Gemma for GST

Fine-tuning Gemma using Ludwig on GST-FAQs dataset

Yogesh Haribhau Kulkarni (PhD)
Published in Google Cloud - Community
9 min read · Just now

Listen Share ••• More

Photo by Nathana Rebouças on Unsplash

Let's dive into the fascinating world of cutting-edge technology as we uncover the extraordinary capabilities of two powerful tools: Gemma, an open-source Large Language Model by Google, and Ludwig, a Declarative Machine Learning framework from Predibase.

Our journey begins with a singular goal: crafting a sophisticated Question Answering model meticulously designed for FAQs (Frequently Asked Questions) on GST (Goods and Services Tax) in India.

Ludwig takes center stage in our endeavor. It empowers us to train machine learning models using the Encoder-Combination-Decoder (ECD) mode and fine-tune Large Language Models (LLMs) via Instruction Tuning mode, all through declarative configuration files.

Before we delve deeper into our quest, let's shed some light on GST (Goods and Services Tax) in India. GST represents a monumental shift in India's taxation landscape, replacing a myriad of previous taxes such as service tax, central excise duty, VAT, and more. It serves as a unified tax structure, simplifying the tax process and streamlining operations across the nation. However, this transition from a multi-tax system to a single-tax regime has inevitably sparked numerous inquiries. These queries, along with their corresponding answers, are compiled into FAQs, forming the foundation of our project. By leveraging machine learning models or finely-tuned LLMs, we aim to develop a chatbot-like application capable of addressing these FAQs with precision and efficiency.

Here is a step-by-step breakdown of the solution.

Installation

First things first, ensure you have the essentials: a HuggingFace API Token, access approval to `Gemma-7b-it` , and a GPU boasting a minimum of 12 GiB of VRAM. For this tutorial, we'll be utilizing the T4 GPU.

To streamline our setup, let's execute a few commands to ensure everything is in place. We'll start by uninstalling TensorFlow and then proceed to install `Cython` . It's essential to install `Cython` before Torch to avoid any potential " `_C` " errors. Next up, we'll install Ludwig, our trusty Declarative Machine Learning framework, followed by the Ludwig Large Language Model (LLM) extension. Additionally, we'll install Accelerate to optimize our performance.

Now, let's take advantage of the `accelerate` package by configuring it for mixed precision, enhancing our GPU utilization. Once that's done, we'll install the latest version of `bitsandbytes` to facilitate our workflow. It's worth noting that for seamless integration with Ludwig, we recommend using `bitsandbytes` version 0.41.3 or higher to support 4-bit converted models.

```
!pip uninstall -y tensorflow --quiet
!pip install Cython # do this before installing torch which is inside ludwig, t
!pip install ludwig
!pip install ludwig[llm]
!pip install accelerate
from accelerate.utils import write_basic_config; write_basic_config(mixed_preci
!pip install -i https://pypi.org/simple/ bitsandbytes  # latest
```

You may need to RFRESH once, but in the next run, the installation goes smooth.

Obtain a HuggingFace API Token and request access to gemma-7b-it before proceeding. You may need to signup on HuggingFace if you don't aleady have an account: https://huggingface.co/join

```
import getpass
# import locale; locale.getpreferredencoding = lambda: "utf-8"
import logging
import os
import torch
import yaml

from ludwig.api import LudwigModel


os.environ["HUGGING_FACE_HUB_TOKEN"] = getpass.getpass("Token:")
assert os.environ["HUGGING_FACE_HUB_TOKEN"]
```

Dataset

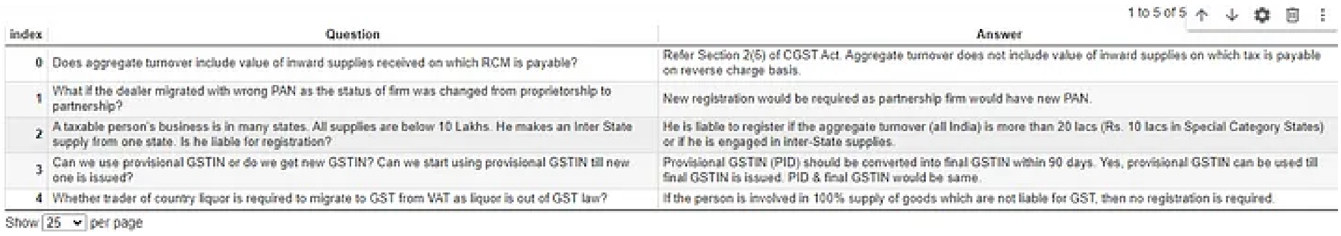
Let's gather our dataset to fuel our machine learning endeavors. You can access the data in CSV format from the GitHub location.

To streamline the process, you can use the `wget` command to download the dataset directly from the provided location. Once downloaded, ensure to place the dataset in a designated `data` folder for easy access.

```
!pip install wget
import wget

# Replace the URL with the raw URL of the file on GitHub
url = "https://raw.githubusercontent.com/yogeshhk/Sarvadnya/master/src/ludwig/c

# Download the file
wget.download(url, 'cbic-gst_gov_in_fgaq.csv')
```

After downloading the dataset and organizing it in the specified folder, you can comment out this cell to proceed with further executions seamlessly.

```
from google.colab import data_table; data_table.enable_dataframe_formatter()
import numpy as np; np.random.seed(123)
import pandas as pd

df = pd.read_csv('cbic-gst_gov_in_fgaq.csv', encoding='cp1252')
df.head()
```



1 to 5 of 5

index	Question	Answer
0	Does aggregate turnover include value of inward supplies received on which RCM is payable?	Refer Section 2(6) of CGST Act. Aggregate turnover does not include value of inward supplies on which tax is payable on reverse charge basis.
1	What if the dealer migrated with wrong PAN as the status of firm was changed from proprietorship to partnership?	New registration would be required as partnership firm would have new PAN.
2	A taxable person's business is in many states. All supplies are below 10 Lakhs. He makes an Inter State supply from one state. Is he liable for registration?	He is liable to register if the aggregate turnover (all India) is more than 20 lacs (Rs. 10 lacs in Special Category States) or if he is engaged in inter-State supplies.
3	Can we use provisional GSTIN or do we get new GSTIN? Can we start using provisional GSTIN till new one is issued?	Provisional GSTIN (PID) should be converted into final GSTIN within 90 days. Yes, provisional GSTIN can be used till final GSTIN is issued. PID & final GSTIN would be same.
4	Whether trader of country liquor is required to migrate to GST from VAT as liquor is out of GST law?	If the person is involved in 100% supply of goods which are not liable for GST, then no registration is required.

Show 25 per page

An essential phase of our journey entails assembling a dataset that accurately reflects the real-world inquiries faced by taxpayers. This dataset is specifically designed for Question Answering purposes. Each entry in the dataset comprises:

- A question detailing a specific query
- An answer corresponding to the question

Configuration

Let’s define the configuration for Instruction Fine Tuning using the Gemma 7B model, a crucial step that lays the foundation for our machine learning journey. Drawing inspiration from this tutorial, we’re customizing our prompt to suit our specific needs and objectives.

In this configuration, we’ll specify the parameters and settings required to fine-tune the Gemma 7B model, leveraging its vast capabilities to achieve our desired outcomes. Instruction Fine Tuning allows us to tailor the model’s behavior and performance to address our unique requirements effectively.

```
instruction_tuning_yaml = yaml.safe_load("""
model_type: llm
base_model: google/gemma-7b-it

quantization:
 bits: 4

adapter:
 type: lora

prompt:
  template: |
    ### Instruction:
    You are a taxation expert on Goods and Services Tax used in India.
    Take the Input given below which is a Question. Give Answer for it as a Res

    ### Input:
    {Question}

    ### Response:

input_features:
 - name: Question
   type: text
   preprocessing:
      max_sequence_length: 1024

output_features:
 - name: Answer
   type: text
   preprocessing:
      max_sequence_length: 384

trainer:
  type: finetune
  epochs: 8
  batch_size: 1
  eval_batch_size: 2
  gradient_accumulation_steps: 16  # effective batch size = batch size * gradie
  learning_rate: 2.0e-4
  enable_gradient_checkpointing: true
  learning_rate_scheduler:
    decay: cosine
    warmup_fraction: 0.03
    reduce_on_plateau: 0

generation:
  temperature: 0.1
  max_new_tokens: 512

backend:
 type: local
""")
```

This Ludwig configuration defines a fine-tuning process for a large language model (LLM) called Gemma, specifically the `google/gemma-7b-it` version. Here's a breakdown of the key parts:

Model and Quantization:

- It sets the `model_type` to `llm` indicating a large language model.
- The `base_model` is set to `google/gemma-7b-it`, which is likely a pre-trained Gemma model with 7.0 billion parameters.
- Quantization with 4 bits is enabled (`quantization`), which aims to reduce model size and improve efficiency during training.

Instruction Tuning:

- An instruction template is defined in the `prompt` section. This template will be used to frame user questions (Input) and expected answers (Response) when training the model.

Input and Output Features:

- The configuration defines two features:

- `Question` : This is a text input with a maximum sequence length of 1024 tokens (words or characters).
- `Answer` : This is the model's generated text response, limited to a maximum of 384 tokens.

Training Parameters:

- The `trainer` section defines the fine-tuning process:
- `type: finetune` specifies fine-tuning the pre-trained Gemma model on your specific data.
- Training parameters like epochs, batch size, learning rate, and scheduler are set.
- Gradient accumulation helps train with a larger effective batch size (`batch_size * gradient_accumulation_steps`).

Generation Parameters:

- The `generation` section controls how the model generates text after training:
- `temperature` of 0.1 controls the randomness of the generated text (lower, more likely to pick common words).
- `max_new_tokens` limits the number of tokens the model generates in a single response (here, 512).

Backend:

- The `backend` is set to `local` , indicating the model will be trained on your local machine.

Overall, this configuration takes a pre-trained Gemma model and fine-tunes it on a dataset of questions and answers related to India’s Goods and Services Tax (GST). This will hopefully allow the model to answer user questions about GST in an informative way.

Training

As we proceed with Ludwig, its declarative nature comes into play, offering us a clear and transparent approach to defining the model architecture. This enables us to gain valuable insights throughout the training process.

```
model_instruction_tuning = LudwigModel(config=instruction_tuning_yaml,  logging
results_instruction_tuning = model_instruction_tuning.train(dataset=df)
```

We'll begin by instantiating the `LudwigModel` with the fine-tuning configuration specified in the `instruction_tuning_yaml`. Our training data will be sourced from a CSV-based dataframe focusing on GST-related information. This sets the stage for our model to learn and adapt to the nuances of the dataset, paving the way for robust performance and accurate predictions.

Testing

When it comes to testing or inferencing our dataset, we're dealing with just a handful of questions for which answers are sought. This streamlined approach allows us to efficiently evaluate the performance of our model on specific queries, providing valuable insights into its accuracy and effectiveness.

```
import pandas as pd
test_df = pd.DataFrame([
    {
        "Question": "If I am not an existing taxpayer and wish to newly registe
    },
    {
        "Question": "Does aggregate turnover include value of inward supplies r
    },
])
```

By focusing on a select set of questions, we can thoroughly assess the model's ability to provide accurate responses, ensuring its reliability in real-world scenarios.

With Ludwig's training journey reaching its conclusion, the explorers eagerly put the model to the test. Armed with a set of questions pertaining to GST, they eagerly awaited the model's responses, eager to witness the declarative AI framework in action.

Now, it's time to unveil the predictions generated by our fine-tuned model. Let's delve into the results and see how our model fares in accurately addressing the queries posed to it.

```
predictions_instruction_tuning_df, output_directory = model_instruction_tuning.
print(predictions_instruction_tuning_df["Answer_response"].tolist())
```

The response in the current run is

```
INFO:ludwig.utils.tokenizers:Loaded HuggingFace implementation of google/gemma-
Asking to truncate to max_length but no maximum length is provided and the mode
Prediction: 100%|██████████| 1/1 [00:12<00:00, 12.52s/it]
INFO:ludwig.utils.tokenizers:Loaded HuggingFace implementation of google/gemma-
/usr/local/lib/python3.10/dist-packages/ludwig/features/feature_utils.py:102: R
  return np.sum(np.log(sequence_probabilities))
INFO:ludwig.api:Finished predicting in: 23.29s.
[['\r You will have to apply for registration under GST.'], ['\r\n\n\r\n\nThe R
```

While the answers generated by the model seem reasonably satisfactory, there's always room for improvement. Enhancing the quality of the Large Language Model (LLM), adjusting training parameters, and, most importantly, augmenting the dataset size for fine-tuning are avenues worth exploring.

Expanding the dataset allows the model to capture a wider range of patterns and nuances, thereby improving its ability to provide accurate responses across various scenarios. Additionally, fine-tuning training parameters and optimizing the LLM's architecture can further refine its performance, ensuring more precise predictions.

By iteratively refining these aspects, we can elevate the model's capabilities and enhance its effectiveness in addressing a broader spectrum of queries related to GST. Let's continue to explore and innovate, striving for excellence in the realm of machine learning and natural language processing.

Conclusions

The fine-tuned model has yielded promising results, showcasing its potential in addressing queries related to GST with reasonable accuracy. Ludwig's declarative approach emerges as a beacon of clarity and efficiency in the realm of machine learning model development. Its methodical framework offers a clear and concise methodology, streamlining the process of building and fine-tuning models.

One of the standout features of Ludwig is its flexibility, allowing seamless transitions between different approaches and base Large Language Models (LLMs). This versatility empowers researchers and practitioners to experiment with various configurations and architectures, facilitating iterative improvements and optimizations.

With Ludwig, the complexities of training and fine-tuning models are demystified, enabling practitioners to focus on the task at hand without getting bogged down by technical intricacies. As we continue to leverage Ludwig's capabilities, we unlock new possibilities and push the boundaries of what's achievable in the field of natural language processing.

In essence, Ludwig stands as a testament to the power of declarative machine learning, offering a reliable and intuitive platform for tackling the challenges of complex domains.

References

Here are some valuable resources to further enhance your understanding and proficiency in fine-tuning Gemma-7B and exploring the capabilities of Ludwig:

1. How to Efficiently Fine-Tune Gemma-7B with Open-Source Ludwig: This blog post provides insights and guidelines on efficiently fine-tuning Gemma-7B using Ludwig, offering practical tips and best practices.

2. Fine-tuning Mistral 7B on a Single GPU with Ludwig: Explore this blog post to learn how to fine-tune Mistral 7B on a single GPU using Ludwig, uncovering techniques for optimizing performance and efficiency.

3. Efficient Fine-Tuning for Llama-v2–7b on a Single GPU: Dive into this informative video to discover efficient fine-tuning strategies for Llama-v2–7b on a single GPU, presented in a clear and concise manner.

4. Webinar: 10 Things to Know about LLMs: If you're new to Large Language Models (LLMs), this webinar by Daliana Liu provides valuable insights into the key aspects and considerations to be aware of.

5. Ludwig 0.8 Release Blogpost: Explore the latest features and enhancements introduced in Ludwig 0.8 through this comprehensive blog post, offering a detailed overview of the toolkit's capabilities.

6. Ludwig Documentation: Refer to the Ludwig documentation for comprehensive guidance, tutorials, and reference materials covering all aspects of the toolkit, from installation to advanced usage.

These resources serve as invaluable guides to deepen your knowledge and proficiency in leveraging Gemma-7B, Ludwig, and other related tools in your machine learning endeavors.

Click image below or visit LinkedIn to know more about the author

Gemma Ludwig Generative Ai Machine Learning Google Cloud Platform

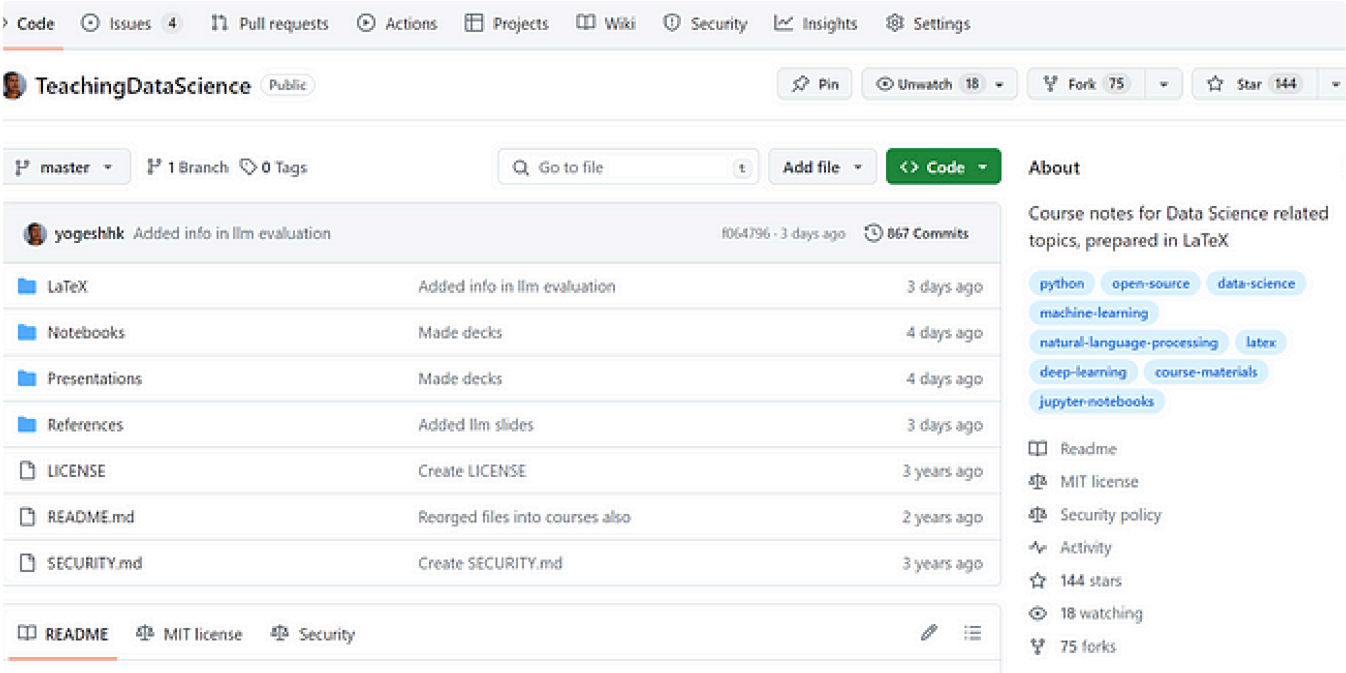
Edit profile

Written by Yogesh Haribhau Kulkarni (PhD)

1.3K Followers · Writer for Google Cloud - Community

PhD in Geometric Modeling | Google Developer Expert (Machine Learning) | Top Writer 3x (Medium) | More at https://www.linkedin.com/in/yogeshkulkarni/

More from Yogesh Haribhau Kulkarni (PhD) and Google Cloud - Community



TeachingDataScience Public

yogeshhk Added info in llm evaluation f064796 · 3 days ago 867 Commits

LaTeX	Added info in llm evaluation	3 days ago
Notebooks	Made decks	4 days ago
Presentations	Made decks	4 days ago
References	Added llm slides	3 days ago
LICENSE	Create LICENSE	3 years ago
README.md	Reorged files into courses also	2 years ago
SECURITY.md	Create SECURITY.md	3 years ago

About

Course notes for Data Science related topics, prepared in LaTeX

python open-source data-science machine-learning natural-language-processing latex deep-learning course-materials jupyter-notebooks

Readme
MIT license
Security policy
Activity
144 stars
18 watching
75 forks

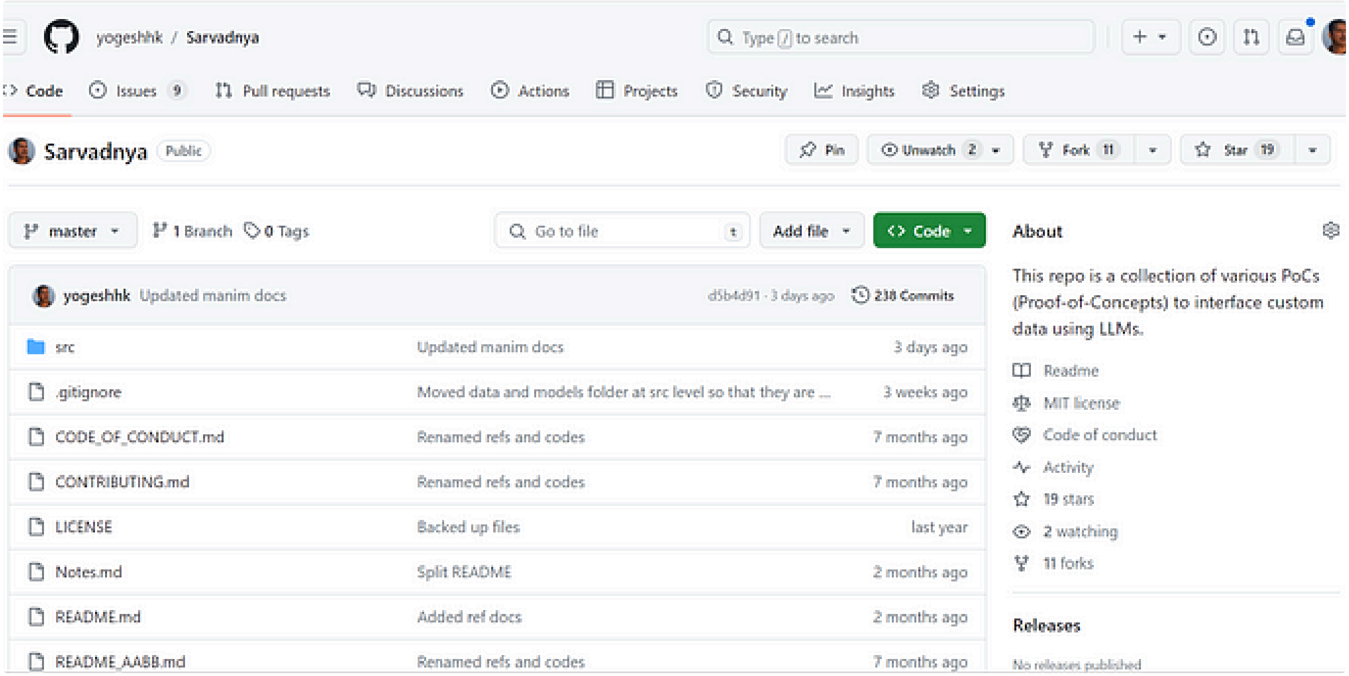
Yogesh Haribhau Kulkarni (PhD) in Technology Hits

Teaching Data Science

An open-source repository for teaching material, open-free to all

4 min read · Mar 4, 2024

145



yogeshhk Updated manim docs d5b4d91 · 3 days ago 238 Commits

src	Updated manim docs	3 days ago
.gitignore	Moved data and models folder at src level so that they are ...	3 weeks ago
CODE_OF_CONDUCT.md	Renamed refs and codes	7 months ago
CONTRIBUTING.md	Renamed refs and codes	7 months ago
LICENSE	Backed up files	last year
Notes.md	Split README	2 months ago
README.md	Added ref docs	2 months ago
README_AABB.md	Renamed refs and codes	7 months ago

About

This repo is a collection of various PoCs (Proof-of-Concepts) to interface custom data using LLMs.

- Readme
- MIT license
- Code of conduct
- Activity
- 19 stars
- 2 watching
- 11 forks

Releases

No releases published

Yogesh Haribhau Kulkarni (PhD) in Technology Hits

Sarvadnya

An open-source repository for All-Knowing Custom Chatbot

3 min read · Mar 4, 2024

9

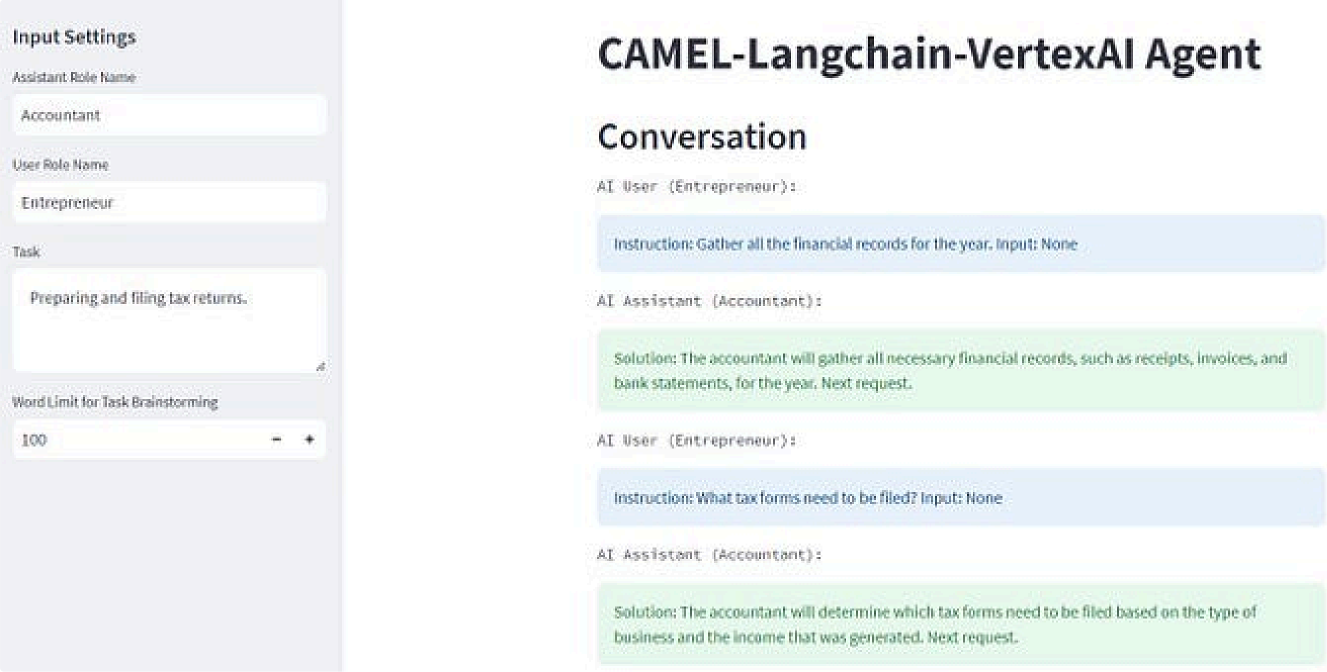
Yogesh Haribhau Kulkarni (PhD) in Technology Hits

Unveiling Manifold learning

What a neural network is really doing?

5 min read · Feb 23, 2024

275 1



Input Settings

Assistant Role Name

Accountant

User Role Name

Entrepreneur

Task

Preparing and filing tax returns.

Word Limit for Task Brainstorming

100

CAMEL-Langchain-VertexAI Agent

Conversation

AI User (Entrepreneur):

Instruction: Gather all the financial records for the year. Input: None

AI Assistant (Accountant):

Solution: The accountant will gather all necessary financial records, such as receipts, invoices, and bank statements, for the year. Next request.

AI User (Entrepreneur):

Instruction: What tax forms need to be filed? Input: None

AI Assistant (Accountant):

Solution: The accountant will determine which tax forms need to be filed based on the type of business and the income that was generated. Next request.

Yogesh Haribhau Kulkarni (PhD) in Google Developer Experts

A CAMEL ride

A Story of AI Role-Playing using CAMEL, Langchain and VertexAI

8 min read · Oct 2, 2023

54

See all from Yogesh Haribhau Kulkarni (PhD)

See all from Google Cloud - Community

Recommended from Medium

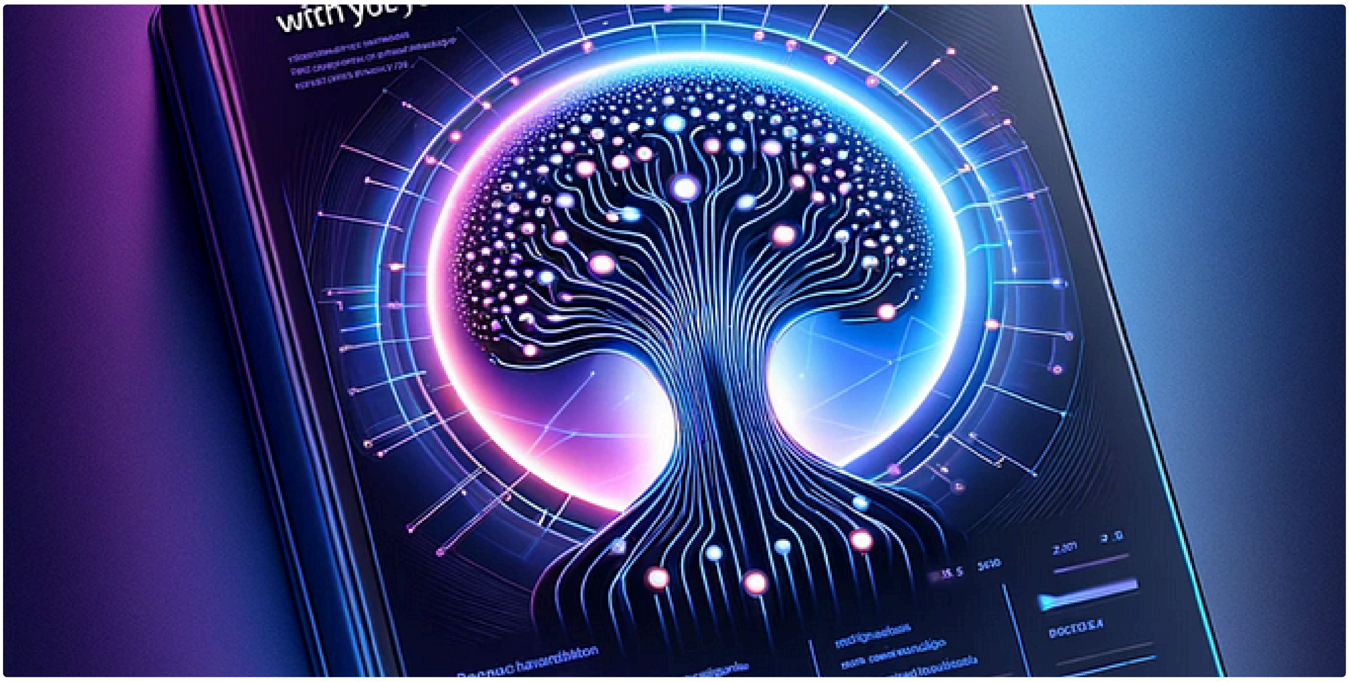
Shaw Talebi in Towards Data Science

How to Improve LLMs with RAG

A beginner-friendly introduction w/ Python code

✦ · 12 min read · 4 days ago

496 2



Rany ElHousieny in Level Up Coding

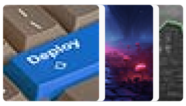
Fine Tune mistral-7b-instruct on Predibase with Your Own Data and LoRAX

In today's rapidly advancing world of artificial intelligence, fine-tuning pre-trained language models has become an essential practice for...

6 min read · Feb 29, 2024

103

Lists

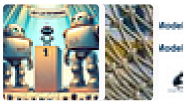


Predictive Modeling w/ Python

20 stories · 999 saves

Practical Guides to Machine Learning

10 stories · 1193 saves

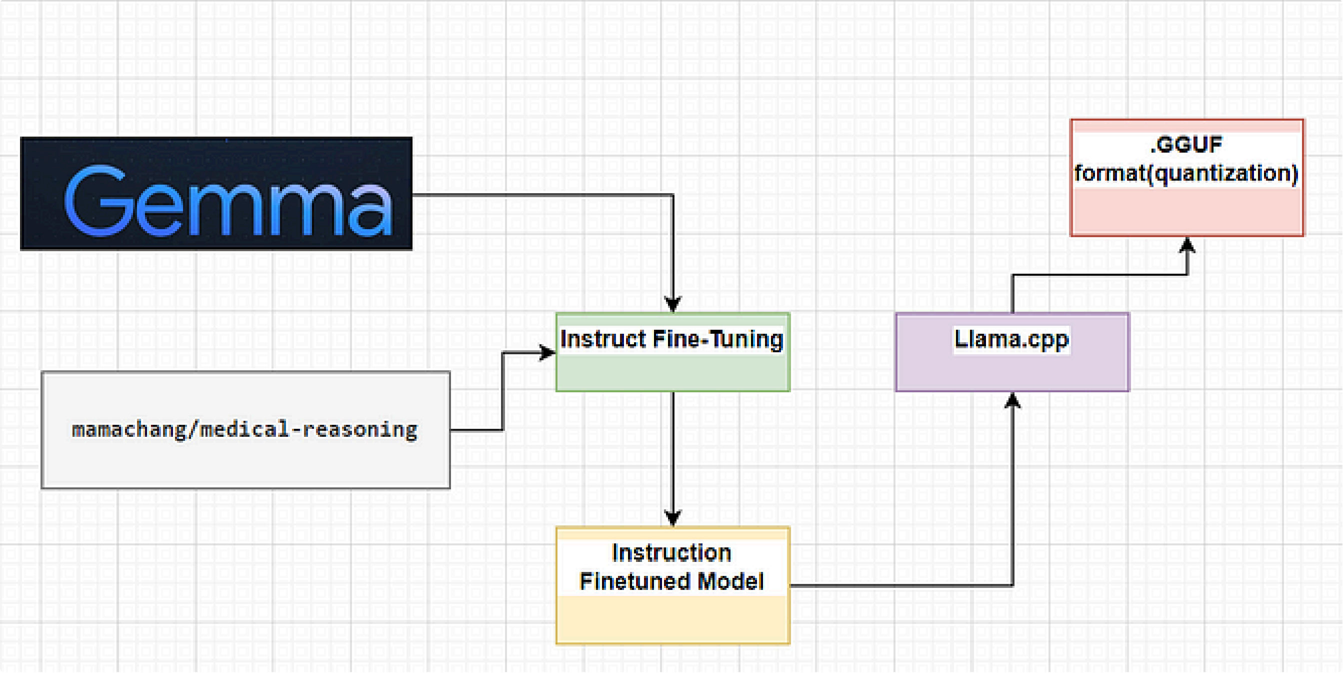


Natural Language Processing

1279 stories · 772 saves

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 333 saves


Gemma
.GGUF format(quantization)
Instruct Fine-Tuning
Llama.cpp
mamachang/medical-reasoning
Instruction Finetuned Model

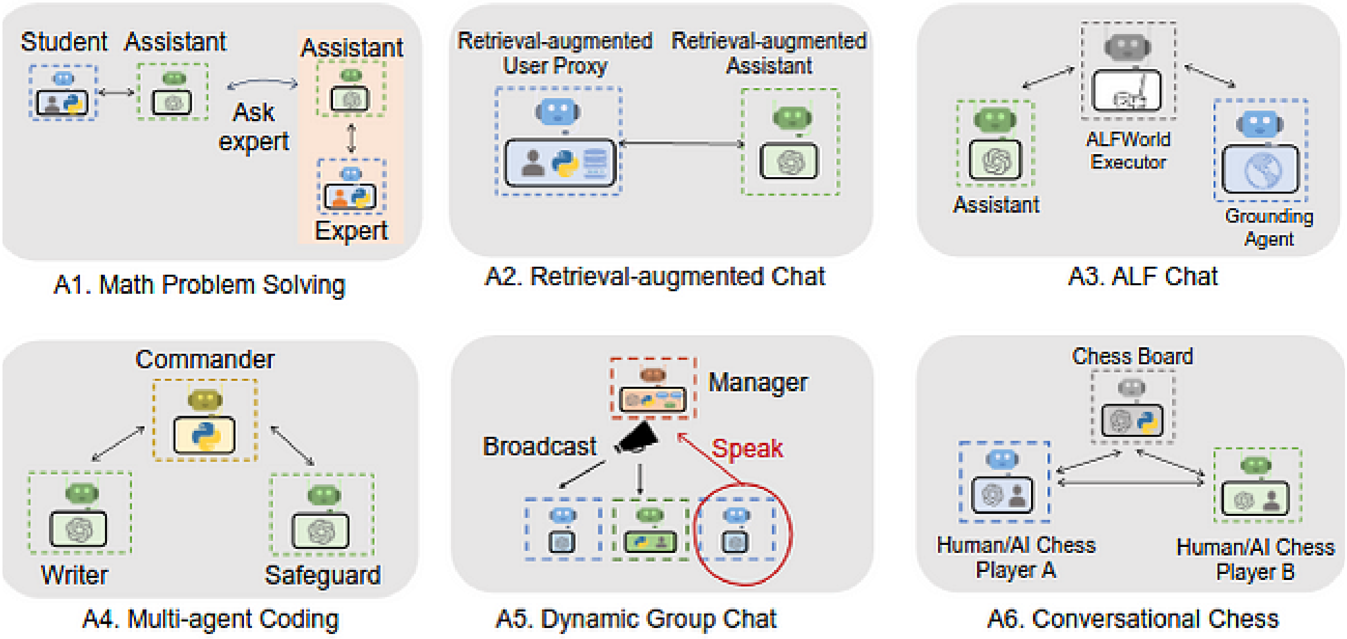
Plaban Nayak in The AI Forum

Instruction Fine-Tuning Gemma-2B on Medical Reasoning and Convert the finetuned model into GGUF…

Gemma is a family of lightweight, state-of-the-art open models built from the same research and technology used to create the Gemini…

33 min read · 3 days ago

30



Rubens Zimbres in Google Cloud - Community

Cost-Efficient Multi-Agent Collaboration with LangGraph + Gemma for Code Generation

In the end of January, 2024, I received an email about LangGraph and LangSmith. Some features were not in GA, so I signed up for the...

12 min read · Feb 24, 2024

159 1

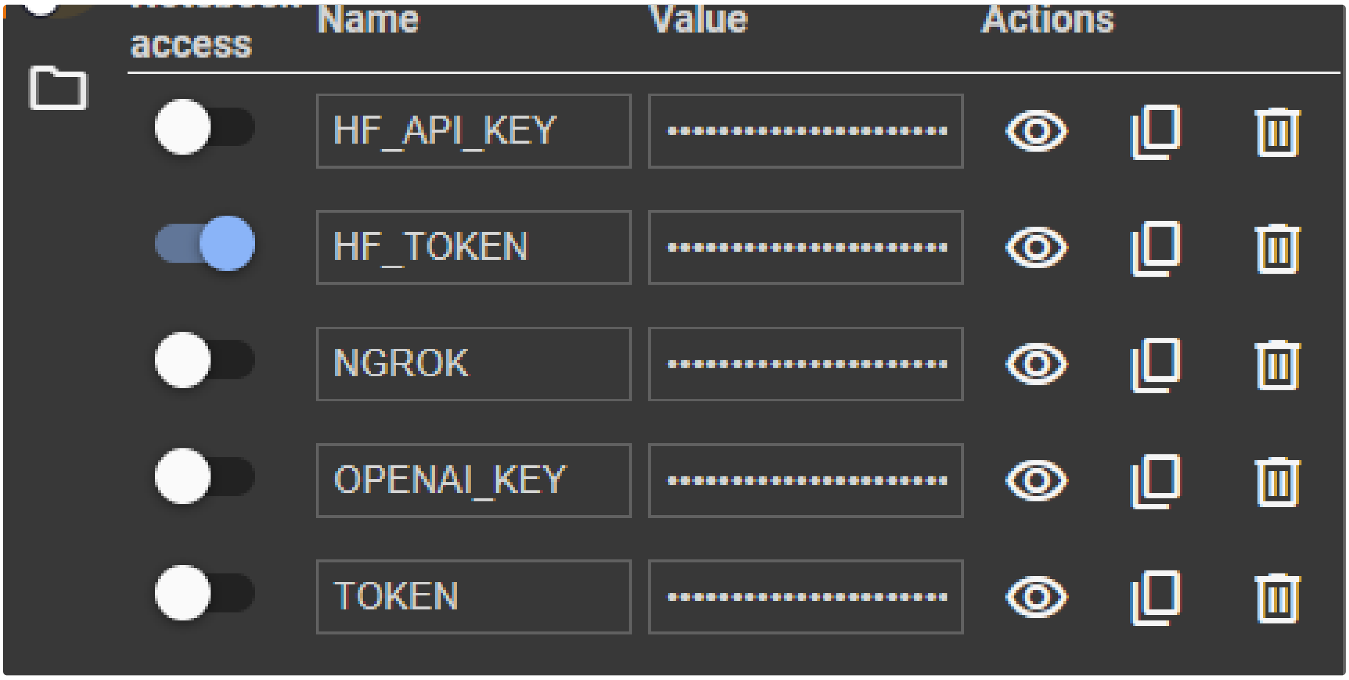
Shravan Kumar

Gemma: Introducing new state-of-the-art open model by Google

Gemma is a family of lightweight, state-of-the-art open models built from the same research and technology used to create the Gemini…

6 min read · Feb 23, 2024

18



access	Name	Value	Actions
○	HF_API_KEY	••••••••••••••••••••••••	
●	HF_TOKEN	••••••••••••••••••••••••	
○	NGROK	••••••••••••••••••••••••	
○	OPENAI_KEY	••••••••••••••••••••••••	
○	TOKEN	••••••••••••••••••••••••	

0Xkoji

Run Gemma-7B Model on Google Colab for Free

What is Gemma?

3 min read · Feb 28, 2024

1

See more recommendations