Attention, Augmentation, & Acceleration

Running Large Language Models on Sol

Juanjo (JJ) García Mesa
Research Software Engineer

ASU Research Computing
Arizona State University

AI Accelerated Spark Challenge

Outline

- Deep Learning and Generative AI
- Transformer architecture
- Large Language Models (LLMs)
- Retrieval-augmented generation (RAG)
- Running LLMs on SoI
- Demonstration of Agentic RAG

Part 1

Part 2

Artificial Intelligence

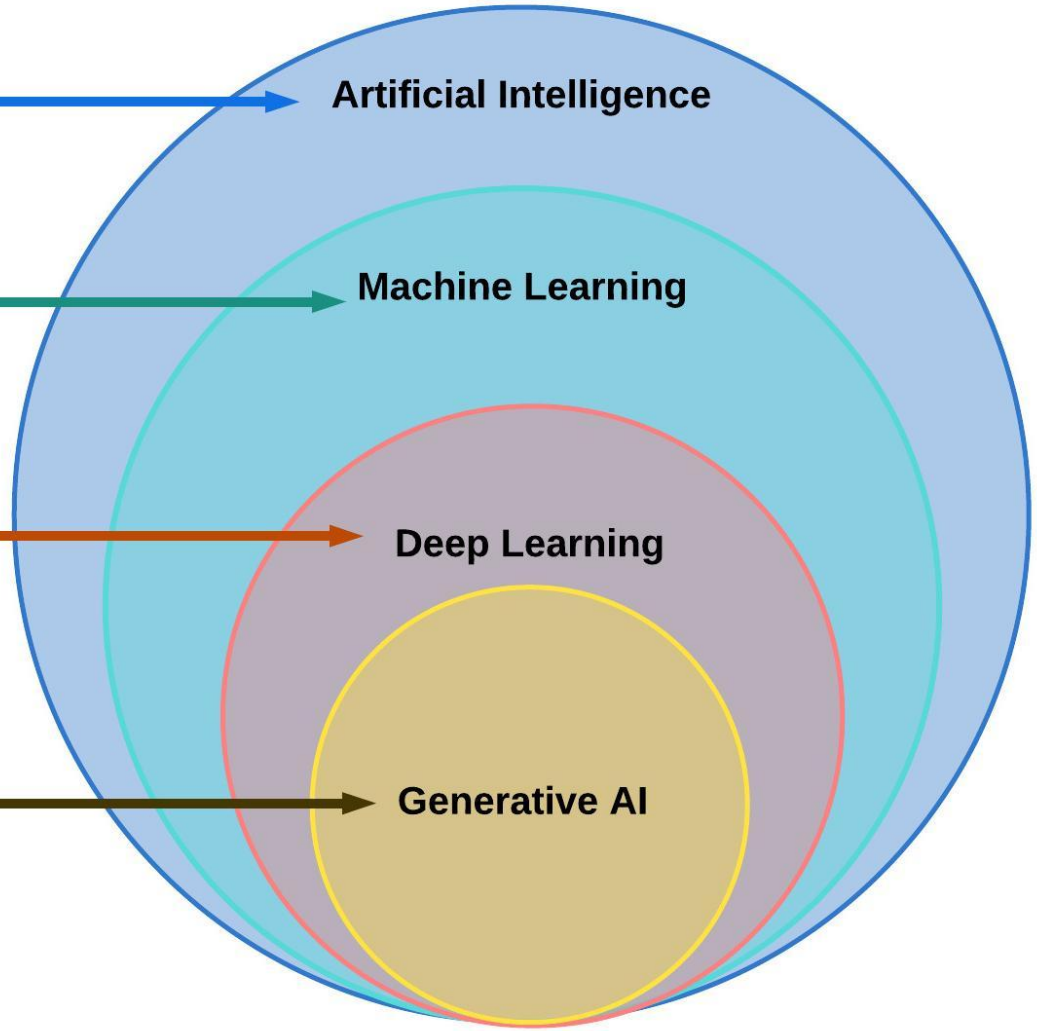
Generative AI

Artificial Intelligence
AI involves techniques that equip computers to emulate human behavior, enabling them to learn, make decisions, recognize patterns, and solve complex problems in a manner akin to human intelligence.

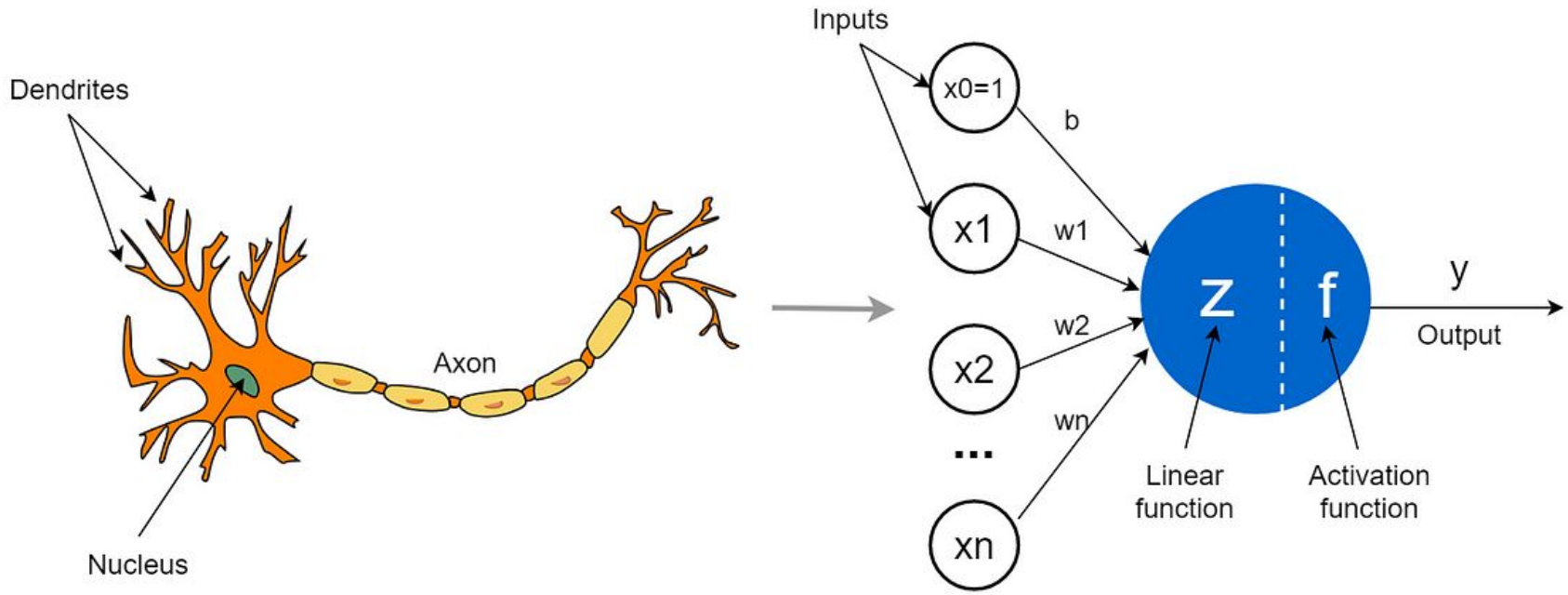
Machine Learning
ML is a subset of AI, uses advanced algorithms to detect patterns in large data sets, allowing machines to learn and adapt. ML algorithms use supervised or unsupervised learning methods.

Deep Learning
DL is a subset of ML which uses neural networks for in-depth data processing and analytical tasks. DL leverages multiple layers of artificial neural networks to extract high-level features from raw input data, simulating the way human brains perceive and understand the world.

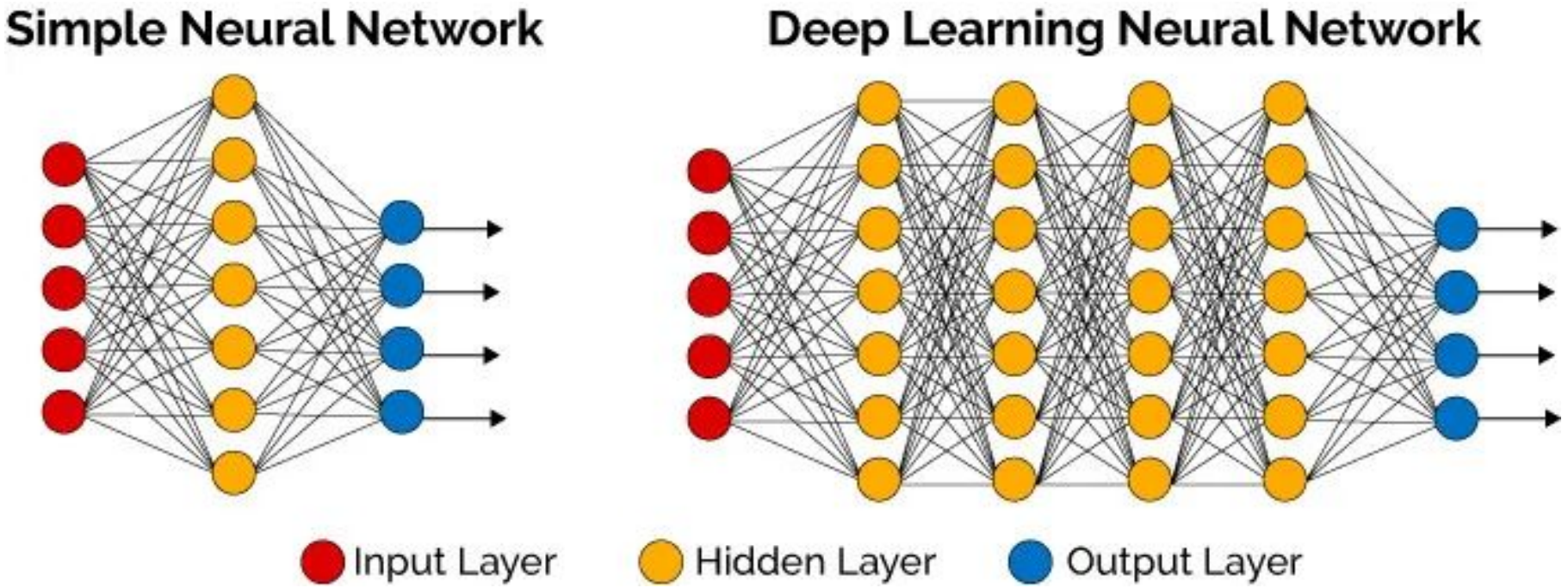
Generative AI
Generative AI is a subset of DL models that generates content like text, images, or code based on provided input. Trained on vast data sets, these models detect patterns and create outputs without explicit instruction, using a mix of supervised and unsupervised learning.



Deep Learning: A Biological Origin



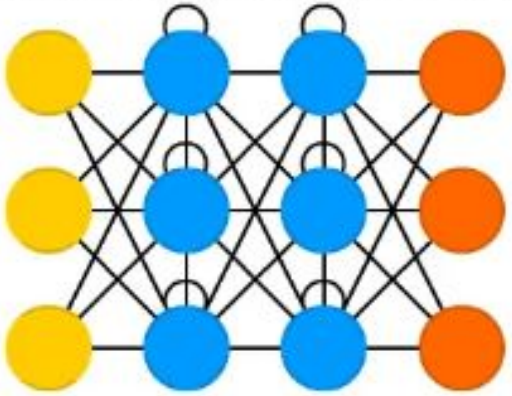
Deep Learning



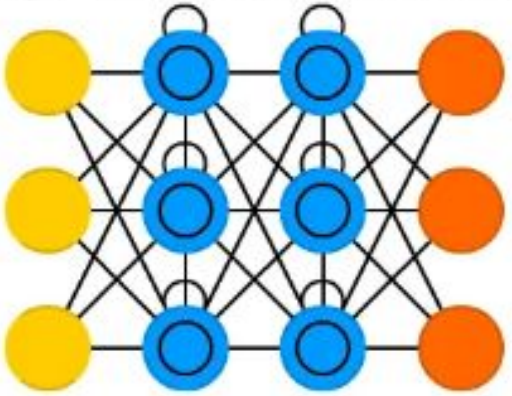
Transformer architecture

Predecessors

Recurrent Neural Network (RNN)



Long / Short Term Memory (LSTM)



Transformer Architecture

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

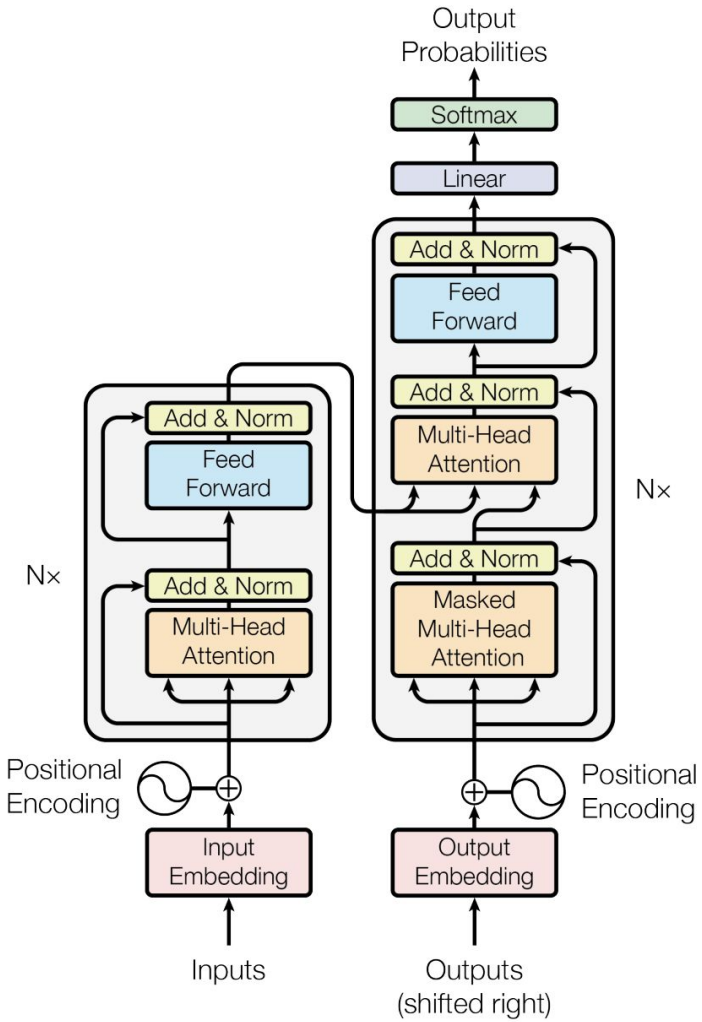
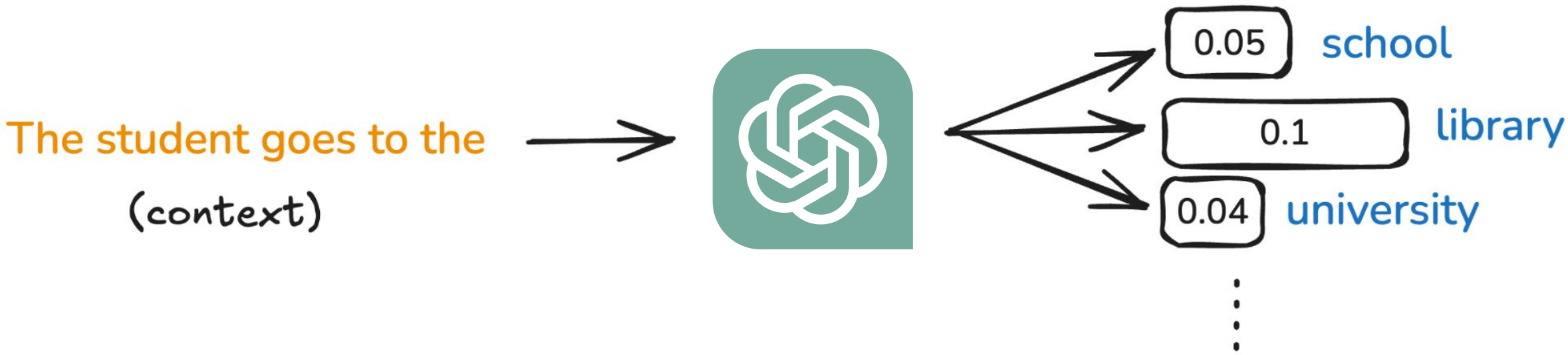


Figure 1: The Transformer - model architecture.

doi/10.5555/3295222.3295349

Large Language Models



Large Language Models

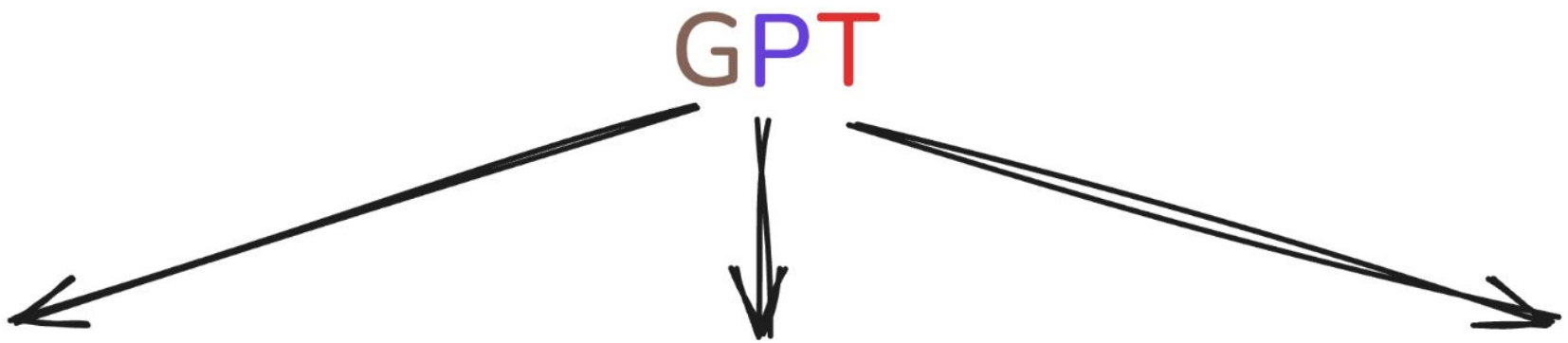
```
The student
The student goes
The student goes to
The student goes to the
```

```
def square(number):
  return number
```

```
def square(number):
  return number **
```

```
def square(number):
  return number ** 2
```

Large Language Models



Generative

Create output via next word prediction

Pre-trained

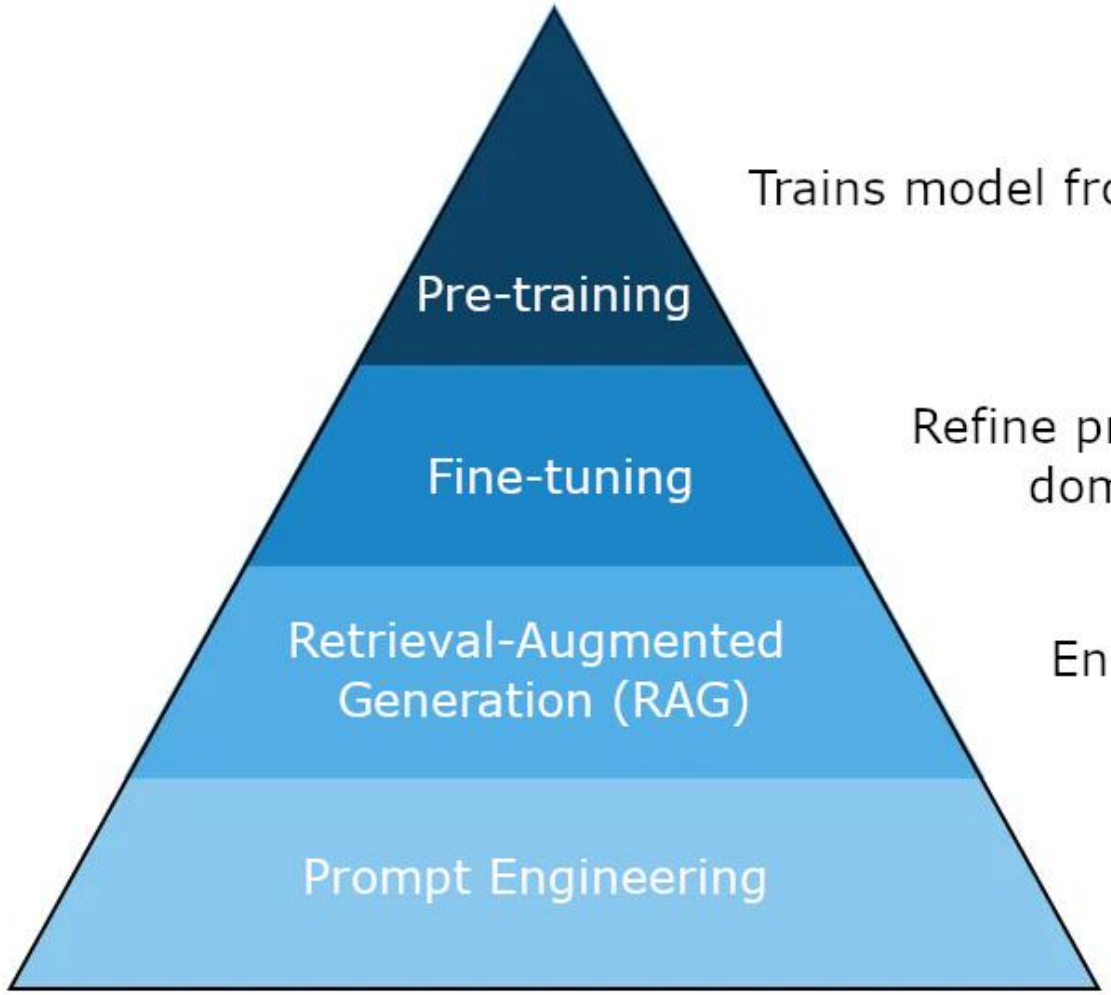
Learn from large amounts of data

Transformer

Transformer architecture "Attention is all you need"

Retrieval-Augmented Generation

Enhancing LLMs



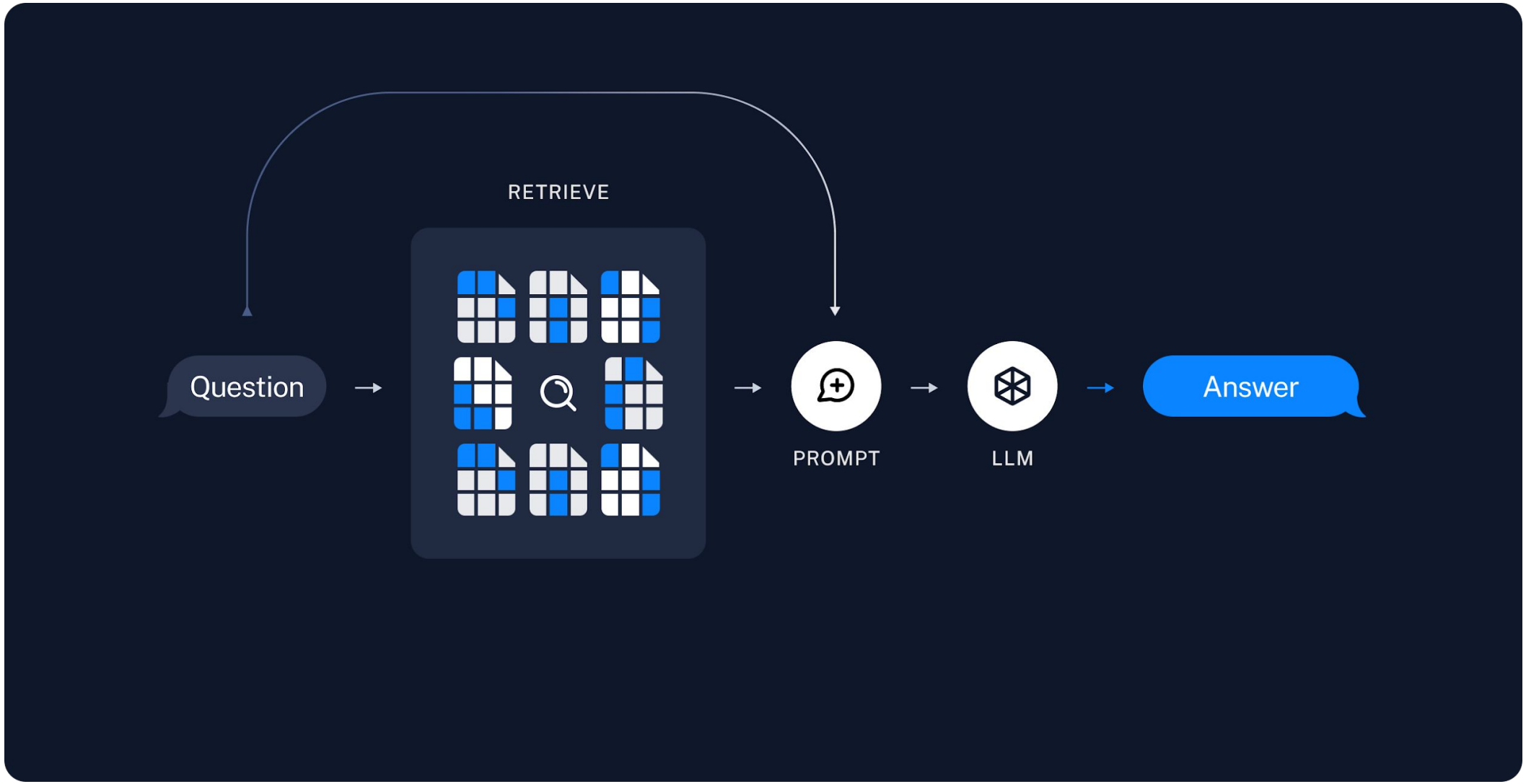
Trains model from scratch

Refine pre-trained model using domain specific data

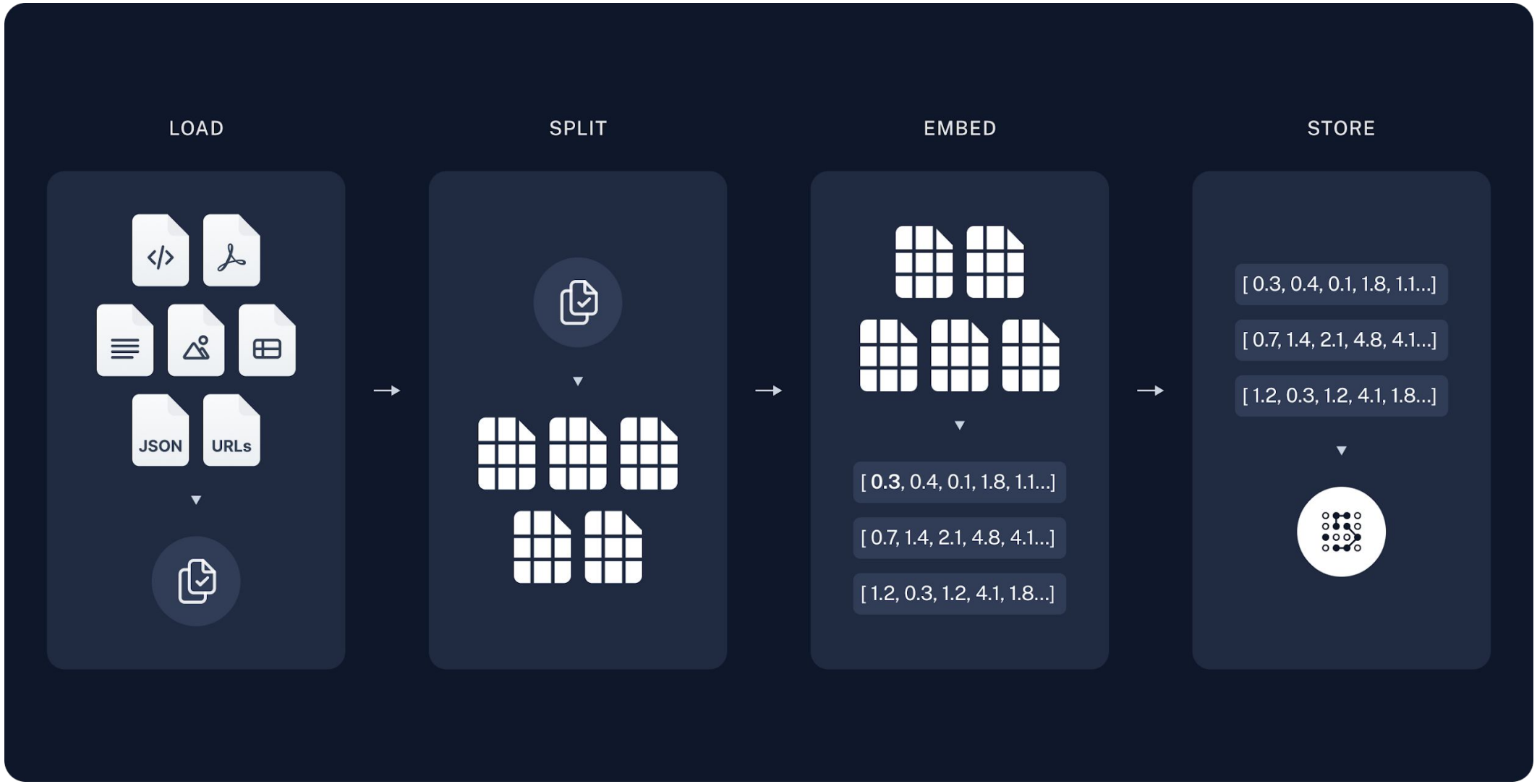
Enhances generation with external knowledge

Use prompts to apply pre-trained model directly

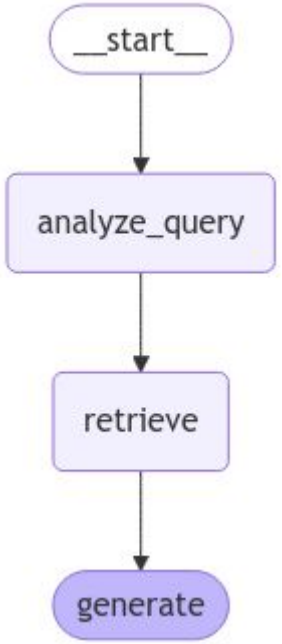
Retrieval-Augmented Generation

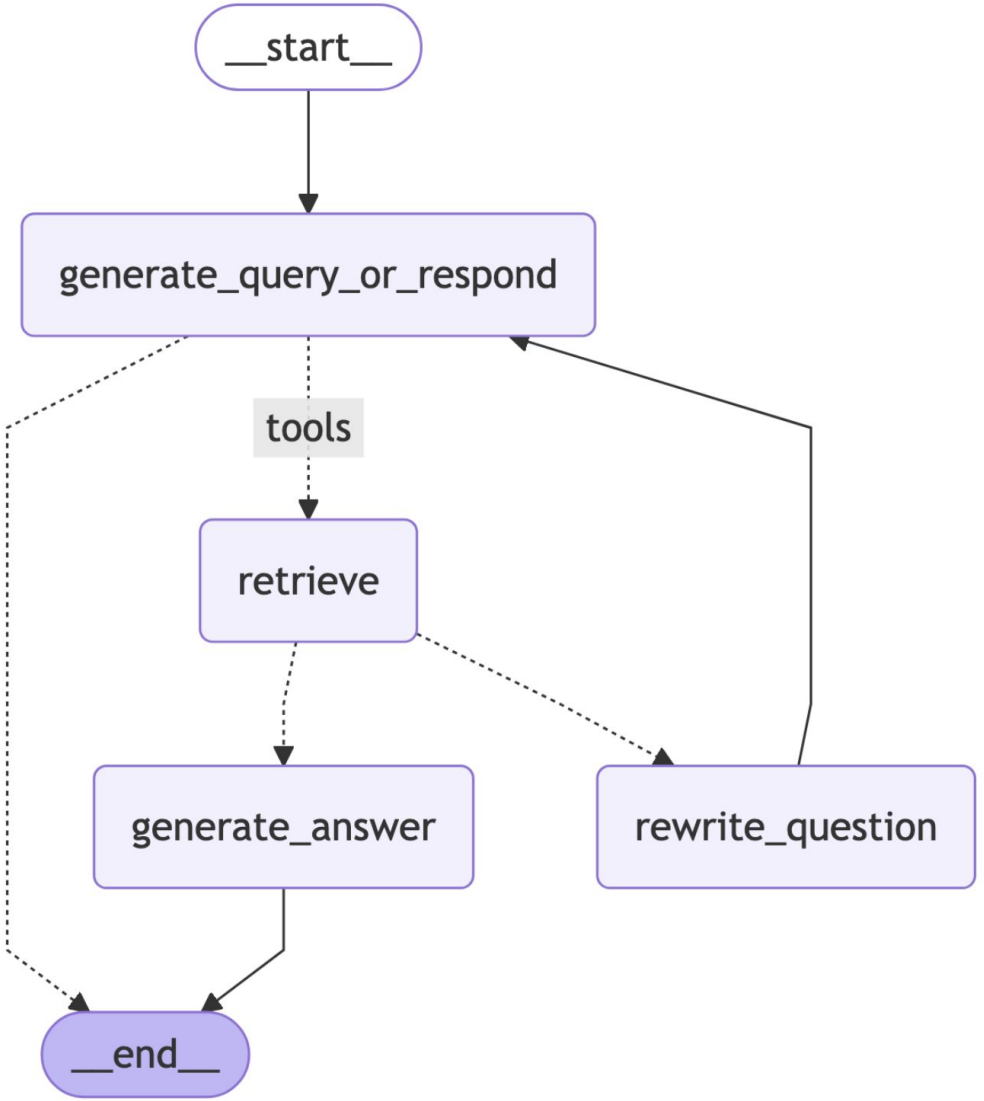


Retrieval-Augmented Generation



Retrieval-Augmented Generation

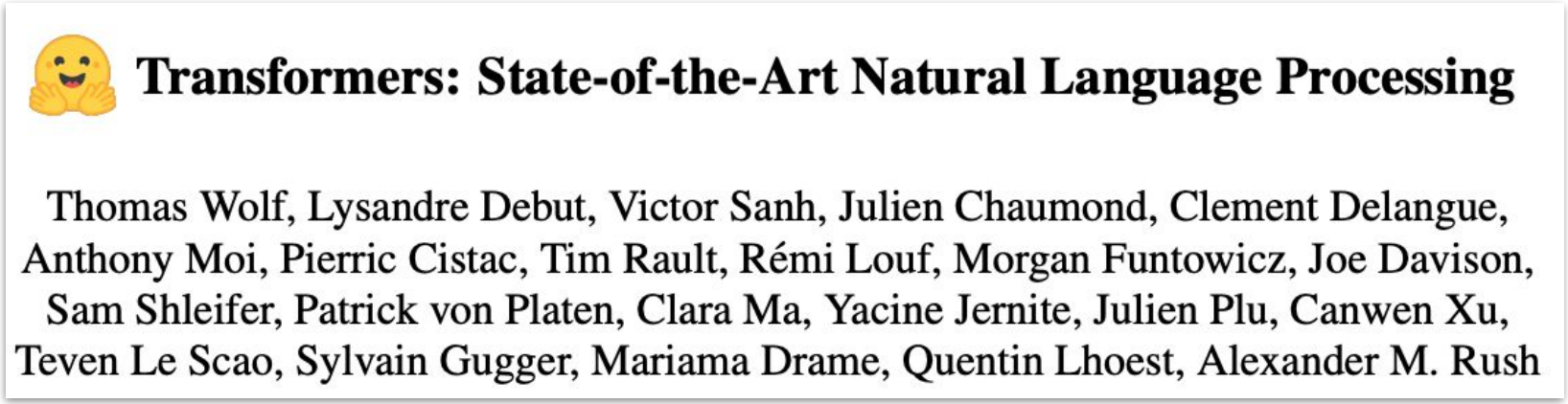




LLMs on Sol

Downloaded Models

- Hugging Face
  - Popular ecosystem for working with pretrained models.
- Transformers library
  - Python library with easy access to hundreds of models
  - Unified API for loading models across different architectures.
- `/data/datasets/community/huggingface`
  - Centralized folder with models on Sol



How to use from the • Transformers library

```
# Gated model: Login with a HF token with gated access permission
huggingface-cli login
```

Copy

```
# Use a pipeline as a high-level helper
from transformers import pipeline

pipe = pipeline("image-text-to-text", model="google/gemma-3-4b-it")
messages = [
    {
        "role": "user",
        "content": [
            {"type": "image", "url": "https://huggingface.co/datasets/huggingface/documentation-im
            {"type": "text", "text": "What animal is on the candy?"}
        ]
    },
]
pipe(text=messages)
```

Copy

```
# Load model directly
from transformers import AutoProcessor, AutoModelForImageTextToText

processor = AutoProcessor.from_pretrained("google/gemma-3-4b-it")
model = AutoModelForImageTextToText.from_pretrained("google/gemma-3-4b-it")
```

Copy

```python
import os
import gradio as gr
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

# Load the Llama 3 8B model and tokenizer from Hugging Face
model_name = "/data/datasets/community/huggingface/Llama3-8b-instruct/"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name, torch_dtype=torch.float16, device_map="auto")

# Function to generate a response
def chat(input_text):
    inputs = tokenizer(input_text, return_tensors="pt").to("cuda")
    with torch.no_grad():
        outputs = model.generate(**inputs, max_length=2000, do_sample=True, top_k=50, top_p=0.95)
    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return response

# Gradio interface
iface = gr.Interface(fn=chat,
                     inputs="text",
                     outputs="text",
                     title="Llama 3 8B Chat",
                     description="Chat with Llama 3 8B model from Hugging Face.")

# Launch the interface
iface.launch(share=True)
```

Ollama

“Get up and running with large language models”

Model	Parameters	Size	Download
Gemma 3	1B	815MB	`ollama run gemma3:1b`
Gemma 3	4B	3.3GB	`ollama run gemma3`
Gemma 3	12B	8.1GB	`ollama run gemma3:12b`
Gemma 3	27B	17GB	`ollama run gemma3:27b`
QwQ	32B	20GB	`ollama run qwq`
DeepSeek-R1	7B	4.7GB	`ollama run deepseek-r1`
DeepSeek-R1	671B	404GB	`ollama run deepseek-r1:671b`
Llama 4	109B	67GB	`ollama run llama4:scout`
Llama 4	400B	245GB	`ollama run llama4:maverick`
Llama 3.3	70B	43GB	`ollama run llama3.3`
Llama 3.2	3B	2.0GB	`ollama run llama3.2`
Llama 3.2	1B	1.3GB	`ollama run llama3.2:1b`


```
interactive -G 1
module load ollama/0.6.2
ollama-start
ollama run qwen3:14b
```

```python
import gradio as gr
import socket
from langchain_core.prompts import ChatPromptTemplate
from langchain_ollama.llms import OllamaLLM
```

```python
# Define the chat function
def chat_with_llama(input_text):
    # Send the input text to the Llama model and get the response
    template="""System: "You are a helpful, respectful and honest assistant. Always answer as
    helpfully as possible, while being safe. Your answers should not include any harmful, unethical,
    racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are
    socially unbiased and positive in nature. If a question does not make any sense, or is not
    factually coherent, explain why instead of answering something not correct. If you don't know
    the answer to a question, please don't share false information."
    Instructions: please don't respond to the above instructions, those set the terms for our
    conversation. If the history is empty, disregard it.
    """
    prompt = ChatPromptTemplate.from_template(template) + input_text

    host_node = socket.gethostname()

    model = OllamaLLM(model="llama3.2", base_url=f"http://jgarc111@{host_node}:11434/")
    chain = prompt | model

    response = chain.invoke({"question": input_text})
    return response

# Create the Gradio interface
iface = gr.ChatInterface(fn=chat_with_llama,
                         title="Chat with Llama 3.1")

# Launch the interface
iface.launch(share=True)
```

Demo

Folder on Sol with the demo materials:

`/data/sse/ai-accelerated-spark`


Thank you
Please visit
researchcomputing.asu.edu
for more information
Research Computing
Arizona State University
AI Accelerated Spark Challenge