Edu Tutor AI: Personalized Learning

Generative AI with IBM

Project Description:

EduTutor AI uses the Granite model from Hugging Face to create simple, personalized learning tools like concept explainers, quizzes generator and add more functionalities that you like. This project is deployed in Google Colab using Granite for low setup effort and reliable performance.

My team has successfully enrolled for the project. Please find the team details below:

Team ID : NM2025TMID06285

Team Size : 4

Team Leader : JAGANATHAN B

Team member : KRISHNAN P

Team member : LOGANATHAN V

Team member : MOHAN S

Project Workflow:

Activity-1: Exploring Naan Mudhalavan Smart Interz Portal.

Activity-2: Choosing a IBM Granite Model From Hugging Face.

Activity-3: Running Application In Google Colab.

Activity-4: Upload your Project in Github.

Activity-1: Exploring Naan Mudhalavan Smart Interz Portal.

- Search for "Naan Mudhalavan Smart Interz" Portal in any Browser.
- Then Click on the first link. (Naanmudhalvan Smartinternz) Then login with your details.
- Then you will be redirected to your account then click on "Projects" Section. There you can see which project you have enrolled in here it is "EduTutor AI".
- Then click on "Access Resources" and go to the "Guided Project" Section.
- Click on the "Go to workspace" section. Then you can find the detailed explanation of Generative AI Project using IBM Watsonx API key.
- Click on "Project Workspace", there you can find your project progress and Place to upload "Demo link".
- Now we have gone through portal understanding, now lets find a IBM granite model from hugging face to integrate in our project.

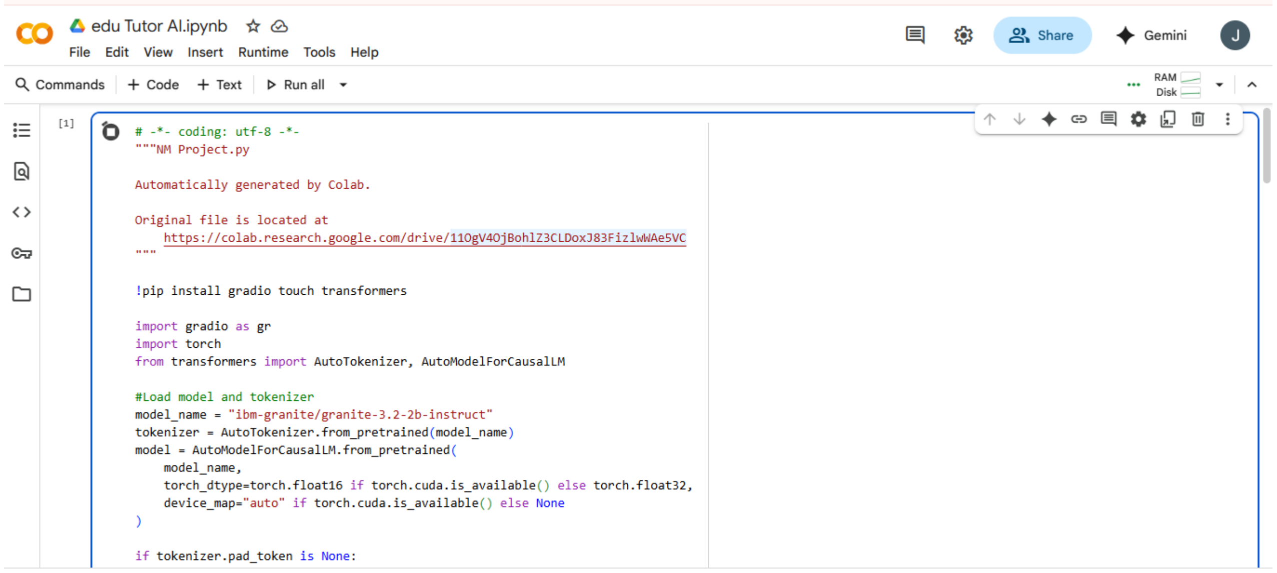
Activity-2: Choose a IBM Granite model From Hugging Face.

- Search for "Hugging face" in any browser.
- Then click on the first link (Hugging Face), then click on signup and create your own account in Hugging Face. Then search for "IBM-Granite models" and choose any model.
- Here for this project we are using "granite-3.2-2b-instruct" which is compatible fast and light weight.

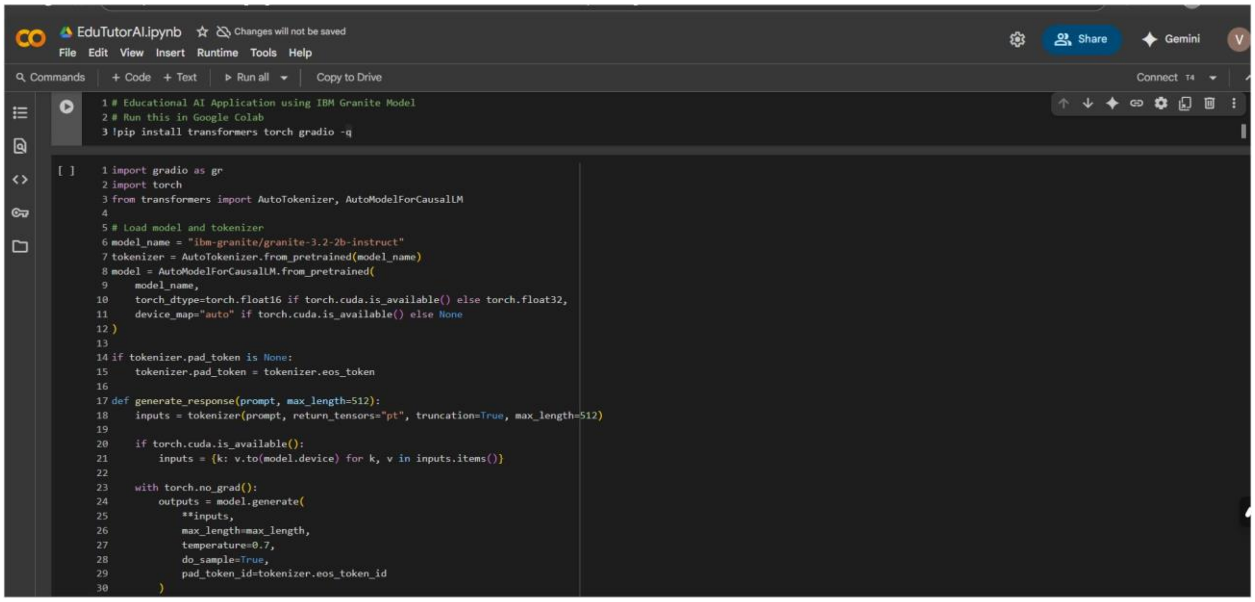
- Now we will start building our project in Google collab.

Activity-3: Running Application in Google Collab.

- Search for “Google collab” in any browser.
- Click on the first link (Google Colab), then click on “Files” and then “Open Notebook”.
- Click on “New Notebook”
- Change the title of the notebook “Untitled” to “Edu Tutor AI”. Then click on “Runtime”, then go to “Change Runtime Type”.
- Choose “T4 GPU” and click on “Save”
- Then run this command in the first cell “`!pip install transformers torch gradio -q`”. To install the required libraries to run our application.
- Then run the rest of the code in the next cell.

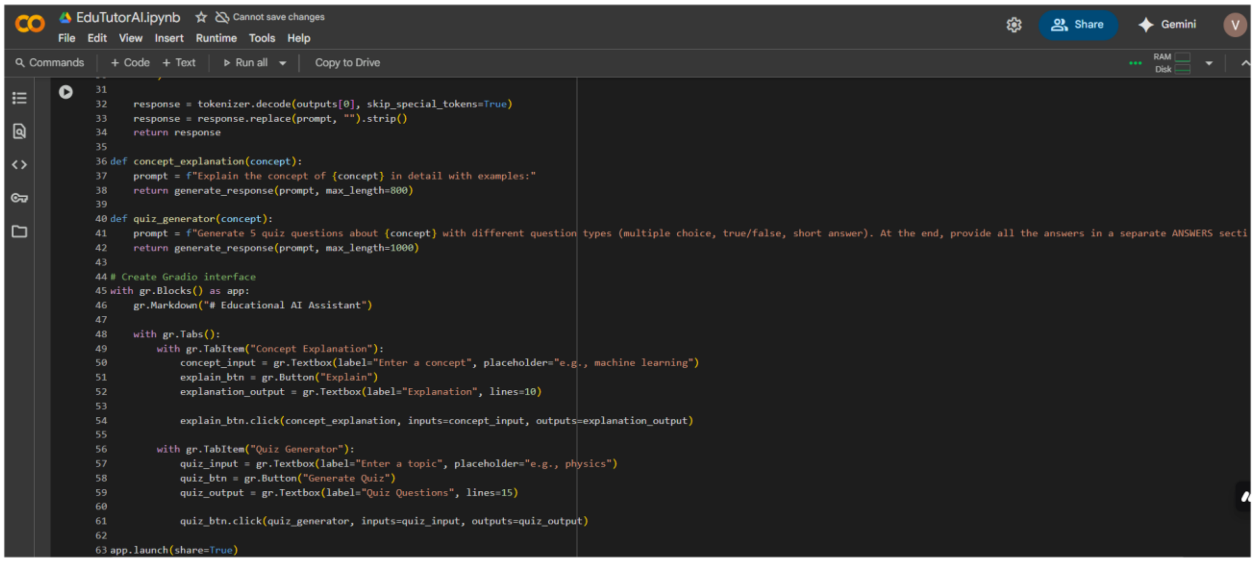

edu Tutor AI.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
RAM
Disk

```python
# -*- coding: utf-8 -*-
"""NM Project.py

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/11OgV4OjBohlZ3CLDoxJ83FizlwWAe5VC
"""

!pip install gradio touch transformers

import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

#Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)

if tokenizer.pad_token is None:
```



```python
# Educational AI Application using IBM Granite Model
# Run this in Google Colab
!pip install transformers torch gradio -q
```

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

# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

def generate_response(prompt, max_length=512):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)

    if torch.cuda.is_available():
        inputs = {k: v.to(model.device) for k, v in inputs.items()}

    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_length=max_length,
            temperature=0.7,
            do_sample=True,
            pad_token_id=tokenizer.eos_token_id
        )
```

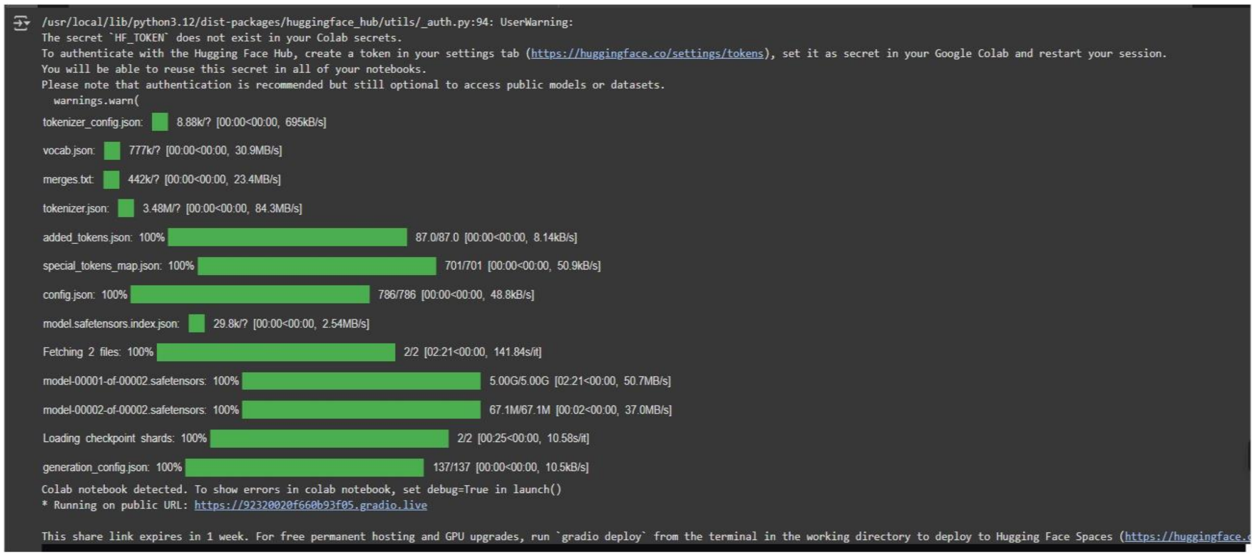


```python

    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    response = response.replace(prompt, "").strip()
    return response

def concept_explanation(concept):
    prompt = f"Explain the concept of {concept} in detail with examples:"
    return generate_response(prompt, max_length=800)

def quiz_generator(concept):
    prompt = f"Generate 5 quiz questions about {concept} with different question types (multiple choice, true/false, short answer). At the end, provide all the answers in a separate ANSWERS secti
    return generate_response(prompt, max_length=1000)

# Create Gradio interface
with gr.Blocks() as app:
    gr.Markdown("# Educational AI Assistant")

    with gr.Tabs():
        with gr.TabItem("Concept Explanation"):
            concept_input = gr.Textbox(label="Enter a concept", placeholder="e.g., machine learning")
            explain_btn = gr.Button("Explain")
            explanation_output = gr.Textbox(label="Explanation", lines=10)

            explain_btn.click(concept_explanation, inputs=concept_input, outputs=explanation_output)

        with gr.TabItem("Quiz Generator"):
            quiz_input = gr.Textbox(label="Enter a topic", placeholder="e.g., physics")
            quiz_btn = gr.Button("Generate Quiz")
            quiz_output = gr.Textbox(label="Quiz Questions", lines=15)

            quiz_btn.click(quiz_generator, inputs=quiz_input, outputs=quiz_output)

app.launch(share=True)
```

- You can find the code here in this link: [Edu Tutor AI Code]

OUTPUT:

- Now you can see our model is being Downloaded and the application is running.



```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json:  8.88k/? [00:00<00:00, 695kB/s]
vocab.json:  777k/? [00:00<00:00, 30.9MB/s]
merges.txt:  442k/? [00:00<00:00, 23.4MB/s]
tokenizer.json:  3.48M/? [00:00<00:00, 84.3MB/s]
added_tokens.json: 100%  87.0/87.0 [00:00<00:00, 8.14kB/s]
special_tokens_map.json: 100%  701/701 [00:00<00:00, 50.9kB/s]
config.json: 100%  786/786 [00:00<00:00, 48.8kB/s]
model.safetensors.index.json:  29.8k/? [00:00<00:00, 2.54MB/s]
Fetching 2 files: 100%  2/2 [02:21<00:00, 141.84s/it]
model-00001-of-00002.safetensors: 100%  5.00G/5.00G [02:21<00:00, 50.7MB/s]
model-00002-of-00002.safetensors: 100%  67.1M/67.1M [00:02<00:00, 37.0MB/s]
Loading checkpoint shards: 100%  2/2 [00:25<00:00, 10.58s/it]
generation_config.json: 100%  137/137 [00:00<00:00, 10.5kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://92320020f660b93f05.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.
```

- Click on the URl to open the Gradio Application click on the link (https://colab.research.google.com/drive/1Md-Np-33EbEeABFWk9Yi03bbwumP1-tH?usp=sharing).

Activity-4: Upload Your Project in GitHub.

- Search for "GitHub" in any browser, then click on the first link (GitHub).
- Then click on "Signup" and create your own account in GitHub. If you already have an account click on "Sign in"
- Click on "Create repository".
- In "General" Name your repo. (Here I have given "https://github.com/jaganathan9626/NM2025TMID06285.git" as my repo name and it is available).
- In "Configurations" Turn On "Add readme" file Option.
- Now Download your code from Google collab by Clicking on "File", then Go to "Download" then download as ".py".
- Then your repository is created, then Click on "Add file" Option. Then Click "Upload files" to upload your files.
- Click on "choose your files".
- Choose your project file and click on "Open".

- After your file has Uploaded Click on “Commit changes”.