Audio_silence_Trimming

```
#!/bin/sh
import os
from os import listdir
import argparse
from glob import glob
import subprocess
import shutil

from pydub import AudioSegment
import librosa
import soundfile as sf

import pandas as pd
from logger import get_logger

logger = get_logger(__name__)

from pydub import AudioSegment

def trim_silence(filename, output_path):
""" Trim silent parts with a threshold and 0.01 sec margin """
try:
wav, sample_rate = librosa.load(filename)

trim_db = 30
frame_length_ms = 50
frame_shift_ms = 12.5

factor = frame_length_ms / frame_shift_ms
assert (factor).is_integer(), " [!] frame_shift_ms should divide frame_length_ms"
hop_length = int(frame_shift_ms / 1000.0 * sample_rate)
win_length = int(hop_length * factor)

margin = int(sample_rate * 0.01)
wav = wav[margin:-margin]
wav2 = librosa.effects.trim(wav, top_db=trim_db, frame_length=win_length,
hop_length=hop_length)[0]
```

```
_, fname = os.path.split(filename)

output_filename = os.path.join(output_path, fname)

# /home/gnani/Downloads/wetransfer_final-hindi-rework_2021-09-
08_2035/second_version/trimmed/utter_ask_less_seven.wav

logger.info("----------> "+str(output_filename))
sf.write(output_filename, wav2, sample_rate)
return True

except Exception as e:
logger.info(f"Exception {filename} file when silence_adding_start_end: "+str(e))
return False

def silence_adding_start_end(FILE):
try:
logger.info(f"Executing silence_adding_start_end...{FILE}")
one_sec_segment = AudioSegment.silent(duration=silent)  #duration in milliseconds
song = AudioSegment.from_wav(FILE)
final_song = one_sec_segment + song + one_sec_segment
final_song.export(FILE, format="wav")
return True
except Exception as e:
logger.info(f"Exception {FILE} file when silence_adding_start_end: "+str(e))
return False

def validate_filename(files):
for file in files:
if ' ' in file:
new_file = file.replace(' ','_')
os.rename(file, new_file)

def other_format_to_wav(file_path):
try:
logger.info("Executing other_format_to_wav...")
dir, filename = os.path.split(file_path)
output_wav = os.path.join(dir, filename[:-4]+".wav")

logger.info('INPUT FILE: '+str(file_path))
logger.info('OUTPUT FILE: '+str(output_wav))
```

```
temp = os.path.join(dir, 'temp.wav')

os.system(f'sox {file_path} -c 1 -r 8000 -b 16 {temp}')
output_wav = file_path[:-4]+".wav"
os.system(f'sox -c 1 -r 8000 -e signed-integer -b 16 {temp} {output_wav}')
os.system(f'rm {temp}')

# os.system(f'ffmpeg {file_path} -ac 1 -ar 16000 -acodec pcm_s16le {output_wav} -y')
return output_wav
except Exception as e:
logger.info(f"Exception {file_path} file when convert another format to wav format: "+str(e))
return None

def get_audio_details(file_path):
try:
logger.info(f"Executing get_audio_details...{file_path}")
audio = AudioSegment.from_file(file_path)
no_of_channel, sample_rate, duration, decibel = audio.channels, audio.frame_rate,
audio.duration_seconds, audio.dBFS
return [no_of_channel, sample_rate, duration, decibel]
except Exception as e:
logger.info(f"Exception {file_path} file when get_audio_details: "+str(e))
return [None, None, None, None]

def gain_check(file_dir):
try:
try:
logger.info("Executing gain_check...")
command2 = 'for f in dir2; do echo $f; sox $f -n stats 2>&1 | grep "Max level" ; done 2>&1 | tee
gain.txt'.replace('dir2', file_dir)
ret2 = subprocess.run(command2, capture_output=True, shell=True)
logger.info(ret2.stdout.decode())
except Exception as e:
logger.info("Max Leave Error:"+str(e))

gain_ls = [g.strip() for g in open('gain.txt', 'r').readlines()]

# logger.info("GAIN TEXT: "+str(gain_ls))

key, value = [], []
for i, val in enumerate(gain_ls):
if i % 2 != 0:
```

```
vals = val.split('   ')
value.append(vals[1])
else:
key.append(val)

gain_dict = {k:v for k,v in zip(key, value)}
logger.info("Exiting gain_check...")
return gain_dict
except Exception as e:
logger.info(f"Exception {file_dir} directory when generate gain dictionary: "+str(e))
return {}

def gain_adjust(file_path, gain_value, gain_dir):
try:
logger.info(f"Executing gain_adjust...{file_path}")

_, file = os.path.split(file_path)
ouput = os.path.join(gain_dir, file)

gain_value = float(gain_value)

if 0.35 < gain_value < 0.50:
os.system(f'sox {file_path} {ouput} gain 3')
elif 0.10 < gain_value < 0.35:
os.system(f'sox {file_path} {ouput} gain 6')
elif gain_value > 0.99:
os.system(f'sox {file_path} {ouput} gain -3')
else:
os.system(f'cp {file_path} {gain_dir}')
return True

except Exception as e:
logger.info(f"Exception for {file_path} when adjust gain: "+str(e))
return False

def sox_silence_trimming(directory, audio_type = ""):
# 1 0 1% short audios
# 1 0.1 1% long audios
# Reference: https://digitalcardboard.com/blog/2009/08/25/the-sox-of-silence/
if audio_type == "long":
logger.info(f"Executing sox_silence_trimming...long")
try:
```

```
command1 = 'for f in directory; do sox "$f" temp.wav silence 1 0.1 1% reverse; sox temp.wav
"$f" silence 1 0.1 1% reverse;rm temp.wav;done'.replace('directory', directory)
ret1 = subprocess.run(command1, capture_output=True, shell=True)
logger.info(ret1.stdout.decode())
except Exception as e:
logger.info("Trimming Part Error:"+str(e))

elif audio_type == "short":
logger.info(f"Executing sox_silence_trimming...short")
try:
command1 = 'for f in directory; do sox "$f" temp.wav silence 1 0 1% reverse; sox temp.wav
"$f" silence 1 0 1% reverse;rm temp.wav;done'.replace('directory', directory)
ret1 = subprocess.run(command1, capture_output=True, shell=True)
logger.info(ret1.stdout.decode())
except Exception as e:
logger.info("Trimming Part Error:"+str(e))

def raw_file_generate(files_list, temp):
logger.info(f"Executing raw_file_generate...")
for file in files_list:
os.system(f'sox {file} -c 1 -r 8000 -b 16 {temp}')
raw = file[:-4]+".raw"
os.system(f'sox -c 1 -r 8000 -e signed-integer -b 16 {temp} {raw}')
os.system(f'rm {temp}')

def make_dir(directory):
if not os.path.exists(directory):
os.makedirs(directory)

def clean_empty_folders(root):
folders = list(os.walk(root))[1:]

for folder in folders:
# folder example: ('FOLDER/3', [], ['file'])
if not folder[2]:
os.rmdir(folder[0])

if __name__ == '__main__':
parser = argparse.ArgumentParser()
parser.add_argument("--input_dir", help="input audio file path")
parser.add_argument("--output_dir", help="output audio file path")
parser.add_argument("--audio_format", help="audio extention format Ex. mp3, wav")
```

```
args = parser.parse_args()

input_dir, output_dir, audio_format = args.input_dir, args.output_dir, args.audio_format

if " " in input_dir:raise Exception(f"Invalid input_dir File Path. Please make sure, folder name without empty space. {input_dir}")
if " " in output_dir:raise Exception(f"Invalid output_dir File Path. Please make sure, folder name without empty space. {output_dir}")

clean_empty_folders(output_dir)

data = pd.DataFrame()
make_dir(output_dir)

############

short_audio_len = 1.0 # in secs
valid_sample_rate = 8000
valid_channel = 1
silent = 150 #duration in milliseconds

############
try:
logger.info(f"Executing phase 1 [other format to proper wav audio format]")

search_files1 = os.path.join(input_dir, f"*.{audio_format}")
# files1 = listdir(input_dir)
# data["filename"] = list(map(lambda x: os.path.join(input_dir,x), files1))

files1 = glob(search_files1)
validate_filename(files1)
files1 = glob(search_files1)

data["filename"] = files1

if audio_format != "wav":
data["filename"] = data["filename"].apply(other_format_to_wav)

data["no_of_channel"], data["sample_rate"], data["duration"], data["decibel"] = zip(*data["filename"].apply(get_audio_details))
data['audio_type'] = data['duration'].apply(lambda x: "long" if x > short_audio_len else "short")
```

```
improper_audios = data[(data['no_of_channel'] > valid_channel) | (data['sample_rate'] !=
valid_sample_rate)]
improper_audios["filename"] = improper_audios["filename"].apply(other_format_to_wav)

except Exception as e:
logger.info(f"Exception in phase 1: "+str(e))

logger.info(list(data['filename'].head(3)))

try:
logger.info(f"Executing phase 2 [generate gain dictionary and increasing gain based in gain
value]")

gain_dict = gain_check(search_files1)

gain_dir = os.path.join(output_dir, "gain_adjusted_1st_time")
make_dir(gain_dir)

if len(gain_dict):
try:
data['gain_value_1st'] = data["filename"].replace(gain_dict)
except Exception as e:
logger.info("------> "+str(e))

try:
for i in range(len(data)):
file_path, gain_value = data.iloc[i][['filename', 'gain_value_1st']]
data.loc[i, 'gain_adjust_1st'] = gain_adjust(file_path, gain_value, gain_dir)
except Exception as e:
logger.info("=======> "+str(e))

except Exception as e:
logger.info(f"Exception in phase 2: "+str(e))

logger.info(list(data['filename'].head(3)))

try:
logger.info(f"Executing phase 3 [increasing gain again based in gain value < 0.5]")

search_files2 = os.path.join(gain_dir, "*.wav")
files2 = listdir(gain_dir)
```

```
regain_dict = gain_check(search_files2)
regain_dir = os.path.join(output_dir, "gain_adjusted_2nd_time")
make_dir(regain_dir)

_, data["filename"] = zip(*data["filename"].apply(os.path.split))
data["filename"] = list(map(lambda x: os.path.join(gain_dir,x), data.filename))

logger.info(list(data['filename'].head(3)))

if len(regain_dict):
data['gain_value_2nd'] = data["filename"].replace(regain_dict)

for i in range(len(data)):
file_path, gain_value = data.iloc[i][['filename', 'gain_value_2nd']]
data.loc[i, 'gain_adjust_2nd'] = gain_adjust(file_path, gain_value, regain_dir)

except Exception as e:
logger.info(f"Exception in phase 3: "+str(e))

search_files3 = os.path.join(regain_dir, "*.wav")

logger.info(list(data['filename'].head(3)))

try:
logger.info("Executing phase 4 [Trimming silence]")

trimmed_dir = os.path.join(output_dir, "trimmed")
make_dir(trimmed_dir)

for i, filename in enumerate(data["filename"].values.tolist()):
logger.info("==================> "+str(filename))

data.loc[i, "trimming"] = trim_silence(filename, trimmed_dir)

except Exception as e:
logger.info(f"Exception in phase 4: "+str(e))

logger.info(f"Executing phase 5 [raw_file_generate]")

try:
files_list, tempfile_path = glob(os.path.join(trimmed_dir, "*.wav")), os.path.join(trimmed_dir,
'temp.wav')
```

```
raw_file_generate(files_list, tempfile_path)

except Exception as e:
logger.info(f"Exception in phase 5: "+str(e))

# logger.info(f"Executing phase 6 [silence_adding_start_end]")

# try:
#     data['silence_added'] = data['filename'].apply(silence_adding_start_end)
# except Exception as e:
#     logger.info(f"Exception in phase 6: "+str(e))

# data['trimming'] = [True] * len(data)

data.to_csv("AudioTrimming_metadata.csv", index=False)
logger.info(list(data['filename'].head(3)))

logger.info("Done!!!")

# try:
#     logger.info(f"Executing phase 4 [silence trimming for short and long audios]")

#     short_trimming_dir, long_trimming_dir, trimmed_dir = os.path.join(output_dir,
"short_trimmed"), os.path.join(output_dir, "long_trimmed"), os.path.join(output_dir, "trimmed")
#     make_dir(short_trimming_dir); make_dir(long_trimming_dir); make_dir(trimmed_dir);

#     _, data["filename"] = zip(*data["filename"].apply(os.path.split))
#     data["filename"] = list(map(lambda x: os.path.join(regain_dir,x), data.filename))

#     logger.info(list(data['filename'].head(3)))

#     try:
#         short_audios, long_audios = list(data[data['audio_type'] == 'short']['filename']),
list(data[data['audio_type'] == 'long']['filename'])

#         logger.info("Short Audios: "+str(len(short_audios)))
#         logger.info("Long Audios: "+str(len(long_audios)))

#         if len(short_audios):
#             logger.info(short_audios[0])
#             sf = os.path.join(short_trimming_dir, "*.wav")
```

```
#             [os.system(f'cp {x} {short_trimming_dir}') for x in short_audios]
#             sox_silence_trimming(sf, audio_type = "short");

#             os.system(f'cp -a {sf} {trimmed_dir}')

#         if len(long_audios):
#             logger.info(long_audios[0])
#             sf = os.path.join(long_trimming_dir, "*.wav")

#             [os.system(f'cp {x} {long_trimming_dir}') for x in long_audios]
#             sox_silence_trimming(sf, audio_type = "long")

#             os.system(f'cp -a {sf} {trimmed_dir}')

#         _, data["filename"] = zip(*data["filename"].apply(os.path.split))
#         data["filename"] = list(map(lambda x: os.path.join(trimmed_dir,x), data.filename))

#     except Exception as e:
#         logger.info(f"Exception in phase 4: "+str(e))


# except Exception as e:
#     logger.info(f"Exception in phase 4: "+str(e))
```