User Guide for TTS Service Using Pre-trained Mongolian Voices

Overview

This guide outlines how to utilize our Text-to-Speech (TTS) service, featuring pre-trained Mongolian celebrity voices. The service, built on FastAPI, is designed to operate efficiently on both CPU and GPU environments. While functional on CPUs, deploying on a GPU (like an NVIDIA T4 or equivalent) is recommended for faster inference and better scalability.

System Requirements

- **Hardware**:
    - Compatible with both CPU and GPU.
    - For enhanced performance and scalability, a GPU (such as NVIDIA T4 or any other compatible GPU) is recommended.
- **Codebase**: Includes **main.py** (FastAPI application), a **weights** folder with pre-trained models, and a **requirements.txt** file.
- **Operating System**: Linux (preferably Ubuntu) with Python 3.6 or higher.

Setting Up the Service

Prepare the Environment:

- Ensure you have the complete codebase, including the weights folder and requirements.txt.
- Install Python dependencies:

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

Start the FastAPI Server:

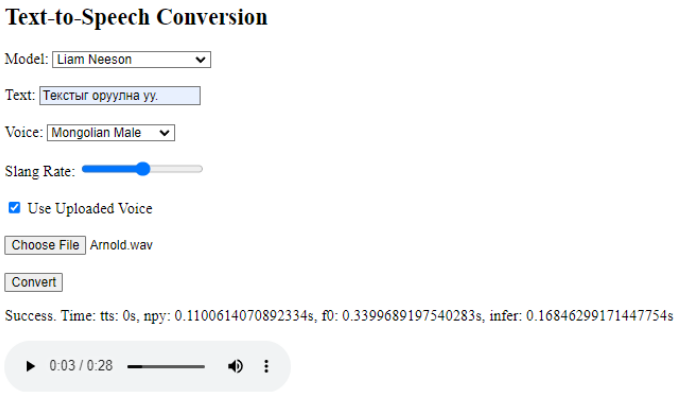
- Launch the service using:

```
- uvicorn main:app --host 0.0.0.0 --port 8000
```

Using the Service

Web Interface

- Access the service at http://<Server-IP>:8000.
- The web interface allows easy interaction: select a voice model, input text, and perform TTS conversion.



Steps for TTS Conversion

1. **Choose a Voice Model**: Select from the available list of Mongolian celebrity voices.
2. **Enter Your Text**: Input the Mongolian text you wish to convert into speech.
3. Voice Upload: Check the 'Use Uploaded Voice' option and upload a voice file.
4. **TTS Conversion**: Click 'Convert' to process your text. The service will generate and play back the speech.
5. **Download Option**: Users can download the audio file for offline use.
6. **Receiving the Output:**

   The converted speech will be played back on the web interface.

   A base64-encoded audio string is also available in the response for API usage.

API Access via curl

```
curl -X 'POST' \
  'http://<Server-IP>:8000/convert' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'model_name=<ModelName>' \
  -F 'tts_text=<TextToConvert>' \
  -F 'selected_voice=<SelectedVoice>' \
  -F 'slang_rate=<RateValue>' \
  -F 'use_uploaded_voice=<true/false>' \
  -F 'voice_upload=@path_to_audio_file;type=audio/wav'
```

- Replace **<Server-IP>**, **<ModelName>**, **<TextToConvert>**, etc., as necessary.

Note

- Ensure the weights folder with pre-trained models is in the same directory as main.py.
- Modify the model_name, tts_text, and other parameters as per your requirement.

Output

The service returns a JSON response with a base64-encoded audio string, which can be decoded to play the converted speech.

Additional Notes

- This TTS service specifically caters to the Mongolian language using pre-trained celebrity voice models.
- While operable on a CPU, using a GPU is advised for handling higher workloads and achieving quicker response times.

Pricing

For deploying TTS service using T4 GPU instances on AWS, the hourly pricing for different instance types (as of the latest available information) is as follows:

- **g4dn.xlarge**: $0.526 per hour (Single GPU, 4 vCPUs, 16 GiB memory)
- **g4dn.2xlarge**: $0.752 per hour (Single GPU, 8 vCPUs, 32 GiB memory)
- **g4dn.4xlarge**: $1.204 per hour (Single GPU, 16 vCPUs, 64 GiB memory)
- **g4dn.8xlarge**: $2.176 per hour (Single GPU, 32 vCPUs, 128 GiB memory)
- **g4dn.16xlarge**: $4.352 per hour (Single GPU, 64 vCPUs, 256 GiB memory)
- **g4dn.12xlarge**: $3.912 per hour (Multi GPU, 48 vCPUs, 192 GiB memory)
- **g4dn.metal**: $7.824 per hour (Multi GPU, 96 vCPUs, 384 GiB memory)

FastAPI application, **main.py**, along with the pre-trained models in the **weights** folder, can be deployed on these instances. For interaction with your service, users can use the HTML UI provided in **main.py** or the API through **curl** commands. While the service will work on a CPU, using a GPU is recommended for faster inference and better handling of multiple concurrent requests. The choice of the instance will depend on the expected workload and the performance requirements of the application.