```
from huggingface_hub import notebook_login
notebook_login()
```

Copy a token from your Hugging Face tokens page and paste it below.

Immediately click login after copying your token or it might be stored in plain text in this notebook file.

Token:

Add token as git credential?

Login

Pro Tip: If you don't already have one, you can create a dedicated 'notebooks' token with 'write' access, that you can then easily reuse for all notebooks.

```
!pip install -q torch torchvision torchaudio
!pip install -q accelerate
!pip install -q transformers
```

```
from transformers import AutoModelForCausalLM, AutoTokenizer, set_seed
import torch

model_path="ibm-granite/granite-3.3-2b-instruct"
device="cuda"
model = AutoModelForCausalLM.from_pretrained(
        model_path,
        device_map=device,
        torch_dtype=torch.bfloat16,
    )
tokenizer = AutoTokenizer.from_pretrained(
        model_path
)

conv = [{"role": "user", "content":"Take a current environmental issue and work backward to devise an innovative prevention strategy that cou

input_ids = tokenizer.apply_chat_template(conv, return_tensors="pt", thinking=True, return_dict=True, add_generation_prompt=True).to(device)

set_seed(42)
output = model.generate(
    **input_ids,
    max_new_tokens=8192,
)

prediction = tokenizer.decode(output[0, input_ids["input_ids"].shape[1]:], skip_special_tokens=True)
print(prediction)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:104: UserWarning:
Error while fetching `HF_TOKEN` secret value from your vault: 'Requesting secret HF_TOKEN timed out. Secrets can only be fetched when
You are not authenticated with the Hugging Face Hub in this notebook.
If the error persists, please let us know by opening an issue on GitHub (https://github.com/huggingface/huggingface_hub/issues/new).
  warnings.warn(
config.json: 100%                                   787/787 [00:00<00:00, 43.4kB/s]
model.safetensors.index.json:    29.8k/? [00:00<00:00, 2.68MB/s]
Fetching 2 files: 100%                               2/2 [01:05<00:00, 65.30s/it]
model-00001-of-00002.safetensors: 100%                    5.00G/5.00G [01:04<00:00, 117MB/s]
model-00002-of-00002.safetensors: 100%                    67.1M/67.1M [00:01<00:00, 51.1MB/s]
Loading checkpoint shards: 100%                          2/2 [00:19<00:00,  8.04s/it]
generation_config.json: 100%                            132/132 [00:00<00:00, 9.09kB/s]
tokenizer_config.json:    9.93k/? [00:00<00:00, 524kB/s]
vocab.json:    777k/? [00:00<00:00, 27.6MB/s]
merges.txt:    442k/? [00:00<00:00, 16.1MB/s]
tokenizer.json:    3.48M/? [00:00<00:00, 78.3MB/s]
added_tokens.json: 100%                              207/207 [00:00<00:00, 11.8kB/s]
special_tokens_map.json: 100%                           801/801 [00:00<00:00, 59.0kB/s]
<think>To address this task, I'll choose the environmental issue of plastic pollution in our oceans as a current problem. I'll then wo

### Current Environmental Issue: Plastic Pollution in Oceans

**Problem Description:**
- Millions of tons of plastic waste enter oceans annually, causing severe harm to marine life, ecosystems, and human health.
- Plastic debris persists in the environment for centuries, breaking down into microplastics that contaminate food chains.

### Working Backward to Develop a Prevention Strategy

1. **Identify Root Causes:**
   - **Consumer Behavior:** Over-reliance on single-use plastics (e.g., bags, bottles, straws).
   - **Industrial Practices:** Lack of effective waste management systems, particularly in developing countries.
   - **Policy and Regulation:** Insufficient or inadequate environmental laws and enforcement.

2. **Develop a Prevention Strategy:**
   - **Innovative Product Design:**
     - **Biodegradable Alternatives:** Develop and promote biodegradable plastics made from renewable resources like algae, seaweed, o
     - **Reusable and Recyclable Packaging:** Encourage the use of reusable containers and packaging, and improve recycling infrastruc

   - **Consumer Education and Incentives:**
     - **Awareness Campaigns:** Launch global campaigns to educate consumers about the impact of plastic pollution on marine life and
     - **Incentive Programs:** Implement reward systems (e.g., discounts, loyalty points) for consumers who choose reusable alternativ

   - **Strengthening Waste Management Systems:**
     - **Extended Producer Responsibility (EPR):** Implement EPR policies that hold manufacturers accountable for the entire lifecycle
     - **Invest in Infrastructure:** Support developing countries in building robust waste collection, sorting, and recycling faciliti

   - **Policy and Legislation:**
     - **Bans and Restrictions:** Enact and enforce bans on single-use plastics in various applications (e.g., bags, straws, cutlery).
     - **International Agreements:** Promote international cooperation to standardize plastic waste management practices and reduce tr

3. **Implementation and Monitoring:**
   - **Collaboration:** Partner with governments, NGOs, and private sectors to implement these strategies at scale.
   - **Monitoring and Evaluation:** Establish metrics to track the reduction in plastic waste entering oceans and assess the effective

### Conclusion

By focusing on innovative product design, consumer education, strengthening waste management systems, and robust policy frameworks, it

**Problem:**
Plastic pollution in oceans is a pressing environmental issue, causing severe harm to marine life, ecosystems, and human health. Milli

**Working Backward to Develop a Prevention Strategy:**

1. **Identify Root Causes:**
   - **Consumer Behavior:** Over-reliance on single-use plastics.
   - **Industrial Practices:** Inadequate waste management systems, particularly in developing regions.
   - **Policy and Regulation:** Insufficient environmental laws and enforcement.

2. **Develop a Comprehensive Prevention Strategy:**

   - **Innovative Product Design:**
     - **Biodegradable Alternatives:** Develop and promote biodegradable plastics from renewable resources such as algae, seaweed, or
     - **Reusable and Recyclable Packaging:** Encourage the use of reusable containers and packaging, and enhance recycling infrastruc
     - **Consumer Education and Incentives:**
       - **Awareness Campaigns:** Launch global campaigns to educate consumers about the environmental impacts of plastic pollution, emp
       - **Incentive Programs:** Implement reward systems (e.g., discounts, loyalty points) for consumers who opt for reusable alternati

     - **Strengthening Waste Management Systems:**
       - **Extended Producer Responsibility (EPR):** Implement EPR policies that hold manufacturers accountable for the entire lifecycle
       - **Infrastructure Development:** Support developing countries in building robust waste collection, sorting, and recycling facili

     - **Policy and Legislation:**
       - **Bans and Restrictions:** Enact and enforce bans on single-use plastics in various applications (e.g., bags, straws, cutlery).
       - **International Cooperation:** Promote international agreements to standardize plastic waste management practices and reduce tr

  3. **Implementation and Monitoring:**
     - **Collaboration:** Partner with governments, NGOs, and private sectors to implement these strategies at scale.
     - **Monitoring and Evaluation:** Establish metrics to track the reduction in plastic waste entering oceans and assess the effective

  ### Conclusion

  By focusing on innovative product design, consumer education, strengthening waste management systems, and robust policy frameworks, it
```

```python
# EchoVerse - No API Audiobook Creator
# Complete local processing version for Google Colab

import subprocess
import sys

def install_packages():
    packages = ["gradio", "gtts", "pydub", "numpy", "requests"]
    for package in packages:
        try:
            subprocess.check_call([sys.executable, "-m", "pip", "install", package, "--quiet"])
            print(f"✅ Installed {package}")
        except:
            print(f"⚠️ Could not install {package}")

print("📦 Installing packages...")
install_packages()

import gradio as gr
import os
import json
import zipfile
import tempfile
import re
from datetime import datetime
from pathlib import Path
import threading
import time

try:
    from gtts import gTTS
    import pydub
    from pydub import AudioSegment
    TTS_AVAILABLE = True
    print("✅ TTS libraries loaded successfully")
except ImportError:
    TTS_AVAILABLE = False
    print("⚠️ TTS libraries not available")

class EchoVerseLocal:
    def __init__(self):
        self.voice_options = {
            "English (US) - Female": {"lang": "en", "tld": "us"},
            "English (US) - Male": {"lang": "en", "tld": "us"},
            "English (UK) - Female": {"lang": "en", "tld": "co.uk"},
            "English (Australia) - Female": {"lang": "en", "tld": "com.au"},
            "English (Canada) - Female": {"lang": "en", "tld": "ca"},
            "English (India) - Female": {"lang": "en", "tld": "co.in"}
        }

        self.tone_templates = {
            "dramatic": {
                "words": {"said": "declared", "walked": "strode", "looked": "gazed", "big": "enormous"},
                "prefixes": ["Suddenly", "In that moment", "Dramatically"],
                "desc": "Transform text with vivid, emotional language and dramatic flair"
            },
            "conversational": {
                "words": {"said": "mentioned", "very": "really", "because": "since"},
                "prefixes": ["You know", "Well", "So"],
                "desc": "Make text sound like a friendly conversation"
            },
            "professional": {
                "words": {"said": "stated", "showed": "demonstrated", "big": "significant"},
                "prefixes": ["Furthermore", "Additionally", "Consequently"],
                "desc": "Use formal, authoritative language with clear structure"
            },
            "mysterious": {
                "words": {"said": "whispered", "walked": "crept", "appeared": "materialized"},
                "prefixes": ["In the shadows", "Mysteriously", "Without warning"],
                "desc": "Create intrigue with atmospheric and suspenseful language"
            },
            "humorous": {
                "words": {"said": "quipped", "walked": "waddled", "big": "ridiculously huge"},
                "prefixes": ["Hilariously", "Amusingly", "Surprisingly"],
                "desc": "Add wit and light-hearted elements"
            },
            "romantic": {
                "words": {"said": "whispered softly", "looked": "gazed lovingly", "beautiful": "breathtaking"},
                "prefixes": ["Tenderly", "With heartfelt emotion", "Lovingly"],
                "desc": "Use warm, intimate language with emotional depth"
            }
        }

    def transform_text(self, text, tone):
        if tone.lower() not in self.tone_templates:
            return text, "❌ Tone not available"

        try:
            template = self.tone_templates[tone.lower()]
            transformed = text

            # Apply word replacements
            for old, new in template["words"].items():
                pattern = r'\b' + re.escape(old) + r'\b'
                transformed = re.sub(pattern, new, transformed, flags=re.IGNORECASE)

            # Add some prefixes to sentences
            sentences = re.split(r'(?<=[.!?])\s+', transformed)
            new_sentences = []

            for i, sentence in enumerate(sentences):
                if sentence.strip() and i % 3 == 0:
                    prefix = template["prefixes"][i % len(template["prefixes"])]
                    sentence = f"{prefix}, {sentence.lower()}"
                new_sentences.append(sentence)

            result = ' '.join(new_sentences)
            return result, f"✅ Text transformed with {tone} tone!"

        except Exception as e:
            return text, f"❌ Error: {str(e)}"

    def enhance_narration(self, text):
        try:
            enhanced = text
            enhanced = re.sub(r'([.!?])\s+', r'\1 [pause] ', enhanced)
            enhanced = re.sub(r'!+', r'! [excited]', enhanced)
            enhanced = re.sub(r'\?+', r'? [questioning]', enhanced)
            enhanced = re.sub(r'"([^"]+)"', r'[dialogue] "\1" [/dialogue]', enhanced)
            return enhanced.strip()
        except:
            return text

    def generate_audio(self, text, voice_config):
        if not TTS_AVAILABLE:
            return None

        try:
            clean_text = re.sub(r'\[.*?\]', '', text)
            clean_text = re.sub(r'\s+', ' ', clean_text).strip()

            if not clean_text:
                return None
            tts = gTTS(
                text=clean_text,
                lang=voice_config.get("lang", "en"),
                tld=voice_config.get("tld", "us"),
                slow=False
            )

            timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
            temp_file = f"/tmp/audio_{timestamp}.mp3"
            tts.save(temp_file)
            return temp_file

        except Exception as e:
            print(f"Audio error: {e}")
            return None

# Initialize
echo_verse = EchoVerseLocal()

# Interface Functions
def rewrite_text(text, tone):
    if not text.strip():
        return "", "❌ Please enter text to rewrite"
    return echo_verse.transform_text(text, tone)

def enhance_text(text):
    if not text.strip():
        return "❌ Please enter text to enhance"
    return echo_verse.enhance_narration(text)

def create_audio(text, voice_name, progress=gr.Progress()):
    if not text.strip():
        return None, "❌ Please enter text"

    if not TTS_AVAILABLE:
        return None, "❌ TTS not available"

    progress(0.3, desc="Generating audio...")

    voice_config = echo_verse.voice_options.get(voice_name, echo_verse.voice_options["English (US) - Female"])
    audio_file = echo_verse.generate_audio(text, voice_config)

    if audio_file:
        progress(1.0, desc="Audio complete!")
        return audio_file, "✅ Audio generated!"
    else:
        return None, "❌ Audio generation failed"

def get_tone_description(tone):
    return echo_verse.tone_templates.get(tone, {}).get("desc", "Select a tone")

def create_download(title, original, rewritten, enhanced, audio):
    try:
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")

        with tempfile.TemporaryDirectory() as temp_dir:
            project_dir = Path(temp_dir) / f"project_{timestamp}"
            project_dir.mkdir()

            if original:
                with open(project_dir / "original.txt", "w", encoding="utf-8") as f:
                    f.write(original)

            if rewritten:
                with open(project_dir / "rewritten.txt", "w", encoding="utf-8") as f:
                    f.write(rewritten)

            if enhanced:
                with open(project_dir / "enhanced.txt", "w", encoding="utf-8") as f:
                    f.write(enhanced)

            if audio and os.path.exists(audio):
                import shutil
                shutil.copy(audio, project_dir / "audio.mp3")

            zip_path = f"/tmp/project_{timestamp}.zip"
            with zipfile.ZipFile(zip_path, 'w') as zipf:
                for file_path in project_dir.rglob("*"):
```

```
                if file_path.is_file():
                    zipf.write(file_path, file_path.relative_to(project_dir))

            return zip_path
    except:
        return None

# Create Interface
def create_interface():
    with gr.Blocks(theme=gr.themes.Soft(), title="EchoVerse - No API") as app:

        gr.HTML("""
        <div style="text-align: center; padding: 20px; background: linear-gradient(135deg, #28a745 0%, #20c997 100%); color: white; border-ra
            <h1>🎧 EchoVerse - Local Audiobook Creator</h1>
            <p>Transform text into audiobooks - No API required!</p>
            <div style="background: rgba(255,255,255,0.2); display: inline-block; padding: 5px 15px; border-radius: 20px; margin: 5px;">
                ✅ NO SETUP REQUIRED
            </div>
            <div style="background: rgba(255,255,255,0.2); display: inline-block; padding: 5px 15px; border-radius: 20px; margin: 5px;">
                🔒 COMPLETELY LOCAL
            </div>
        </div>
        """)

        with gr.Tabs():
            with gr.TabItem("📝 Text Processing"):
                with gr.Row():
                    with gr.Column():
                        gr.Markdown("### Original Text")
                        original_input = gr.Textbox(
                            label="Input Text",
                            placeholder="Enter your text here...",
                            lines=10
                        )

                        with gr.Row():
                            tone_selector = gr.Dropdown(
                                choices=list(echo_verse.tone_templates.keys()),
                                label="Tone Style",
                                value="conversational"
                            )

                        tone_desc = gr.Textbox(
                            label="Tone Description",
                            interactive=False,
                            value=echo_verse.tone_templates["conversational"]["desc"]
                        )

                        transform_btn = gr.Button("✨ Transform Text", variant="primary")
                        status = gr.Textbox(label="Status", interactive=False)

                    with gr.Column():
                        gr.Markdown("### Transformed Text")
                        transformed_output = gr.Textbox(
                            label="Transformed Text",
                            lines=10
                        )

                        enhance_btn = gr.Button("🎙 Add Narration Cues")
                        enhanced_output = gr.Textbox(
                            label="Enhanced Text",
                            lines=6
                        )

            with gr.TabItem("🎵 Audio Generation"):
                with gr.Row():
                    with gr.Column():
                        audio_text = gr.Textbox(
                            label="Text for Audio",
                            lines=8,
                            placeholder="Text will auto-populate from transformation..."
                        )

                        voice_select = gr.Dropdown(
                            choices=list(echo_verse.voice_options.keys()),
                            label="Voice",
                            value="English (US) - Female"
                )

                audio_btn = gr.Button("🎵 Generate Audio", variant="primary")
                audio_status = gr.Textbox(label="Audio Status", interactive=False)

            with gr.Column():
                audio_output = gr.Audio(label="Generated Audio", type="filepath")

    with gr.TabItem("📦 Download"):
        with gr.Row():
            with gr.Column():
                project_title = gr.Textbox(label="Project Title", placeholder="My Audiobook")
                download_btn = gr.Button("📦 Create Package", variant="primary")
                download_file = gr.File(label="Download", visible=False)

            with gr.Column():
                gr.Markdown("""
                ### Package Contents:
                - Original text file
                - Transformed text file
                - Enhanced text with cues
                - Generated audio file
                """)

    with gr.TabItem("❓ Help"):
        gr.Markdown("""
        # EchoVerse Local - Quick Guide

        ## How to Use:
        1. **Text Processing**: Enter your text and choose a tone style
        2. **Transform**: Click "Transform Text" to apply the tone
        3. **Enhance**: Add professional narration cues
        4. **Audio**: Generate high-quality audio from your text
        5. **Download**: Get complete project package

        ## Available Tones:
        - **Dramatic**: Emotional, vivid language
        - **Conversational**: Friendly, casual tone
        - **Professional**: Formal, authoritative
        - **Mysterious**: Atmospheric, suspenseful
        - **Humorous**: Light-hearted, witty
        - **Romantic**: Warm, intimate language

        ## Features:
        - ✅ No API keys required
        - ✅ Works completely offline
        - ✅ High-quality Google TTS
        - ✅ Professional narration cues
        - ✅ Complete project packages

        Ready to create your audiobook? Start with the Text Processing tab!
        """)

# Event handlers
transform_btn.click(
    fn=rewrite_text,
    inputs=[original_input, tone_selector],
    outputs=[transformed_output, status]
)

enhance_btn.click(
    fn=enhance_text,
    inputs=[transformed_output],
    outputs=[enhanced_output]
)

audio_btn.click(
    fn=create_audio,
    inputs=[audio_text, voice_select],
    outputs=[audio_output, audio_status]
)

tone_selector.change(
    fn=get_tone_description,
    inputs=[tone_selector],
    outputs=[tone_desc]
)
```

```
        transformed_output.change(
            lambda x: x,
            inputs=[transformed_output],
            outputs=[audio_text]
        )

        def handle_download(title, original, transformed, enhanced, audio):
            zip_file = create_download(title, original, transformed, enhanced, audio)
            if zip_file:
                return zip_file, gr.update(visible=True)
            return None, gr.update(visible=False)

        download_btn.click(
            fn=handle_download,
            inputs=[project_title, original_input, transformed_output, enhanced_output, audio_output],
            outputs=[download_file, download_file]
        )

    return app

# Launch
if __name__ == "__main__":
    print("=" * 50)
    print("🎧 ECHOVERSE - LOCAL MODE")
    print("=" * 50)
    print("✅ No API required")
    print("🔒 Completely local processing")
    print("🚀 Ready to launch...")

    app = create_interface()

    try:
        demo = app.launch(
            share=True,
            inbrowser=False,
            server_name="0.0.0.0",
            show_error=True
        )

        print("✅ EchoVerse is running!")
        print("🌐 Use the public URL above or the embedded interface")
        print("🎯 No setup required - start creating audiobooks now!")

    except Exception as e:
        print(f"❌ Launch error: {e}")
        print("🔧 Try restarting your runtime")
```