EDU TUTOR AI

Introduction

Team NM_ID:NM2025TMID01891)

Team Members : V R SANDHYA(TL) (4A7F63F004F2799EFD4DF0672FD38FD)

S SAKTHI : (D236451AFAB4A24BA794D8EF98A8B643)

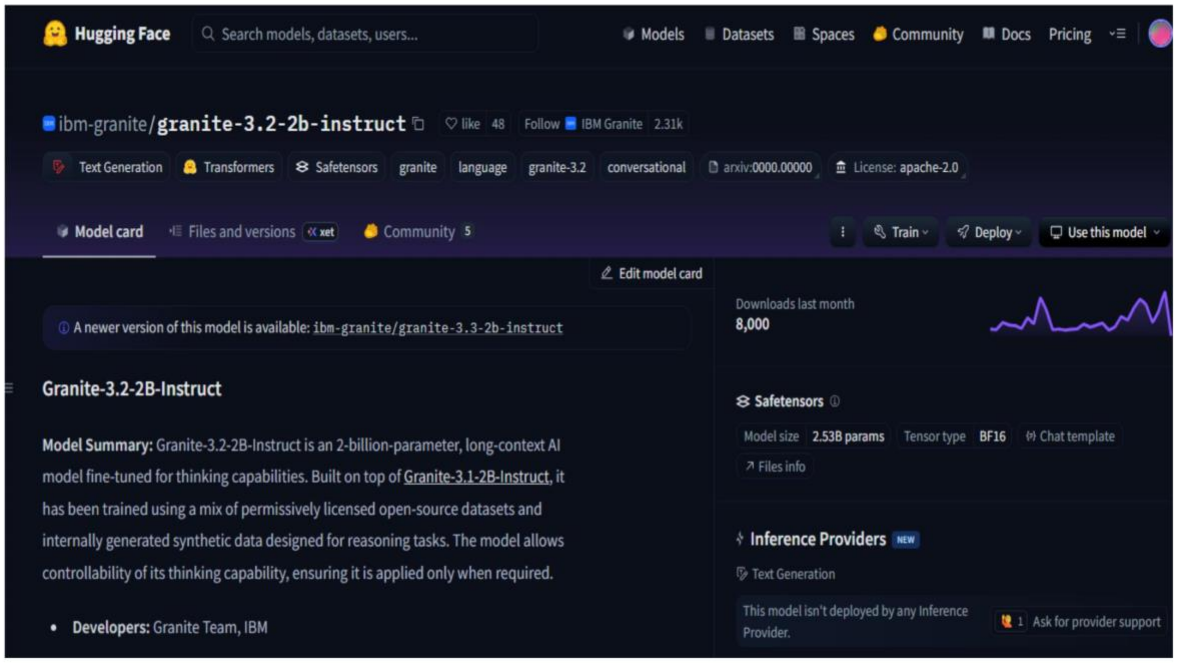
P SREE JEGADEESWARI : (514E32A481322E996C14A68BBA312BBF)

S TAMILSELVI : (A59091E27F5979EB4A647F002DDE5F93)

V VINOTHINI : (28269151D4B53BA89D341E8FFD827EB4)

DOCUMENTATION

Documentation for Educational AI AssistantTable of ContentsIntroductionSystem OverviewFeaturesDependencies and RequirementsInstallation GuideCode Structure OverviewDetailed Code WalkthroughModel SetupTokenizer SetupUtility FunctionsConcept Explanation FunctionQuiz Generator FunctionGradio InterfaceApplication WorkflowUser Interaction FlowError Handling and Edge CasesPerformance ConsiderationsDeployment GuidePossible EnhancementsSecurity ConsiderationsTesting StrategyExample Usage Screenshots (Placeholder)Future ScopeConclusion1. IntroductionThis project is an AI-powered Educational Assistant that leverages large language models (LLMs) to explain academic concepts and generate quizzes interactively using a web-based UI powered by Gradio. The primary model used is IBM's Granite family of open-weight LLMs.The application's key purposes are:Enable students/learners to understand complex concepts interactively.Provide quizzes for self-assessment.Serve as a lightweight educational tool with real-time AI assistance.2. System OverviewThe system integrates:Hugging Face Transformers: To load and run the pretrained large language model.PyTorch: To run inference on CPU or GPU for faster generation.Gradio: To create a user-friendly, shareable web interface.Users can type in concepts or topics, and the model will return detailed explanations or custom quizzes.3. FeaturesConcept Explanation: Detailed breakdown of any concept with real-world examples.Quiz Generator: Creates 5 diverse questions with different formats (MCQ, True/False, short answer).User-friendly Interface: Powered by Gradio Tabs



ibm-granite/**granite-3.2-2b-instruct**

A newer version of this model is available: `ibm-granite/granite-3.3-2b-instruct`

Granite-3.2-2B-Instruct

Model Summary: Granite-3.2-2B-Instruct is an 2-billion-parameter, long-context AI model fine-tuned for thinking capabilities. Built on top of Granite-3.1-2B-Instruct, it has been trained using a mix of permissively licensed open-source datasets and internally generated synthetic data designed for reasoning tasks. The model allows controllability of its thinking capability, ensuring it is applied only when required.

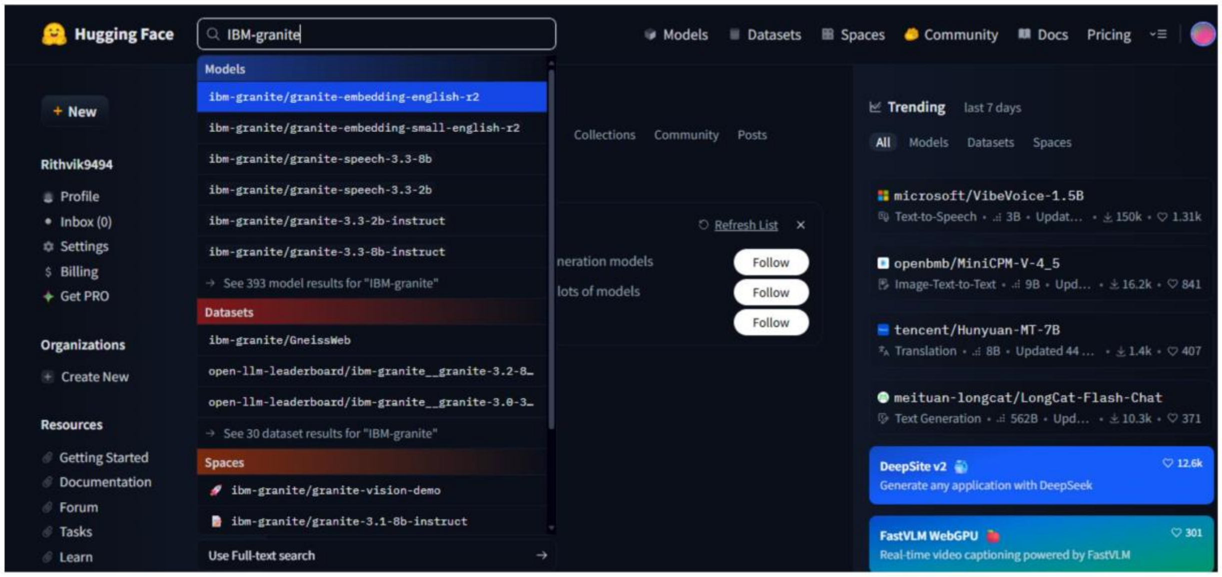
- **Developers:** Granite Team, IBM

Downloads last month 8,000

Safetensors — Model size 2.53B params — Tensor type BF16 — Chat template — Files info

Inference Providers NEW — Text Generation

This model isn't deployed by any Inference Provider. 1 Ask for provider support



IBM-granite

Models
- ibm-granite/granite-embedding-english-r2
- ibm-granite/granite-embedding-small-english-r2
- ibm-granite/granite-speech-3.3-8b
- ibm-granite/granite-speech-3.3-2b
- ibm-granite/granite-3.3-2b-instruct
- ibm-granite/granite-3.3-8b-instruct
- See 393 model results for "IBM-granite"

Datasets
- ibm-granite/GneissWeb
- open-llm-leaderboard/ibm-granite__granite-3.2-8…
- open-llm-leaderboard/ibm-granite__granite-3.0-3…
- See 30 dataset results for "IBM-granite"

Spaces
- ibm-granite/granite-vision-demo
- ibm-granite/granite-3.1-8b-instruct

…neration models
…lots of models

microsoft/VibeVoice-1.5B — Text-to-Speech • 3B • Updat… • 150k • 1.31k
openbmb/MiniCPM-V-4_5 — Image-Text-to-Text • 9B • Upd… • 16.2k • 841
tencent/Hunyuan-MT-7B — Translation • 8B • Updated 44… • 1.4k • 407
meituan-longcat/LongCat-Flash-Chat — Text Generation • 562B • Upd… • 10.3k • 371
DeepSite v2 — 12.6k — Generate any application with DeepSeek
FastVLM WebGPU — 301 — Real-time video captioning powered by FastVLM

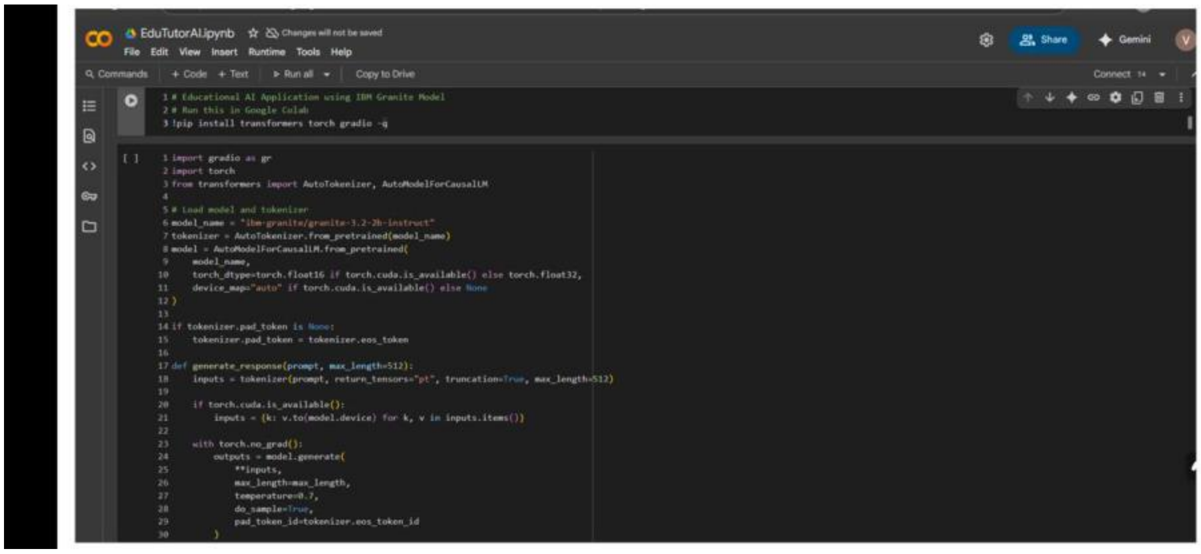
for easy navigation.Cross-device Support: Shareable Gradio links allow users to access the app in a browser.GPU Utilization:

If CUDA is available, the model runs efficiently on GPU.4. Dependencies and RequirementsProgramming Language: Python 3.8+Libraries Used:torch (PyTorch)transformers (Hugging Face LLM toolkit)gradio (Web interface)Hardware Requirements:CPU-only mode: Basic usage possible with slower generation.GPU (Recommended): CUDA-enabled device for better performance.5. Installation Guide# Create a virtual environment

python -m venv edai_env

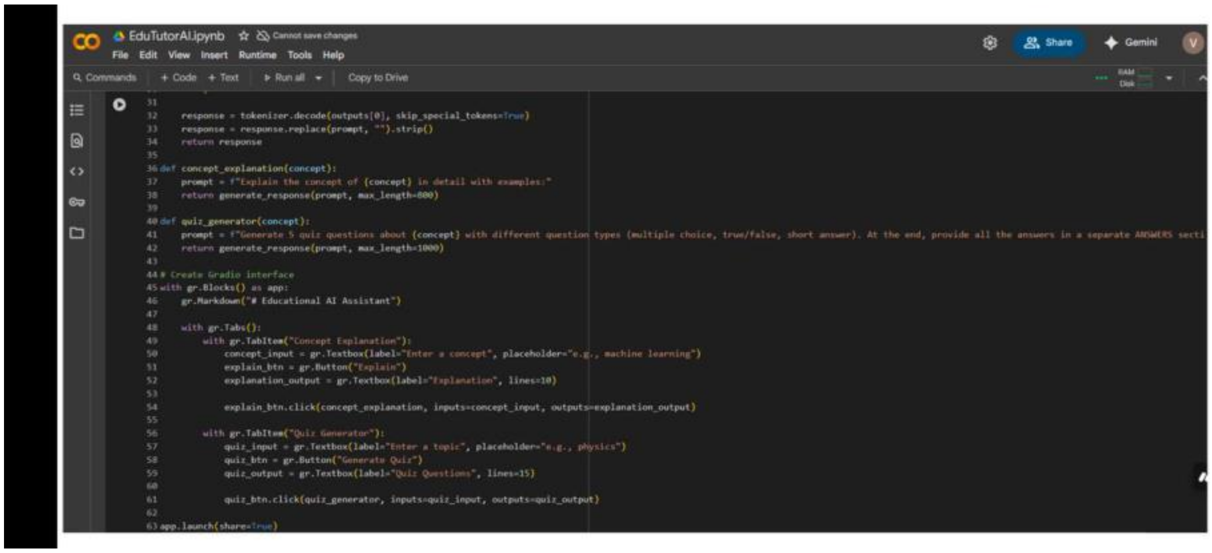
source edai_env/bin/activate # Linux/Mac

edai_envScriptsactivate   # Windows


EduTutorAI.ipynb
Changes will not be saved
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all Copy to Drive
Share
Gemini
Connect

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

```
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)
```

```
# Install dependencies
pip install torch
pip install transformers
```

pip install gradio6. Code Structure OverviewModel and Tokenizer Loading: Defines which LLM is being used.Utility Functions: Handles prompt construction and response generation.Concept Explanation Module: Returns AI-generated explanation of topics.Quiz Generator Module: Returns 5 quiz questions + answers.Gradio UI Setup: Creates tabs with input fields and interactive buttons.7. Detailed Code WalkthroughModel Setupmodel_name = "ibm-granite/granite-3.2-2b-instruct"

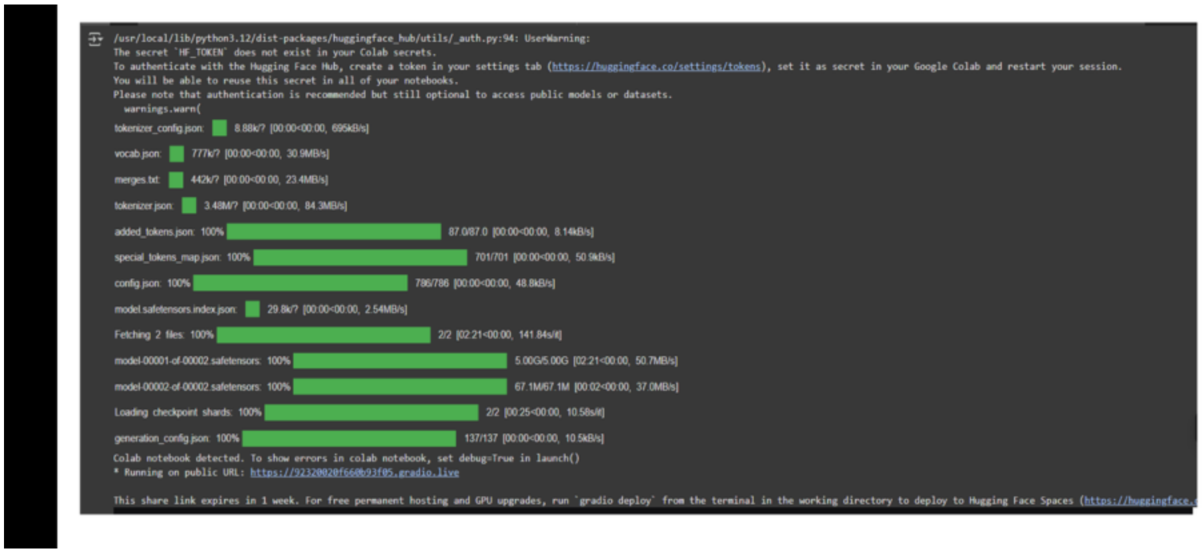
```
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
```

)Loads IBM Granite model for text generation.Chooses float16 for efficiency when GPU is available.Automatically assigns device placement if CUDA is available.Tokenizer Setupif tokenizer.pad_token is None:

```
    tokenizer.pad_token = tokenizer.eos_tokenEnsures
```

tokenizer has a valid pad_token.Required because Hugging Face models sometimes lack padding tokens.Utility Function: Response Generatordef 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
        )


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



```
/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.
```

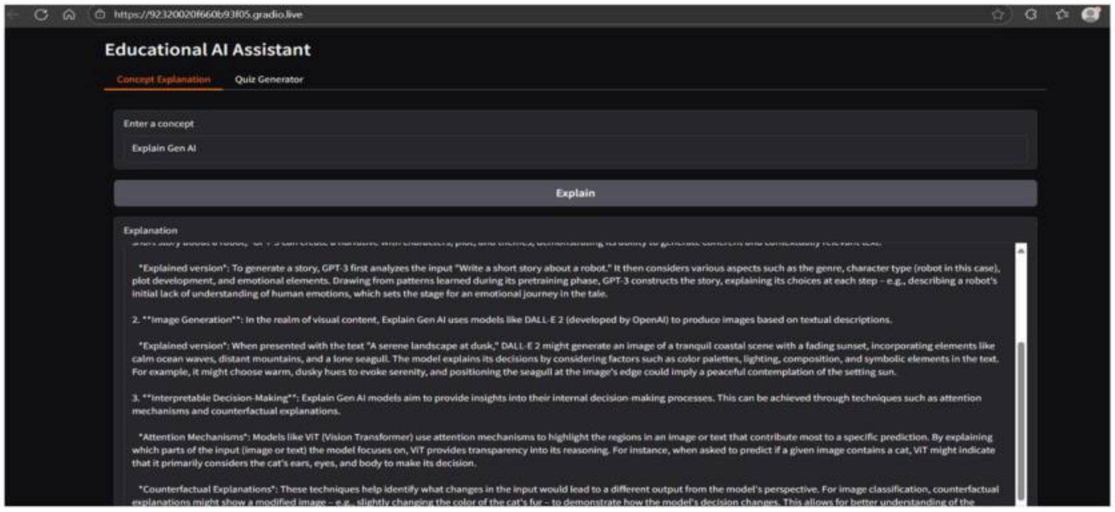
return responseHandles prompt creation and processing through the LLM.temperature=0.7 ensures balanced creativity.Strips prompt from final response.Concept Explanationdef concept_explanation(concept):

prompt = f"Explain the concept of {concept} in detail with examples:"

return generate_response(prompt, max_length=800)Quiz Generatordef 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 section."

return generate_response(prompt, max_length=1000)Gradio UIwith gr.Blocks() as app:



Educational AI Assistant

Enter a concept

Explain Gen AI

Explanation

short story about a robot," GPT-3 can create a narrative with characters, plot, and themes, demonstrating its ability to generate coherent and contextually relevant text.

Explained version: To generate a story, GPT-3 first analyzes the input "Write a short story about a robot." It then considers various aspects such as the genre, character type (robot in this case), plot development, and emotional elements. Drawing from patterns learned during its pretraining phase, GPT-3 constructs the story, explaining its choices at each step – e.g., describing a robot's initial lack of understanding of human emotions, which sets the stage for an emotional journey in the tale.

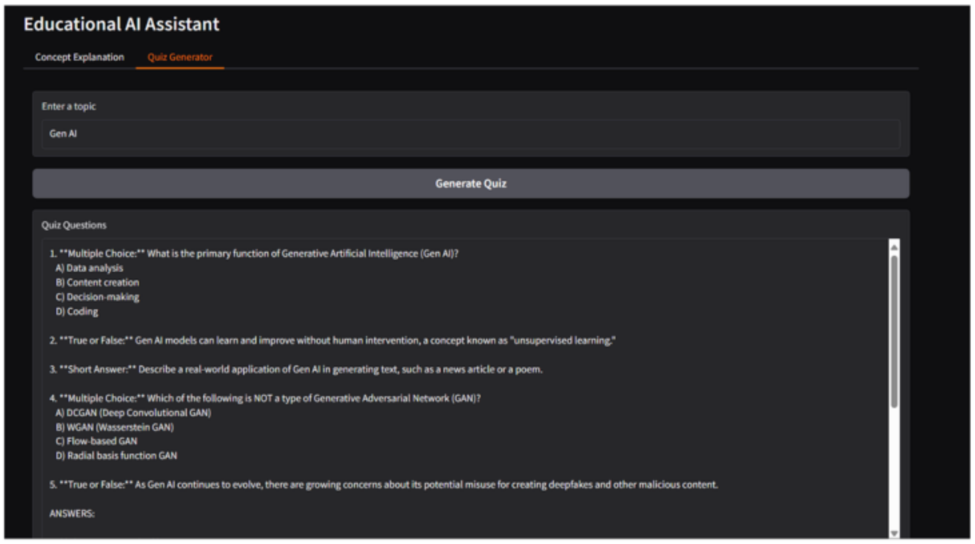
2. **Image Generation**: In the realm of visual content, Explain Gen AI uses models like DALL-E 2 (developed by OpenAI) to produce images based on textual descriptions.

Explained version: When presented with the text "A serene landscape at dusk," DALL-E 2 might generate an image of a tranquil coastal scene with a fading sunset, incorporating elements like calm ocean waves, distant mountains, and a lone seagull. The model explains its decisions by considering factors such as color palettes, lighting, composition, and symbolic elements in the text. For example, it might choose warm, dusky hues to evoke serenity, and positioning the seagull at the image's edge could imply a peaceful contemplation of the setting sun.

3. **Interpretable Decision-Making**: Explain Gen AI models aim to provide insights into their internal decision-making processes. This can be achieved through techniques such as attention mechanisms and counterfactual explanations.

Attention Mechanisms: Models like ViT (Vision Transformer) use attention mechanisms to highlight the regions in an image or text that contribute most to a specific prediction. By explaining which parts of the input (image or text) the model focuses on, ViT provides transparency into its reasoning. For instance, when asked to predict if a given image contains a cat, ViT might indicate that it primarily considers the cat's ears, eyes, and body to make its decision.

Counterfactual Explanations: These techniques help identify what changes in the input would lead to a different output from the model's perspective. For image classification, counterfactual explanations might show a modified image – e.g., slightly changing the color of the cat's fur – to demonstrate how the model's decision changes. This allows for better understanding of the



Educational AI Assistant

Enter a topic

Gen AI

Quiz Questions

1. **Multiple Choice:** What is the primary function of Generative Artificial Intelligence (Gen AI)?
   A) Data analysis
   B) Content creation
   C) Decision-making
   D) Coding

2. **True or False:** Gen AI models can learn and improve without human intervention, a concept known as "unsupervised learning."

3. **Short Answer:** Describe a real-world application of Gen AI in generating text, such as a news article or a poem.

4. **Multiple Choice:** Which of the following is NOT a type of Generative Adversarial Network (GAN)?
   A) DCGAN (Deep Convolutional GAN)
   B) WGAN (Wasserstein GAN)
   C) Flow-based GAN
   D) Radial basis function GAN

5. **True or False:** As Gen AI continues to evolve, there are growing concerns about its potential misuse for creating deepfakes and other malicious content.

ANSWERS:

```
    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)
```

Simple two-tab layout.One for explanations, one for quizzes.The share=True parameter generates a public Gradio link.8. Application WorkflowUser enters a concept/topic.Backend sends request to model with custom prompt.Model generates a textual response.Response displayed in Gradio interface.9. User Interaction FlowTab 1 (Concept Explanation): Input concept →Click "Explain" → Read detailed answer.Tab 2 (Quiz Generator): Input topic → Click "Generate Quiz" →Get 5 questions + answers.10. Error Handling and Edge CasesIf user inputs an empty string → Returns minimal or blank response.If GPU not available → Runs smoothly on CPU with slower speed.If concept is too broad →Model may generate generic results.11. Performance

ConsiderationsGPU usage boosts performance.Batch processing could improve scalability.Current setup is user-friendly but not optimized for multi-user loads.12. Deployment GuideCan be hosted using Gradio share link.For production: Deploy on Hugging Face Spaces or Streamlit Cloud.For enterprise: Containerize with Docker and serve behind API.13. Possible EnhancementsAdd speech-to-text input for learners.Include PDF export feature for quizzes.Store user quiz history in a backend database.Multi-language explanation support.14. Security ConsiderationsSince prompts are user-inputted, guard against malicious usage.Ensure prompt injection attacks are minimized.When deployed publicly, add query filters.15. Testing StrategyUnit Tests: For generate_response, concept_explanation, quiz_generator.Integration Tests: Verify end-to-end model responses in the UI.User Acceptance Testing: Gather feedback from educators and students.16. Example Usage Screenshots(Placeholder – include screenshots of Gradio UI once running.)17. Future ScopeAdaptive learning path generation.Integration with classroom dashboards.Expand question variety with difficulty levels.18. ConclusionThis Educational AI Assistant demonstrates how LLMs can enhance learning experiences. With concept explanations and quiz generation, it supports both teaching and self-study. While minimalistic in design, it offers strong potential for extension to a full-fledged learning system.Would you like me to expand this into a polished PDF report (with headers, images, and proper formatting) so you can directly use it as a 15–20 page document?

BenefitTailored MaterialsReal-Time FeedbackProgress TrackingFlexible ExportCustomizationCollaborationEdutorAI ExampleQuizzes/flashcards from student input

□ Auto grading, solution hints

□ Analytics dashboard

□ Save/share as PDF, PPT, CSV

□ Edit questions/edit flashcards

□ Group study and sharing

□ How It WorksAI processes student work for relevanceAI assesses and explains answers instantlyMonitors performance and adapts contentAny device, any format for personalized reviewUser fine-tunes resources for preferencesBuild, review, and learn interactivelyBenefitEdutorAI ExampleHow It WorksTailored MaterialsQuizzes/flashcards from student input

□ AI processes student work for relevanceReal-Time FeedbackAuto grading, solution hints

□ AI assesses and explains answers instantlyProgress TrackingAnalytics dashboard

□ Monitors performance and adapts contentFlexible ExportSave/share as PDF, PPT, CSV

□ Any device, any format for personalized reviewCustomizationEdit questions/edit flashcards User fine-tunes resources for preferencesCollaborationGroup study and sharing

□ Build, review, and learn interactivelyEdutorAI enhances personalized learning experiences by dynamically generating study materials and assessments tailored to each learner’s unique needs, preferences, and real-time performance

- Its AI-powered features help students, teachers, and parents create, modify, and share customized quizzes, flashcards, and worksheet content, ensuring learning is more relevant and engaging
- Key Ways EdutorAI Personalizes LearningAdaptive Content GenerationEdutorAI analyzes student-supplied text, images, or PDFs to create personalized learning resources, such as quizzes and flashcards, focusing on topics where learners need improvement or more practice
- The platform uses performance analytics to assess progress and tailors subsequent material based on strengths and gaps, allowing for individualized practice and review
- Real-Time Feedback and Progress TrackingAfter students interact with quizzes or submit answers, EdutorAI offers instant evaluations and actionable feedback, making learning adaptive and iterative
- Students and educators can monitor advancement, identify difficult concepts, and adjust their study strategies accordingly

- Flexible Modes of DeliveryLearning resources can be exported in multiple formats—PDFs, text, CSV, PPT—for convenient study, review, and collaboration
- The system supports both self-paced learning and classroom use, adapting to individual and group learning contexts
- Customizable and Editable ResourcesUsers can modify the AI-generated questions, quizzes, and flashcards, refining them to match specific learning goals, levels, or interests

 Teachers can adapt content for differentiated instruction, while students can create material suited to their learning pace
 Collaboration and Peer EngagementEdutorAI’s sharing features promote collaborative learning by allowing classmates, teachers, and parents to exchange materials, feedback, and study progress
 This interaction helps learners benefit from tailored group activities and peer-to-peer support
 Benefits TableBenefitEdutorAI ExampleHow It WorksTailored MaterialsQuizzes/flashcards from student input
 AI processes student work for relevanceReal-Time FeedbackAuto grading, solution hints
 AI assesses and explains answers instantlyProgress TrackingAnalytics dashboard
 Monitors performance and adapts contentAny device, any format for personalized reviewFlexible ExportSave/share as PDF, PPT, CSV
 CustomizationEdit questions/edit flashcards
 User fine-tunes resources for preferencesCollaborationGroup study and sharing
 Build, review, and learn interactivelyEdutorAI’s adaptive, interactive tools ensure that every learner receives the right material at the right time, maximizing engagement, retention, and success

What is Editor AI?

Editor AI is a tool that uses artificial intelligence to help users write, edit, and improve text (like documents, emails, or articles) automatically or with smart suggestions.

🧩Main Components

1. User Interface (UI)

Text editor where users type or paste content

Buttons for grammar check, rewrite, summarize, etc.

2. Input Processing

Converts user text into a format the AI can understand

Detects language, tone, and writing style

3. AI Engine (Core Brain)

Uses Natural Language Processing (NLP)

Performs tasks like:

Grammar & spell check

Rewriting sentences

Summarizing text
Improving clarity or tone

4. Knowledge Base (Optional)

Domain-specific rules or style guides
Custom vocabulary (e.g., brand terms)

5. Output Generator

Returns edited or improved text
May offer suggestions with explanations

6. Feedback Loop (Learning)

Tracks user choices (accepted or rejected edits)
Improves AI suggestions over time (if learning is enabled)

🔄How It Works – Step by Step

1. 📝User types or pastes text

2. 📜System analyzes text (grammar, tone, structure)

3. 🤖AI processes and suggests improvements

4. ✅User accepts, edits, or ignores suggestions

5. 🔁(Optional) AI learns from feedback to improve future results

🔑Extra Key Features of EduTutorAI

24/7 Virtual Tutor

Acts like a personal AI tutor available anytime.

Students can ask doubts instantly without waiting for teachers.

Gamified Learning

Adds badges, leaderboards, and rewards to keep students motivated.

Makes learning fun and engaging like a game.

Adaptive Assessments

Difficulty of questions changes based on student performance.

Provides customized question banks for weak areas.

Content Generation for Teachers

Teachers can use EduTutorAI to generate lesson plans, worksheets, and quizzes in minutes.

Saves time and effort in preparing study material.

Data-Driven Insights

Generates performance analytics and progress reports.

Helps teachers and parents monitor learning outcomes.

Multilingual Support

Supports regional languages (Tamil, Hindi, Telugu, etc.) along with English.

Encourages inclusive education for rural and urban learners.

AI-Powered Revision Tools

Creates flashcards, mind maps, summaries automatically.

Suggests revision tests before exams.

Seamless Integration

Can connect with Google Classroom, MS Teams, and other LMS platforms.

Compatible with mobile, tablet, and PC devices.

Offline Learning Support

Provides downloadable notes, PDFs, and recorded lectures.

Helps students with low internet access.

⚙️ Dependencies and Requirements of EduTutorAI

1. Hardware Requirements

Minimum

Processor: Dual Core CPU (Intel i3 / AMD equivalent)

RAM: 4 GB

Storage: 20 GB free space

Internet: 2–5 Mbps connection

Recommended

Processor: Quad Core CPU (Intel i5 / AMD Ryzen 5 or above)

RAM: 8–16 GB (for smooth AI model execution)

Storage: 50 GB SSD (for faster access to AI datasets & models)

GPU: NVIDIA T4 / RTX (for running Generative AI models in Colab or locally)

Internet: Stable 10+ Mbps broadband

2. Software Requirements

Operating System

Windows 10/11, Linux (Ubuntu 20.04+), or macOS

Programming Languages

Python 3.9+ (primary language)

JavaScript (for frontend integration if needed)

Libraries / Frameworks (Python)

transformers (Hugging Face – AI models)

torch (PyTorch – AI/ML framework)

gradio (for interactive AI apps)

scikit-learn (machine learning utilities)

numpy, pandas (data handling)

matplotlib / seaborn (data visualization)

Web / App Tools

Flask / FastAPI (for backend services)

HTML, CSS, JavaScript (for UI/UX)

Tailwind / Bootstrap (optional styling)

3. AI Dependencies

Model Requirements

Pre-trained LLMs like IBM Granite 3.2, LLaMA, GPT, or Falcon

Fine-tuned models for question answering, summarization, and translation

Cloud / Runtime Environment

Google Colab (GPU T4 runtime for training & testing)

Optionally: AWS, Azure, or IBM Cloud for scalability

Datasets Needed

Educational datasets (NCERT, CBSE, AI-generated content)

Custom training sets for Tamil + English bilingual education

4. Platform Dependencies

LMS Integration

Moodle, Google Classroom, or custom LMS support

Database

MySQL / PostgreSQL (for storing user data, scores, progress)

Version Control

Git / GitHub (for project collaboration)

5. User Requirements

Students need only:
A mobile/PC with internet
A registered account (EduTutorAI portal / app login)
Teachers/Admins need:
Access to dashboard for monitoring progress & uploading resources

✅ In short:

Hardware →Laptop/PC with GPU preferred

Software →Python + AI libraries + LMS integration tools

AI →Hugging Face / IBM Granite models, Colab runtime

Platform →LMS + Database + Cloud hosting

⚙️ Installation of EduTutorAI

1. Prerequisites

Before installation, ensure the following are ready:
A system with Python 3.9+ installed

Stable internet connection

Access to Google Colab / VS Code / Jupyter Notebook

GitHub account (for cloning repositories)

2. Installation Steps

◈Step 1: Clone / Download the Project

Open Google Colab or your terminal.

Run the following command:

```
git clone https://github.com/your-repo/edututorai.git
cd edututorai
```

◈Step 2: Install Python Dependencies

Install required Python libraries using pip:

```
pip install torch torchvision torchaudio
pip install transformers
pip install gradio
pip install pandas numpy matplotlib scikit-learn
```

(You can also create a requirements.txt and install everything at once)

```
pip install -r requirements.txt
```

◈Step 3: Setup Runtime (Google Colab / Local)

If using Google Colab:

Go to Runtime →Change Runtime Type →Select GPU (T4 preferred)

If using Local system:

Ensure your GPU drivers + CUDA Toolkit are installed (for NVIDIA GPUs).

◈Step 4: Configure Model

Load a pre-trained AI model (example: IBM Granite or Hugging Face model):

```
from transformers import AutoTokenizer, AutoModelForCausalLM

import torch


model_name = "ibm-granite/granite-3.2-2b-instruct" # Example

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"
)
```

◈Step 5: Run EduTutorAI Interface

Launch the Gradio app for interactive use:

```
import gradio as gr

def tutor_ai(input_text):
    inputs = tokenizer(input_text,
return_tensors="pt").to(model.device)
    outputs = model.generate(**inputs, max_length=200)
    return tokenizer.decode(outputs[0],
skip_special_tokens=True)

gr.Interface(fn=tutor_ai, inputs="text", outputs="text",
title="EduTutorAI").launch()
```

👉This will open a web-based interface where users can chat with EduTutorAI.

🔷Step 6: Database & LMS Integration (Optional)

Setup MySQL/PostgreSQL database for user progress tracking.

Connect EduTutorAI with Moodle / Google Classroom API if required.

3. Verification of Installation

Run a sample query:

Input: "Explain Merge Sort in simple terms"
Output: AI-generated explanation
If the response is correct, EduTutorAI is successfully installed
🎉

✅With these steps, EduTutorAI can be installed and run either on Colab (easiest) or local system with GPU.

Conclusion

EdutorAIdemonstrateshowGenerativeAIcantransformeducati
onby
providingapersonalized,interactive,andscalablelearningexperi
ence.By
integratingwithLearningManagementSystems(LMS)andleveraging
advancedAImodels,theplatformadaptstotheuniqueneeds,pac
e,and
learningstyleofeachstudent.

Throughfeaturessuchaspersonalizedprompts,automatedassist
ance,
multilingualsupport,andreal-
timeguidance,EdutorAIbridgesthegap
betweentraditionalclassroomteachingandmodernAI-

powereddigital

learning.Itnotonlyenhancesacademicoutcomesbutalsobuildsst

udents’

criticalthinking,creativity,andproblem-solvingskills.

Forteachersandinstitutions,EdutorAIreducesworkload,improves
student
engagement,andenablesdata-drivendecision-
makingtotrackprogress
effectively.Withitsuser-
friendlydesignandrobustAIintegration,theproject
holdspotentialtobecomeascalableEdTechsolutionthatempowersb
oth
learnersandeducators.

Inessence,EdutorAIisnotjustatoolbutasteptowardthefutureofsm
art,
inclusive,andadaptiveeducation,aligningwithglobaltrendsinAI-
driven
personalizedlearning.