description for main.py :

This script is the main entry point for generating a speech quality dataset with different distortions. It imports the generate\_dataset function from the dataset\_generator module and provides the necessary input parameters such as the source folder, output folder, noise folder, and the number of files to generate. It also includes an example condition table defining the different distortions and their settings. The generate\_dataset function is then called with the provided parameters to generate the dataset.

description for dataset\_generator.py:

**Speech Quality Dataset Generator**

The dataset\_generator.py script is used to generate a speech quality dataset with different distortions applied to clean speech signals. The script applies various distortions, such as packet loss, background noise, white noise, lowpass filtering, and clipping, to create a diverse set of distorted speech signals for evaluation.

**Dependencies**

The script relies on the following libraries:

"os ": Provides a way to interact with the operating system and manage file paths.

"Pandas ": Used for creating and manipulating data structures and analysis tools.

"pydub ": Enables audio file manipulation and processing.

"distortions ": A module containing functions to apply different types of distortions.

Make sure these libraries are installed before running the script.

**Usage**

To use the script, follow these steps:

Create a clean speech signal file in WAV format and specify its path using the clean\_path variable in the generate\_dataset function.

Prepare a noise file in WAV format and specify its path using the noise\_path variable in the generate\_dataset function.

Set the output directory path using the output\_dir variable in the generate\_dataset function. The distorted files will be saved in this directory.

Specify the number of files to generate using the num\_files variable in the generate\_dataset function.

Adjust the distortion parameters in the generate\_dataset function as needed. The example parameters provided can be modified to suit your specific requirements.

Run the script.

The script will generate a set of distorted speech files based on the specified parameters. Each distortion type will produce multiple variations based on the provided parameter values.

The conditions for each generated file, including the distortion type and parameter values, will be saved in an Excel file named conditions.xlsx. This file will be saved in the same directory as the distorted files.

**Code Structure**

The script consists of the following components:

write\_conditions\_to\_excel: This function writes the conditions for the dataset to an Excel file. It takes the output directory, number of files, codec names, packet loss rates, background noise levels, white noise levels, filter cutoff frequencies, and clipping thresholds as input. It generates a list of conditions based on these inputs and saves them in an Excel file named conditions.xlsx.

generate\_dataset: This function generates the speech quality dataset with different distortions. It takes the clean speech file path, noise file path, output directory path, and number of files as input. It applies various distortions to the clean speech file using different codecs and distortion parameters. The distorted files are saved in the output directory, and the conditions for each generated file are written to an Excel file using the write\_conditions\_to\_excel function.

Distortion Functions: The script imports distortion functions from a separate module named distortions. These functions implement the specific distortions applied to the speech signals, such as packet loss, background noise, white noise, lowpass filtering, and clipping. These functions are called within the generate\_dataset function to apply the distortions to the clean speech signals.

description for distortions.py:

Introduction:

The distortions.py module provides functions to apply various distortions to audio signals using the pydub library. The code contains functions for adding packet loss, background noise, white noise, low-pass filtering, clipping, and applying different audio codecs to an audio signal.

Functions:

add\_packet\_loss(signal, loss\_rate):

Description: This function introduces packet loss to an audio signal by setting a portion of the signal samples to zero.

Parameters:

signal: The input audio signal as an instance of the AudioSegment class.

loss\_rate: The proportion of samples to set to zero, specified as a value between 0 and 1.

Returns: The modified audio signal with packet loss.

add\_background\_noise(signal, noise\_path, noise\_level):

Description: This function adds background noise to an audio signal by overlaying a noise signal.

Parameters:

signal: The input audio signal as an instance of the AudioSegment class.

noise\_path: The path to the background noise audio file.

noise\_level: The desired level of background noise relative to the input signal.

Returns: The audio signal with added background noise.

add\_white\_noise(signal, noise\_level):

Description: This function adds white noise to an audio signal by overlaying a silent white noise signal.

Parameters:

signal: The input audio signal as an instance of the AudioSegment class.

noise\_level: The desired level of white noise relative to the input signal.

Returns: The audio signal with added white noise.

apply\_lowpass\_filter(signal, cutoff\_freq):

Description: This function applies a low-pass filter to an audio signal, allowing frequencies below the cutoff frequency to pass through and attenuating frequencies above it.

Parameters:

signal: The input audio signal as an instance of the AudioSegment class.

cutoff\_freq: The cutoff frequency for the low-pass filter in Hertz (Hz).

Returns: The filtered audio signal.

apply\_clipping(signal, threshold):

Description: This function applies clipping to an audio signal by setting samples above the threshold to the maximum level, effectively limiting the signal amplitude.

Parameters:

signal: The input audio signal as an instance of the AudioSegment class.

threshold: The clipping threshold in decibels (dB) relative to the maximum level of the signal.

Returns: The clipped audio signal.

apply\_codec(signal, codec):

Description: This function applies a specific audio codec to an audio signal using the ffmpeg library.

Parameters:

signal: The input audio signal as an instance of the AudioSegment class.

codec: The desired audio codec to apply. Supported codecs include 'amrnb', 'amrwb', 'evs', 'opus', 'g711', and 'g722'.

Returns: The audio signal encoded with the specified codec.

Dependencies:

The numpy library is used for array manipulation.

The pydub library is used for audio processing.

The subprocess module is used for executing external commands to apply codecs using ffmpeg.

Usage:

To use the functions in this module, you need to import it and have the necessary dependencies installed.