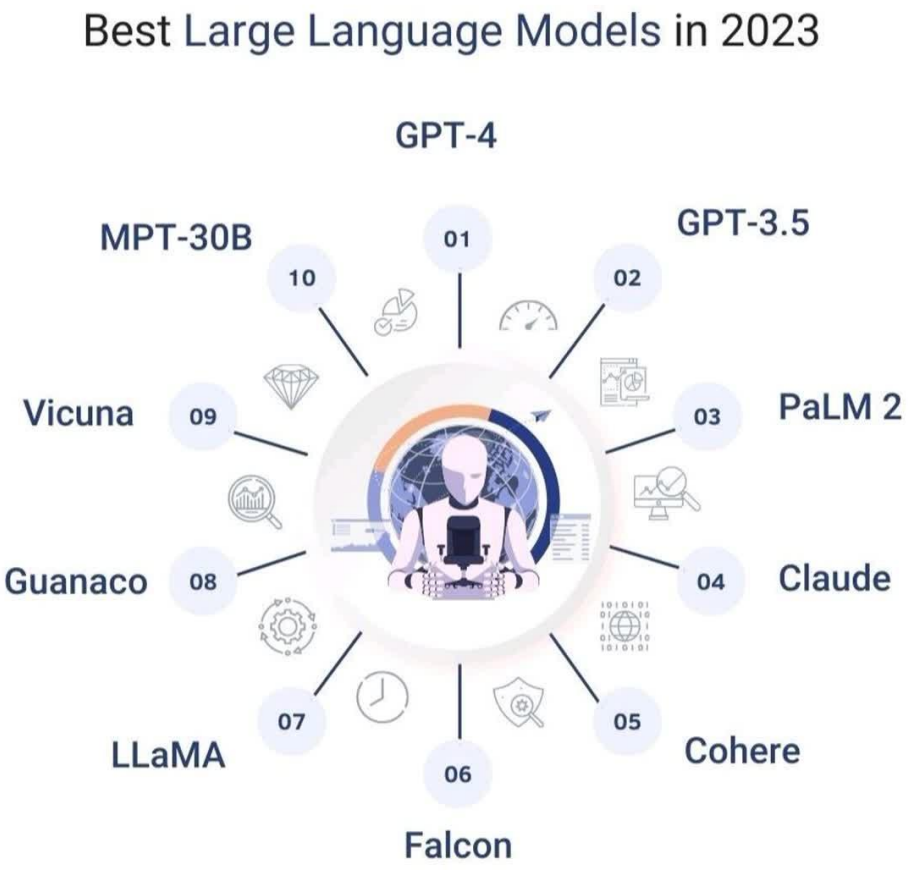
Prepared by Kaveh Karimadini

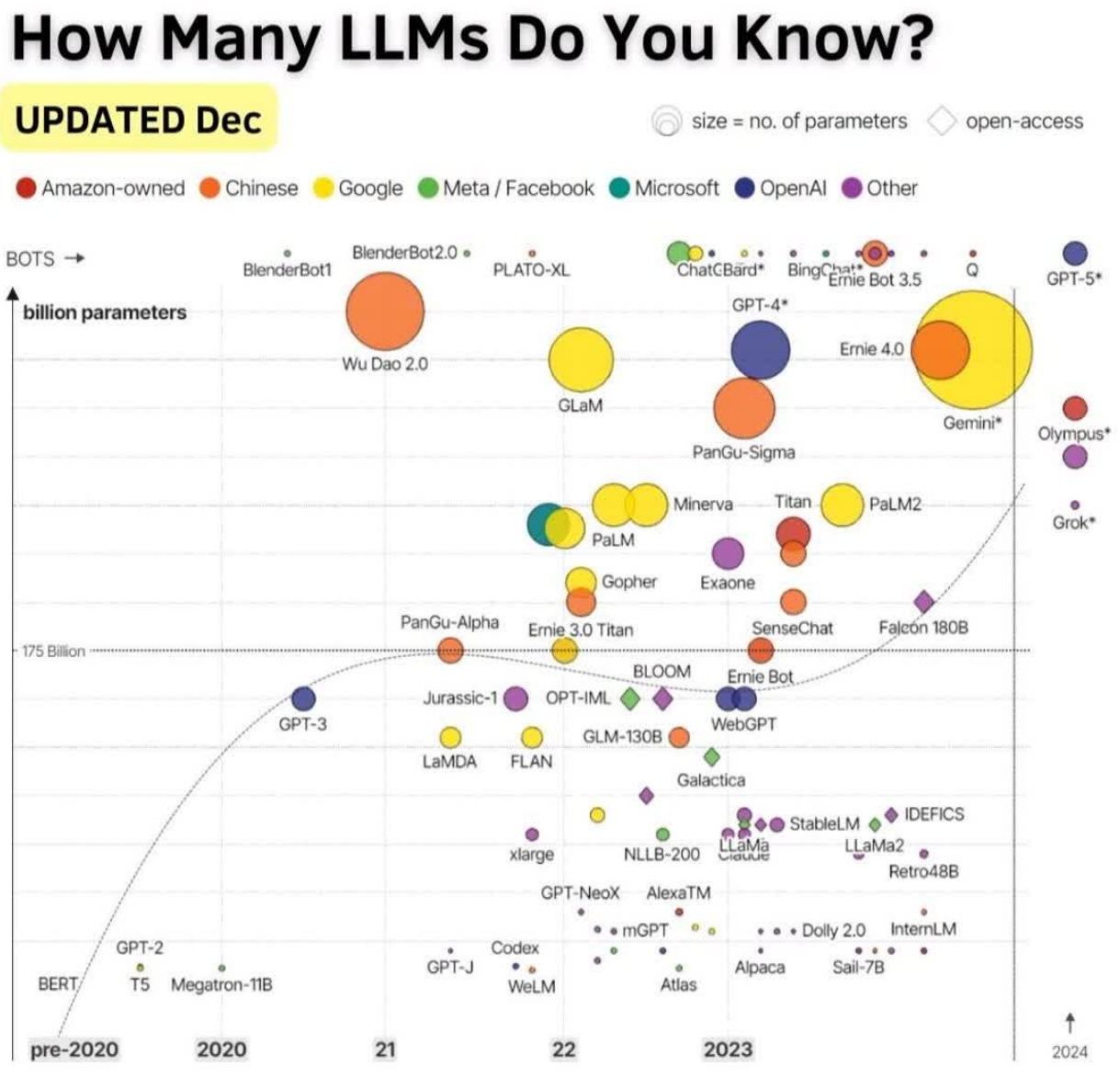
OVERVIEW & PURPOSE

Prompt engineering is the process of **guiding generative artificial intelligence** (AI) to **produce desired outputs** by designing and refining prompts, which are questions or instructions that steer an AI model towards a specific output.

Course Overview

1. An overview on LLM (Large language models)
2. Most common prompt styles used in ChatGPT
3. Hugging Face🤗

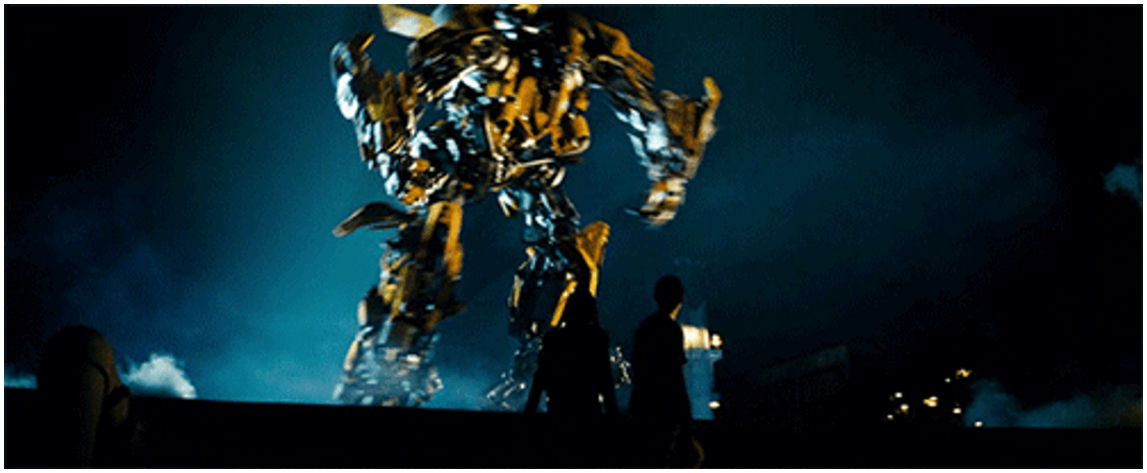




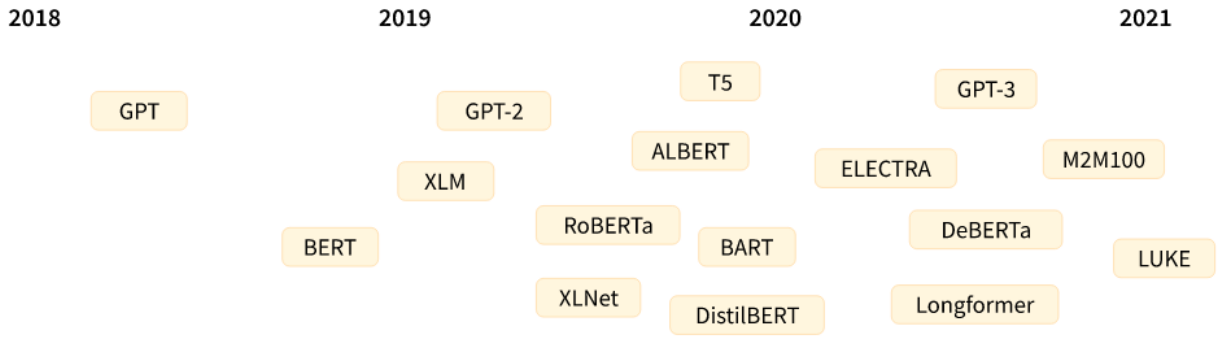
LLM

1. November 30th, 2022, marks a significant chapter in the History of machine learning. It was the day OpenAI released ChatGPT, setting a new benchmark for chatbots powered by Large Language Models
2. Use cases of LLMs:
    a. **Text Summarization**: These models are able to perform a summarization of large texts, including legal texts, reviews, dialogues, among many others.

b. **Sentiment Analysis**: They can read through reviews of products and services and classify them as positive, negative, or neutral. These can also be used in Finance to see if the general public feels Bullish or Bearish on certain securities.
c. **Language Translation**: They can provide real-time translations from one language to another.

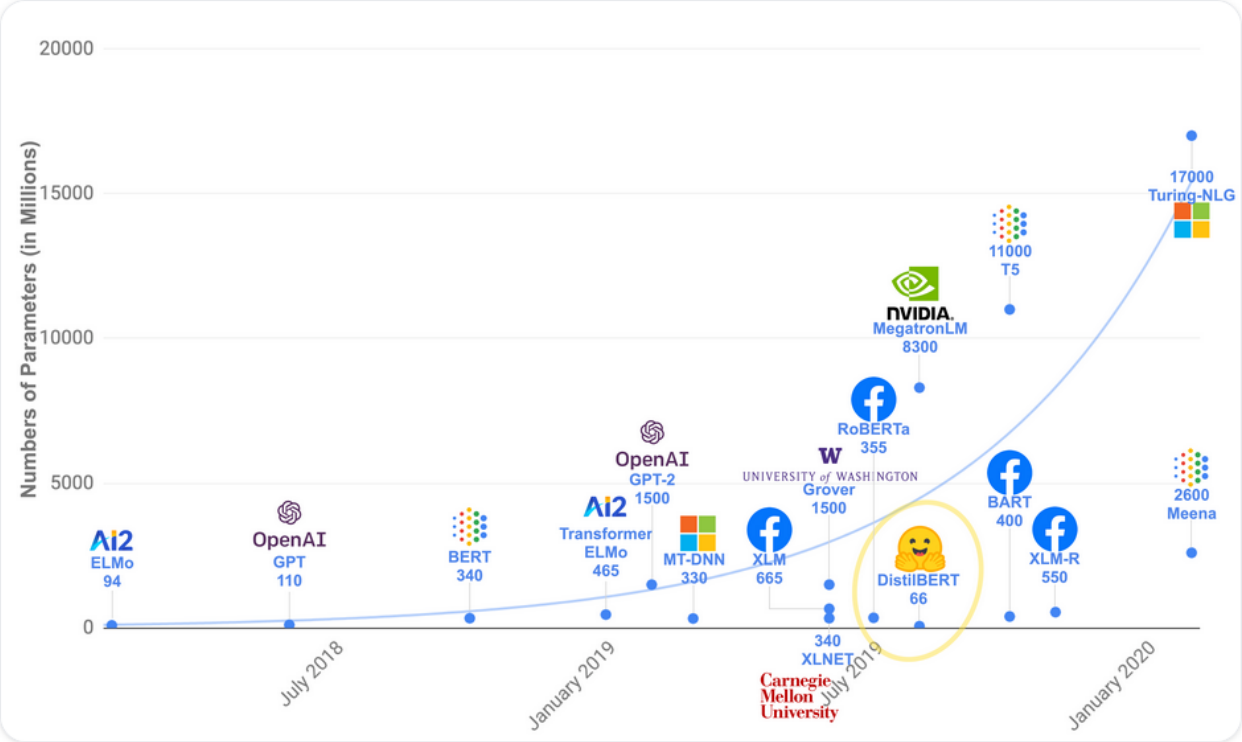


3. ChatGPT Architecture comes from the **transformer**.



4. The Transformer architecture was introduced in June 2017. The focus of the original research was on translation tasks. This was followed by the introduction of several influential models, including:
    a. **June 2018**: GPT, the first pretrained Transformer model, used for fine-tuning on various NLP tasks and obtained state-of-the-art results
    b. **October 2018**: BERT, another large pretrained model, this one designed to produce better summaries of sentences (more on this in the next chapter!)

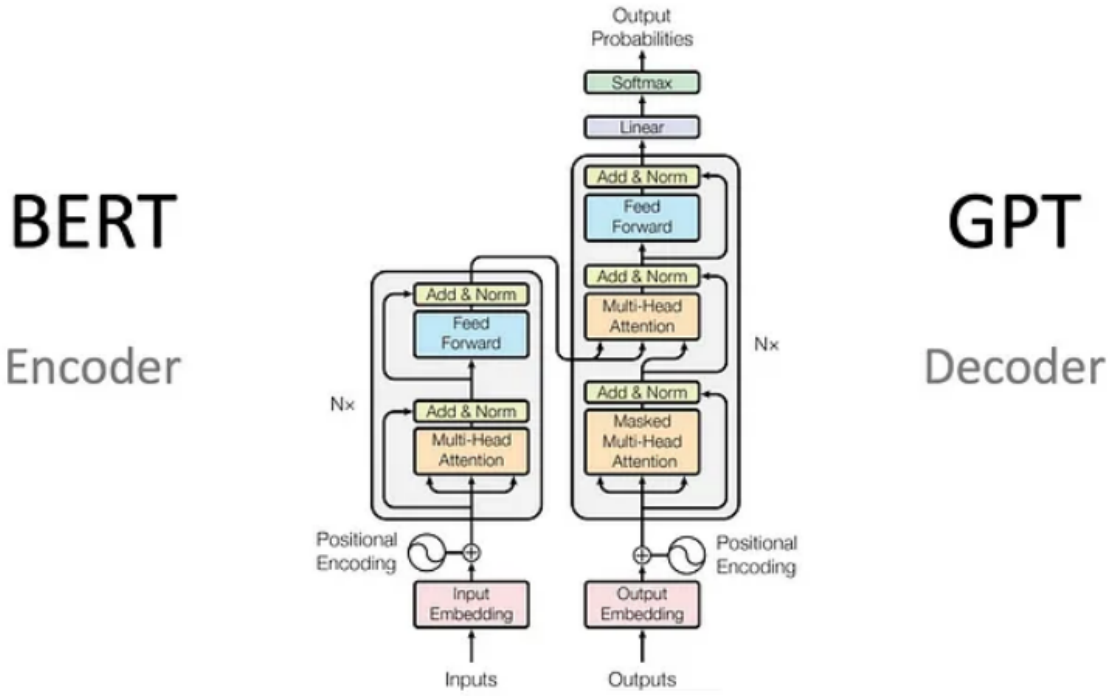
c. **February 2019**: GPT-2, an improved (and bigger) version of GPT that was not immediately publicly released due to ethical concerns
d. **October 2019**: DistilBERT, a distilled version of BERT that is 60% faster, 40% lighter in memory, and still retains 97% of BERT's performance
e. **October 2019**: BART and T5, two large pretrained models using the same architecture as the original Transformer model (the first to do so)
f. **May 2020**, GPT-3, an even bigger version of GPT-2 that is able to perform well on a variety of tasks without the need for fine-tuning (called *zero-shot learning*)



Broadly, they can be grouped into three categories:

1) GPT-like (also called *auto-regressive* Transformer models)
2) BERT-like (also called *auto-encoding* Transformer models)
3) BART/T5-like (also called *sequence-to-sequence* Transformer models)

The paper: Google's 2017 Attention is All You Need, the Transformer architecture was introduced to the world:



Model	Examples	Tasks
Encoder	ALBERT, BERT, DistilBERT, ELECTRA, RoBERTa	Sentence classification, named entity recognition, extractive question answering
Decoder	CTRL, GPT, GPT-2, Transformer XL	Text generation
Encoder-decoder	BART, T5, Marian, mBART	Summarization, translation, generative question answering

LLM Visualization (only purpose is for the embedding and normalization steps)

Key notes:

➔ The importance of providing **Context** when using ChatGPT to facilitate meaningful conversations:

- Context helps ChatGPT understand the background, purpose, and constraints of the conversation, which leads to more accurate responses.
- Three tips for providing context: **introducing the topic, continuing the conversation, and clarifying parameters.**
- Example:
  "Task: Explain the process of photosynthesis."
  “Context: Photosynthesis is a crucial process in our ecosystem, as it allows plants and other photosynthetic organisms to convert sunlight into chemical energy. Please provide a detailed explanation of the process, including the role of chlorophyll, the energy sources involved, and the products formed.”
  - This prompt follows the best practices for providing context, introducing the topic, and clarifying parameters, making it easier for ChatGPT to understand the specific task and generate a relevant response.

➔ **Specificity** is important to ensure that ChatGPT accurately grasps the user's intent and generates relevant responses.

- Three tips for writing specific prompts: **clear questions, detail-oriented requests, and examples and scenarios**.
  - Example:
    "Task: Write a Python function that takes a list of numbers as input and returns the sum of all the even numbers in the list. Please provide the function signature, input parameters, expected output, and a brief example of how the function should be used."
    - This prompt follows the best practices for coding prompts, including being clear, specific, and providing detailed instructions for the task. It specifies the programming language, the function's requirements, and the expected output, allowing ChatGPT to understand the specific coding task and generate a relevant response.

DIFFERENT PROMPT FORMATS AND STYLES in ChatGPT:

- ★ Direct Questions:
  - Direct questions are a common prompt format that allows you to guide Chat GPT to provide a focused answer.
  - They are particularly useful when seeking factual information, explanations, or opinions on a specific topic.
  - To receive the most accurate and relevant responses, ensure that the question is clear and unambiguous.
  - Example: "What are the main causes of climate change?"

- ★ Instructional Prompts:
  - Instructional prompts provide guidance and instructions to Chat GPT, directing the conversation in a specific direction, solving problems, or engaging in creative exercises.
  - They work well when you want Chat GPT to follow a specific process or provide step-by-step instructions.
  - Example: "Please explain the steps involved in setting up a basic website using HTML and CSS."
- ★ Scenario-Based Prompts:
  - Scenario-based prompts involve presenting a hypothetical situation or scenario to Chat GPT, encouraging creative thinking and providing insights based on the given context.
  - They are effective when you want Chat GPT to generate imaginative responses, explore possibilities, or provide creative suggestions.
  - Example: "You are a time traveler from the future. Describe a world-altering invention that will shape the future of humanity."
- ★ Conversational Prompts:
  - Conversational prompts aim to establish a conversational tone with Chat GPT, treating it as a partner in the dialogue.
  - By framing prompts as part of an ongoing conversation, you maintain continuity and provide Chat GPT with contextual information from previous prompts.
  - Conversational prompts can be useful for building rapport, exploring complex topics, and seeking follow-up responses.

- Example: "Based on what we discussed earlier about renewable energy, what are the potential challenges in implementing widespread solar power adoption?"

★ Comparative Prompts:

- Comparative prompts involve presenting two or more options, choices, or scenarios to Chat GPT, encouraging it to compare and contrast the given options, offer opinions, or analyze the pros and cons.
- They are useful when seeking evaluations, preferences, or insights on different alternatives.
- Example: "Compare the advantages and disadvantages of traditional classroom learning versus online education."

★ Fill-in-the-Blank Prompts:

- Fill-in-the-blank prompts provide a partially completed sentence or phrase for Chat GPT to complete.
- This format engages Chat GPT in a more interactive manner and encourages it to generate text that fits the given context.
- They are useful for creative writing exercises, completion tasks, or collaborative storytelling.
- Example: "The key to success is _____."

Hugging Face 🤗

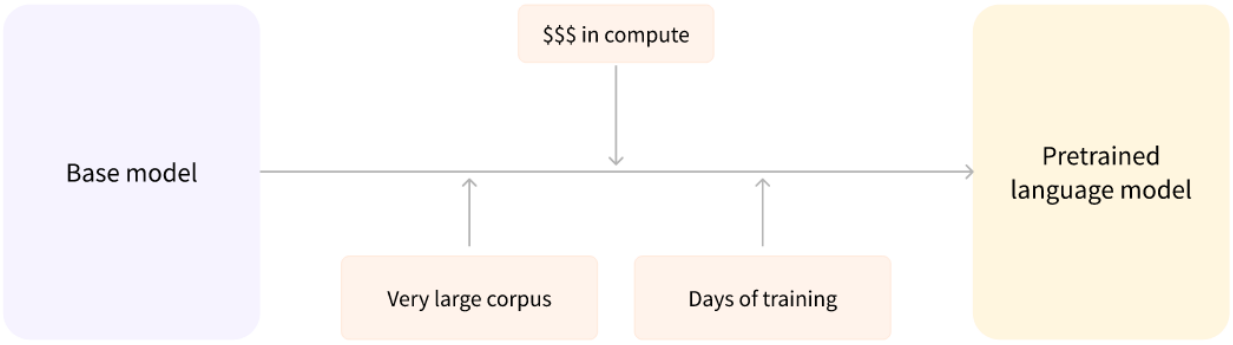
➢ Hugging Face is a platform that provides a user-friendly library called Transformers, designed to be accessible to both beginners and experts in the field of natural language processing (NLP)

- The platform offers a wide range of resources, including:
  - Pre-trained models, such as BERT and GPT, that can be fine-tuned for specific tasks
  - A pipeline architecture that simplifies the process of building and fine-tuning models
  - A tokenizer, which is a tool that converts text into a sequence of tokens, a fundamental step in processing text data

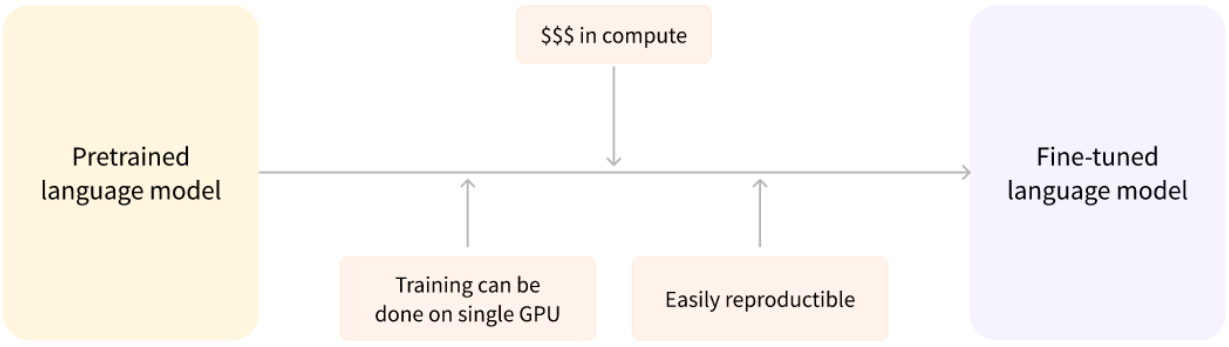
- A community platform where researchers and developers can share their models, datasets, and applications
- https://huggingface.co/docs/transformers/installation

Why Hugging-face?

1) *Pretraining* is the act of training a model from scratch: the weights are randomly initialized, and the training starts without any prior knowledge.



2) The fine-tuning will only require a limited amount of data: the knowledge the pretrained model has acquired is "transferred," hence the term *transfer learning*.



Types of models

➢ The majority of modern LLMs are decoder-only transformers. Some examples include: LLaMA, Llama2, Falcon, GPT2. However, you may encounter encoder-decoder transformer LLMs as well, for instance, Flan-T5 and BART.

➢ Encoder-decoder-style models are typically used in generative tasks where the output **heavily** relies on the input, for example, in translation and summarization. The decoder-only models are used for all other types of generative tasks.

➢ When using a pipeline to generate text with an LLM, it's important to know what type of LLM you are using, because they use different pipelines.

currently available pipelines:

- `feature-extraction` (get the vector representation of a text)
- `Fill-mask:`
    - The idea of this task is to fill in the blanks in a given text.

```
# top_k argument controls how many possibilities you want to be displayed.
Note that here the model fills in the special <mask> word, which is often
referred to as a mask token.
unmasker = pipeline("fill-mask")
unmasker("This course will teach you all about <mask> models.", top_k=2)
```

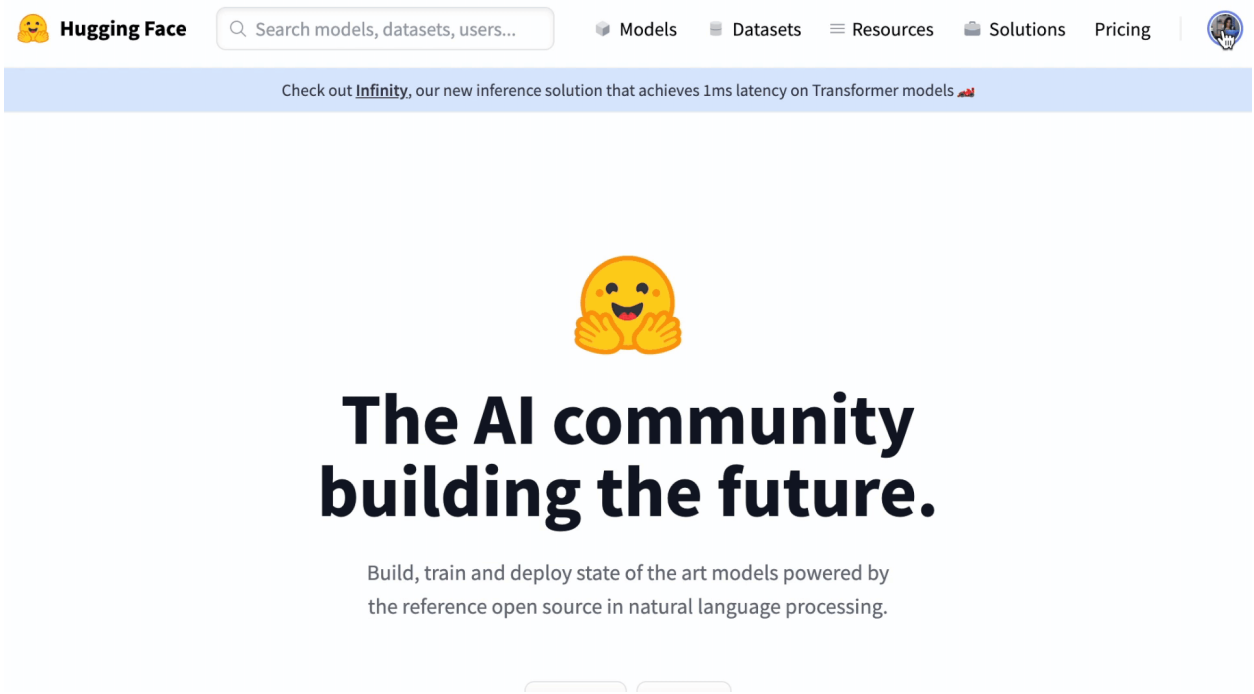
- `ner` (named entity recognition):
    - Named entity recognition (NER) is a task where the model has to find which parts of the input text correspond to entities such as persons, locations, or organizations.

- `Question-answering:`

    - a question-answering pipeline answers questions using information from a given context.

```
question_answerer(
    question="Where do I work?",
    context="My name is Sylvain and I work at Hugging Face in Brooklyn",
)
```

- `Sentiment-analysis:`
- `Summarization:`
    - Summarization is the task of reducing a text into a shorter text while keeping all (or most) of the important aspects referenced in the text.
- `Text-generation:`
    - The main idea here is that you provide a prompt and the model will auto-complete it by generating the remaining text.
    - This is similar to the predictive text feature that is found on many phones.
    - Note: You can control how many different sequences are generated with the argument `num_return_sequences` and the total length of the output text with the argument `max_length`.
- `Translation:`
    - For translation, you can use a default model if you provide a language pair in the task name (such as "translation_en_to_fr"), but the easiest way is to pick the model you want to use on the Model Hub.
- `Zero-shot-classification:`
    - Challenge: we need to classify texts that haven't been labeled.
    - This is a common scenario in real-world projects because annotating text is usually time-consuming and requires domain expertise.
    - Solution: Use zero-shot-classification pipeline is very powerful: it allows you to specify which labels to use for the classification, so you don't have to rely on the labels of the pretrained model.
    - This pipeline is called *zero-shot* because you don't need to fine-tune the model on your data to use it. It can directly return probability scores for any list of labels you want!



Reference:

https://huggingface.co/learn/nlp-course/chapter0/1

Let's Do Some Coding🤩