git remote set-url origin https://pkailin:ghp\_g47v6N1Ln4UjOuE7KWBRXNGnpd6dfq0hCfTD@github.com/pkailin/SPAPL\_KidsASR.git

**Setting up the Python Environment:**

conda create --name spapl\_kidsasr python=3.9.7

in venv:

pip install torch transformers==4.32.1 accelerate==0.20.3 datasets soundfile librosa numpy evaluate dataclasses-json jiwer pygments ffmpeg openai-whisper whisper-normalizer

add to path (if warning):

nano ~/.bashrc

add this line: export PATH=$PATH:/home/klp65/.local/bin

source ~/.bashrc

**Kaldi Installation:**

a) Clone Kaldi repository

git clone <https://github.com/kaldi-asr/kaldi.git>

instructions in <https://github.com/kaldi-asr/kaldi/blob/master/INSTALL>

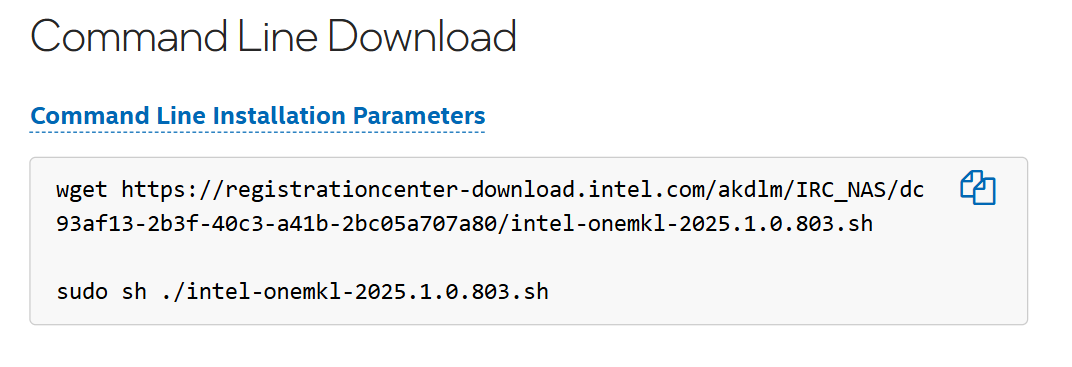
b) Build the tools (including sclite) – instructions in <https://github.com/kaldi-asr/kaldi/blob/master/tools/INSTALL>

cd kaldi/tools

extras/check\_dependencies.sh

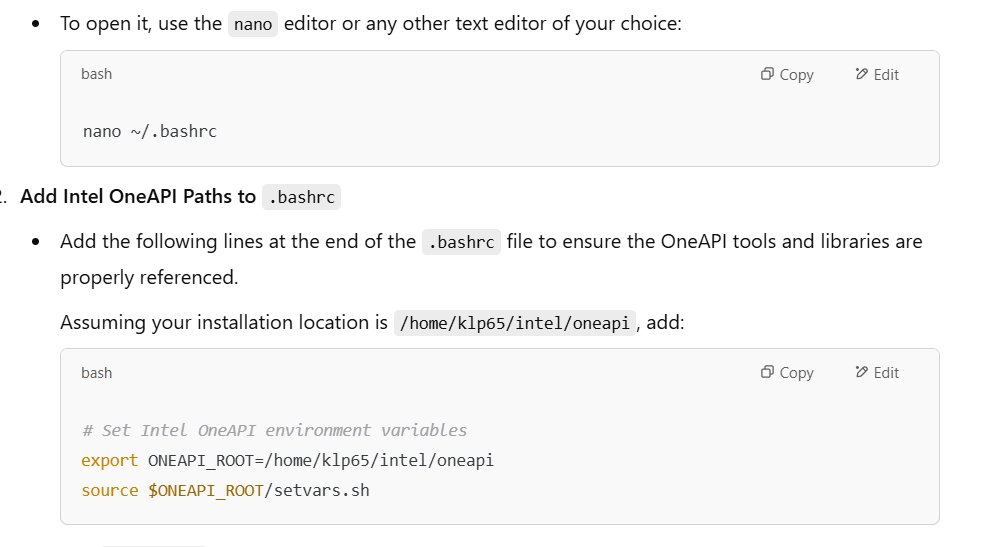
reported missing MKL:

<https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl-download.html?operatingsystem=linux&linux-install=online>

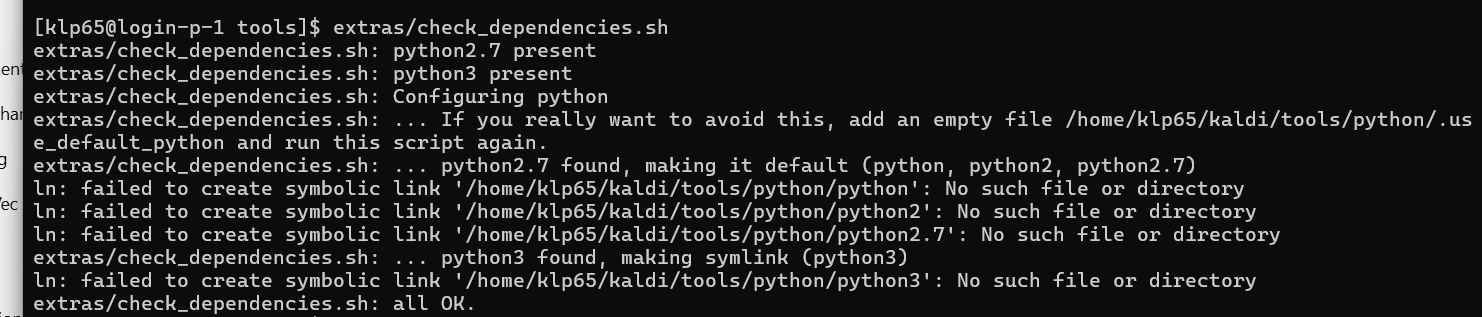


Installation location will show after successful installation. No need for sudo.

After you’re done, check installation location and update bashrc and do source ~/.bashrc after



Make sure all ok:



Openfst issue: (downgrade to 1.6.7 instead of 1.8.4)

cd /home/klp65/kaldi/tools

wget http://www.openfst.org/twiki/pub/FST/FstDownload/openfst-1.6.7.tar.gz

tar -xzf openfst-1.6.7.tar.gz

cd openfst-1.6.7

./configure --prefix=/home/klp65/kaldi/tools/openfst --enable-shared --enable-ngram-fsts --enable-lookahead-fsts LDFLAGS="-pthread"

make -j4

cd ..

**in Makefile change the OPENFST VERSION:**

****

after that: in tools

make clean

make

\*\* remember to “make clean” before “make” if you changed compiler

Ensure GCC is being used:

export CC=gcc

export CXX=g++

c) build the source -- [follow instructions in https://github.com/kaldi-asr/kaldi/blob/master/src/INSTALL](https://github.com/kaldi-asr/kaldi/blob/master/src/INSTALL)

cd ../src

./configure --shared

make depend -j 8

make -j 8

**include PATH for sclite:**

(spapl\_kidsasr) [klp65@login-q-1 OGI]$ find /home/klp65/kaldi -name "sclite"

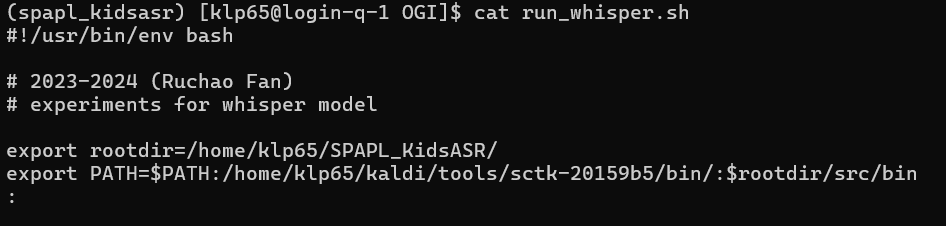
/home/klp65/kaldi/tools/sctk-20159b5/bin/sclite

/home/klp65/kaldi/tools/sctk-20159b5/src/sclite

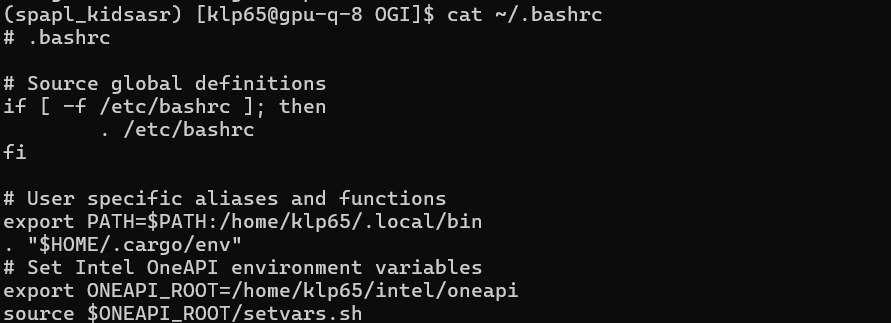
/home/klp65/kaldi/tools/sctk-20159b5/src/sclite/sclite

(spapl\_kidsasr) [klp65@login-q-1 OGI]$ export PATH=$PATH:/home/klp65/kaldi/tools/sctk-20159b5/bin

**Change paths at the top for run\_whisper.sh:**

****

**Bashrc file:**

****

**Data format in OGI Kids:**

<https://github.com/OSU-slatelab/OGI-kids-phoneme-recognition/blob/main/ogi_prepare.py>

.wav files are in:

{data\_folder}/speech/scripted/

transcription files are in:

with open(train\_align\_file) as f:

train\_alignments = json.load(f)

with open(valid\_align\_file) as f:

valid\_alignments = json.load(f)

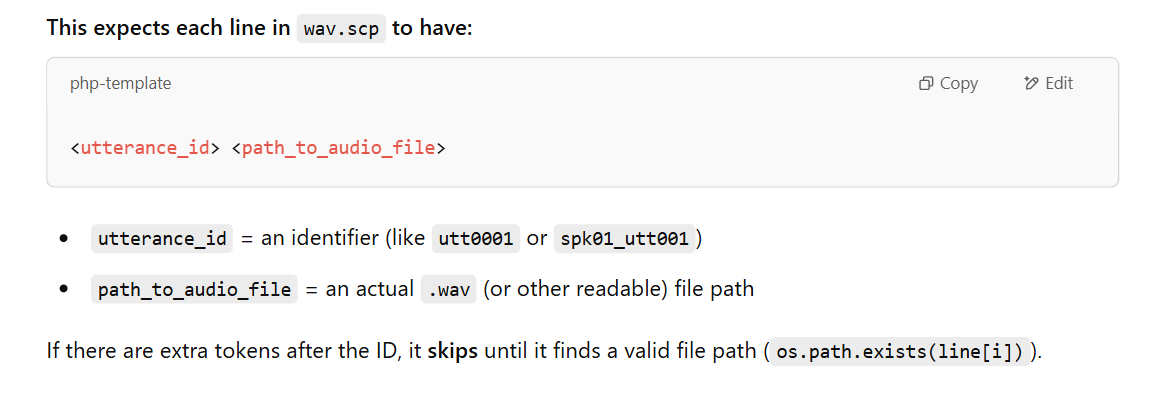
JSON files contain alignments —> probably mappings from sentence ID → transcribed words

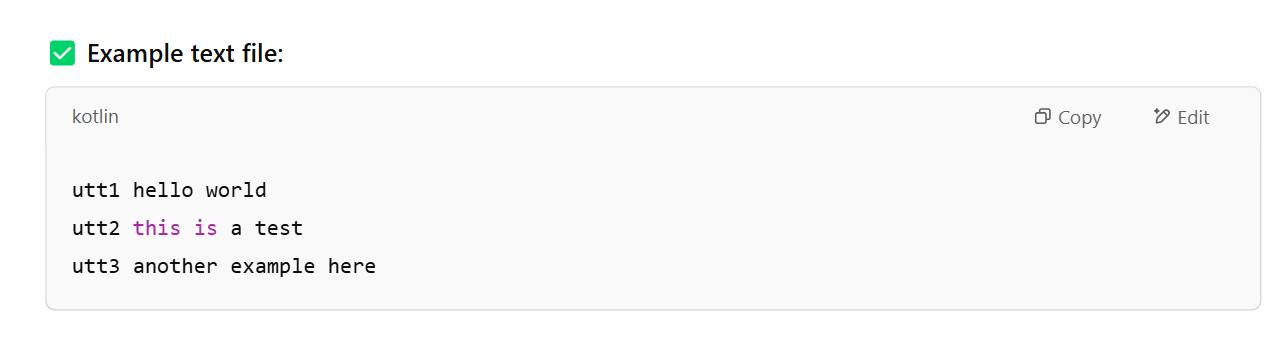
**Data accepted in code:**

wav.scp file: connects every utterance (sentence said by one person during particular recording session) with an audio file related to this utterance.

/src/data/whisper\_loader.py: loads dataset for Whisper, does data augmentation based on arguments in whisper\_small\_train.yaml

From whisper\_loader.py:





**For OGI dataset:**

Stage 1/3: Evaluation of Whisper (in run\_whisper.sh)

/egs/OGI/data/dev/wav.scp

/egs/OGI/data/dev/text

/egs/OGI/data/test/wav.scp

/egs/OGI/data/test/text

/egs/OGI/data/spont\_al/wav.scp

/egs/OGI/data/spont\_al/text

Stage 2: Finetuning of Whisper (in whisper\_small\_train.yaml)

Training data:

/egs/OGI/data/train/wav.scp

/egs/OGI/data/train/text

Validation data:

/egs/OGI/data/dev/wav.scp

/egs/OGI/data/dev/text

tasks, both OGI and MyST:

1. stage 1: evaluation of baseline whisper model

2. run stage 2 and 3: full-finetuning without data augmentation

3. run stage 2 and 3: full-finetuning with VTLP

4. run stage 2 and 3: full-finetuning with SP

5. run stage 2 and 3: full-finetuning with PP

6. run stage 2 and 3: full-finetuning with SA

try combinations:

1. SA + PP

2. SA + SP

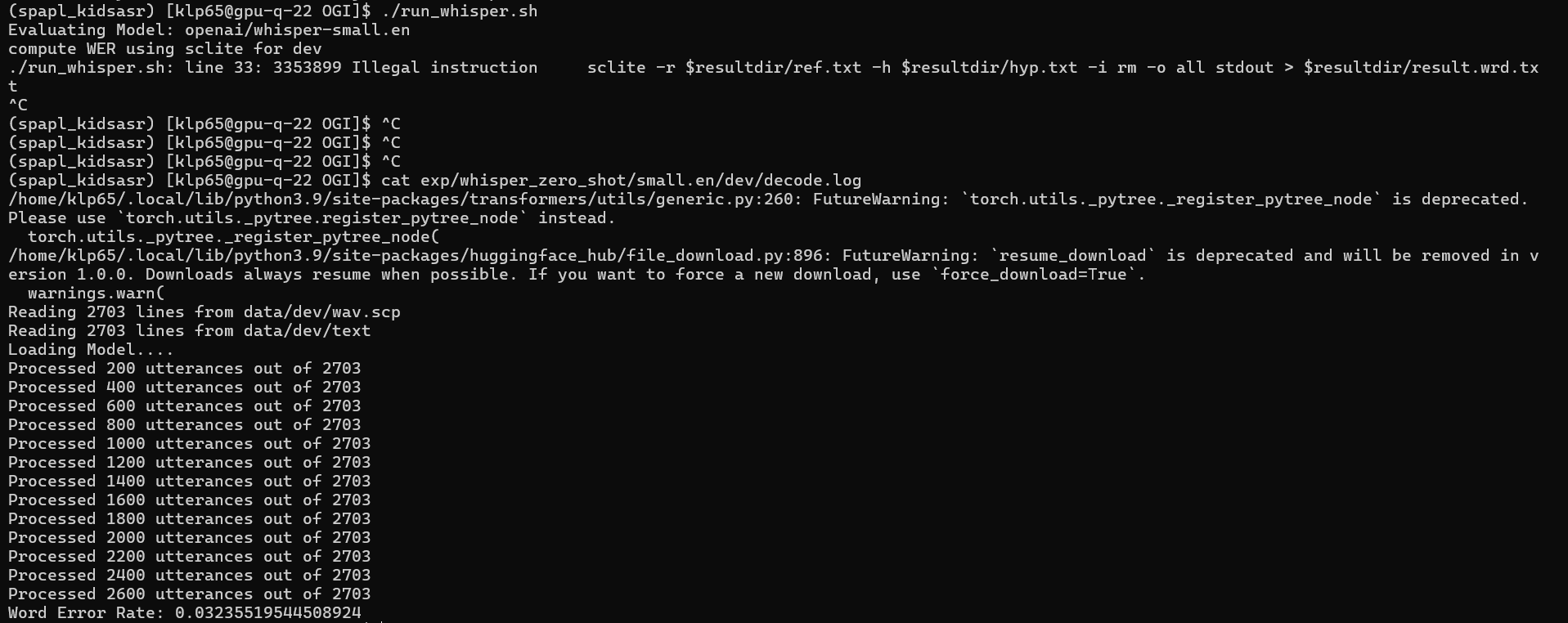
3. SA + VTLP

try PIF with VLTP and PP.

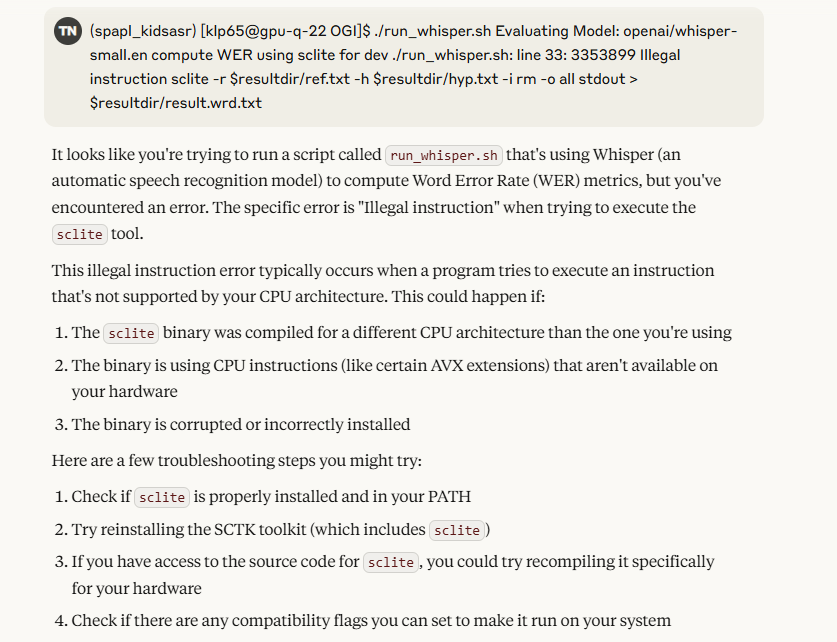
**Testing OGI corpus code**

**Current Issues for Stage 1:**

*cat exp/whisper\_zero\_shot/small.en/dev/decode.log for debugging*

****

Sclite incompatibility but WER is recorded in decode.log file.



**Issues for Stage 2 (No DA, No PEFT):**

*For debugging: ./run\_whisper.sh 2>&1 | tee logfile.txt (this prints output in terminal and saves in logfile.txt)*

To navigate to end of the file: press G

**Issue 1:**

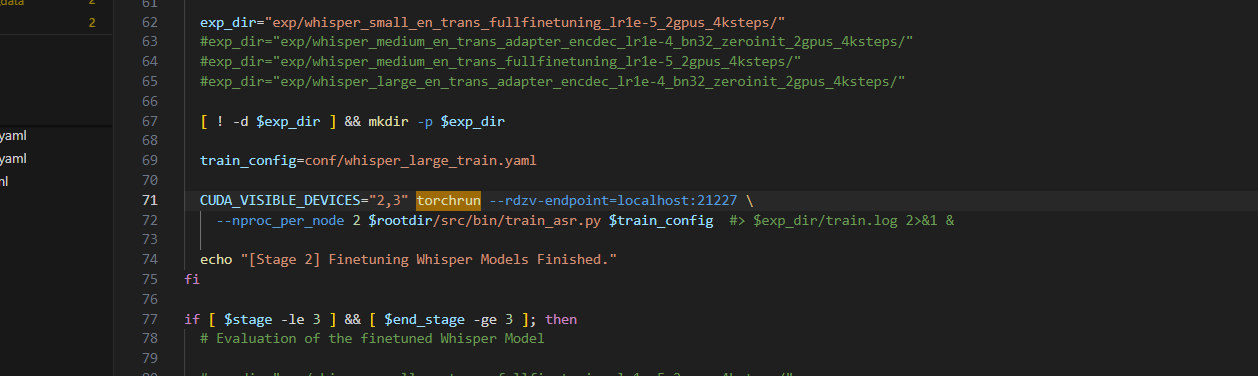
[rank0]: ValueError: FP16 Mixed precision training with AMP or APEX (`--fp16`) and FP16 half precision evaluation (`--fp16\_full\_eval`) can only be used on CUDA or NPU devices.

W0409 23:25:53.419351 3354519 site-packages/torch/distributed/elastic/multiprocessing/api.py:897] Sending process 3354522 closing signal SIGTERM

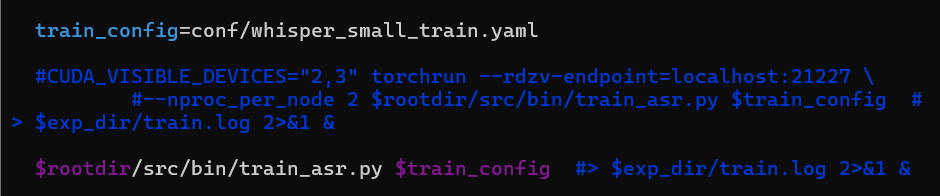
E0409 23:25:53.427443 3354519 site-packages/torch/distributed/elastic/multiprocessing/api.py:869] failed (exitcode: 1) local\_rank: 0 (pid: 3354521) of binary: /home/klp65/.conda/envs/spapl\_kidsasr/bin/python

Solution: set fp16 to False in yaml config file.

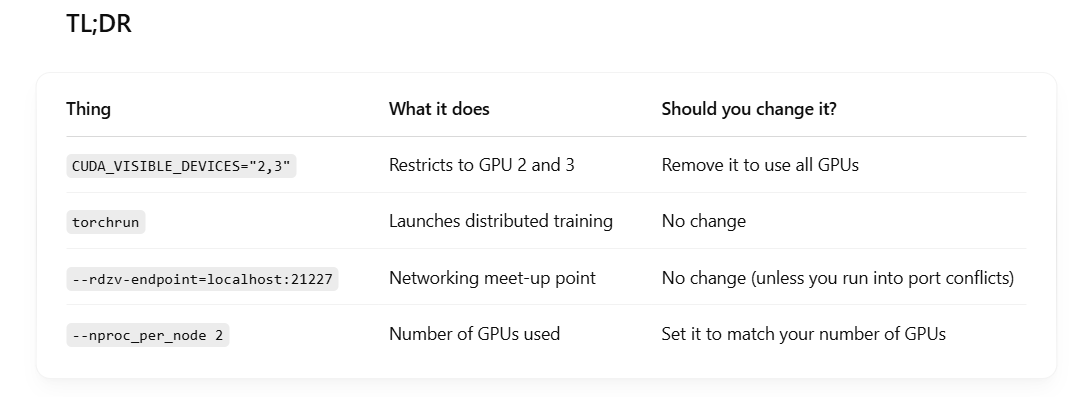
**Issue 2:**

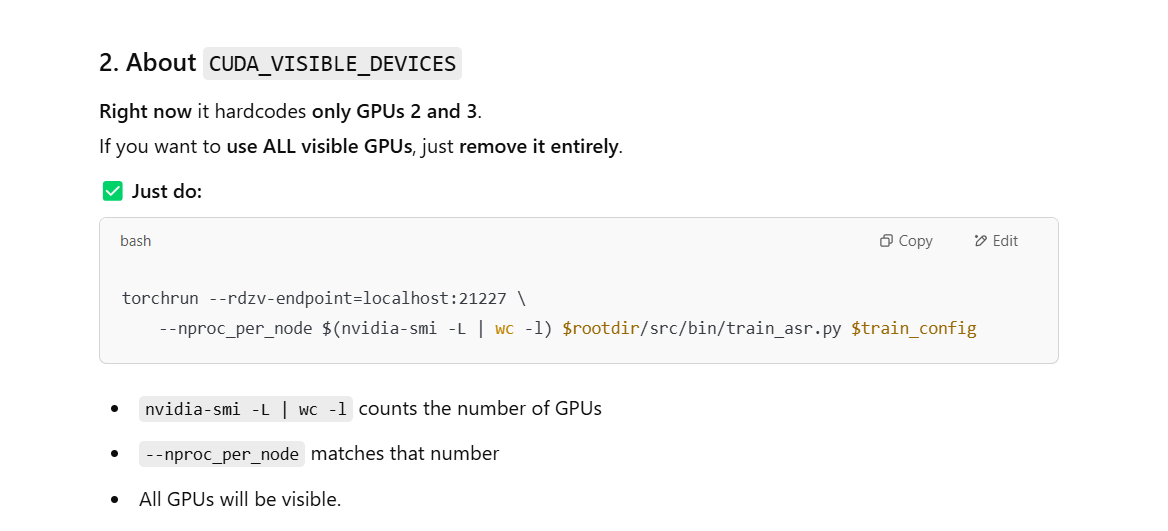
****

Remove line 71 in run\_whisper.sh since only 1 GPU, run .py directly



Trying to Adapt Code for Distributed Training (Multiple GPUs)

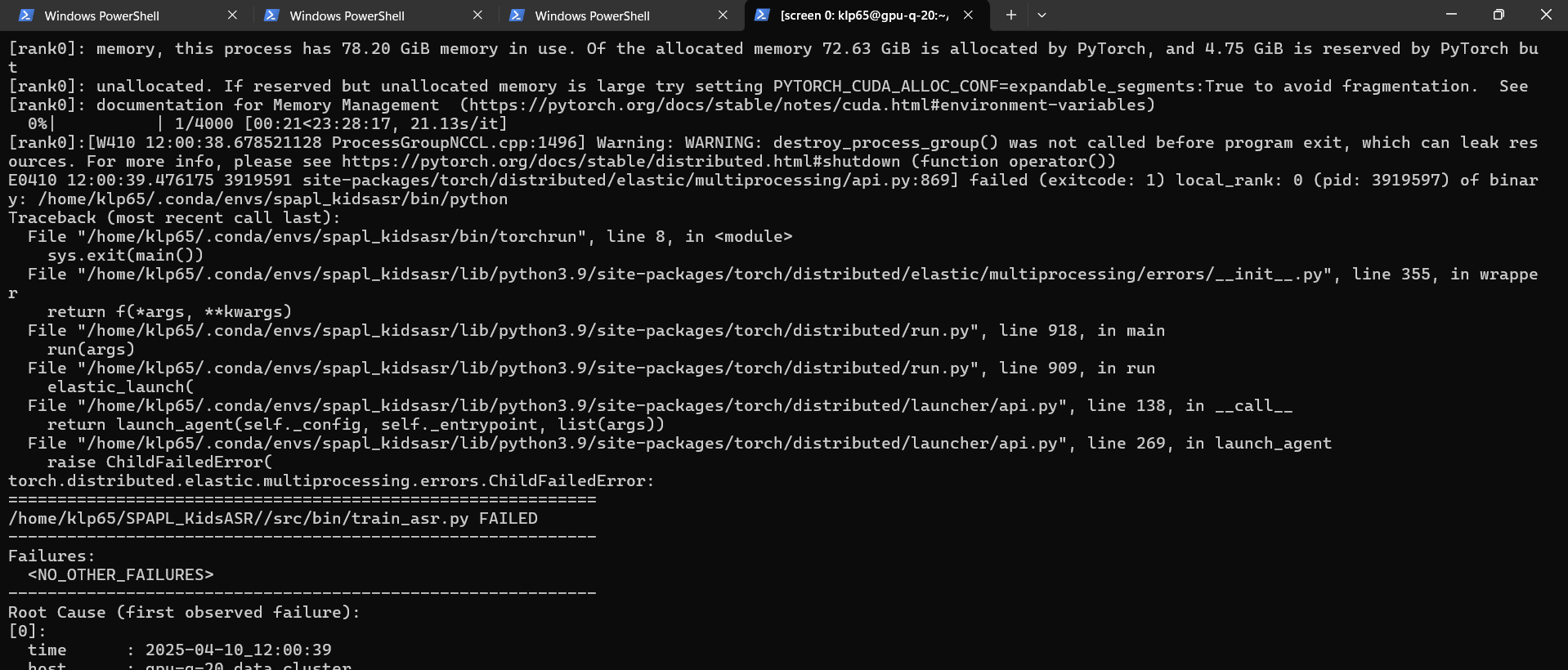




torchrun --rdzv-endpoint=localhost:21227 \

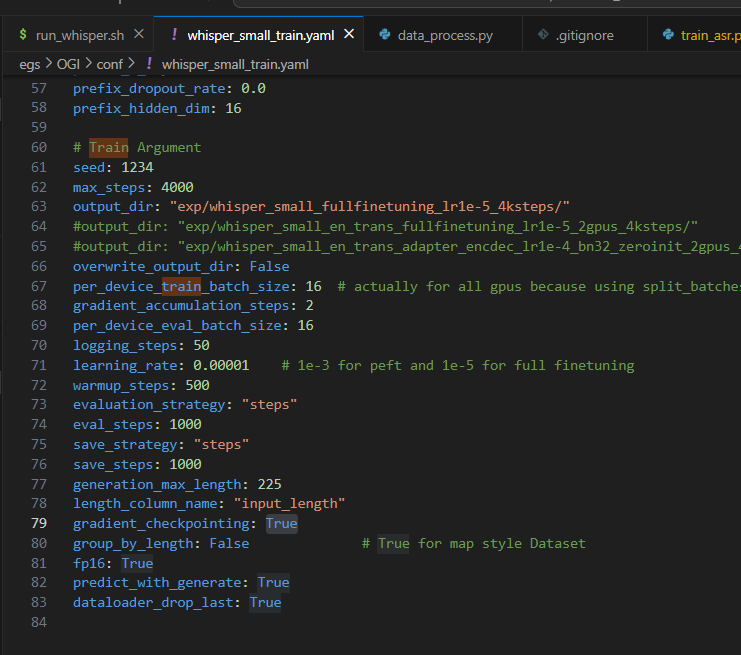
--nproc\_per\_node $(nvidia-smi -L | wc -l) $rootdir/src/bin/train\_asr.py $train\_config

But this causes out-of-memory error:

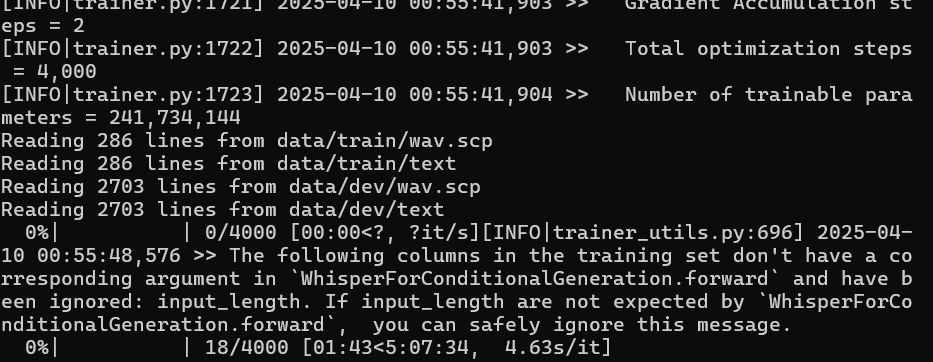


**Issue 3:**

AttributeError: 'WhisperForConditionalGeneration' object has no attribute '\_set\_gradient\_checkpointing'



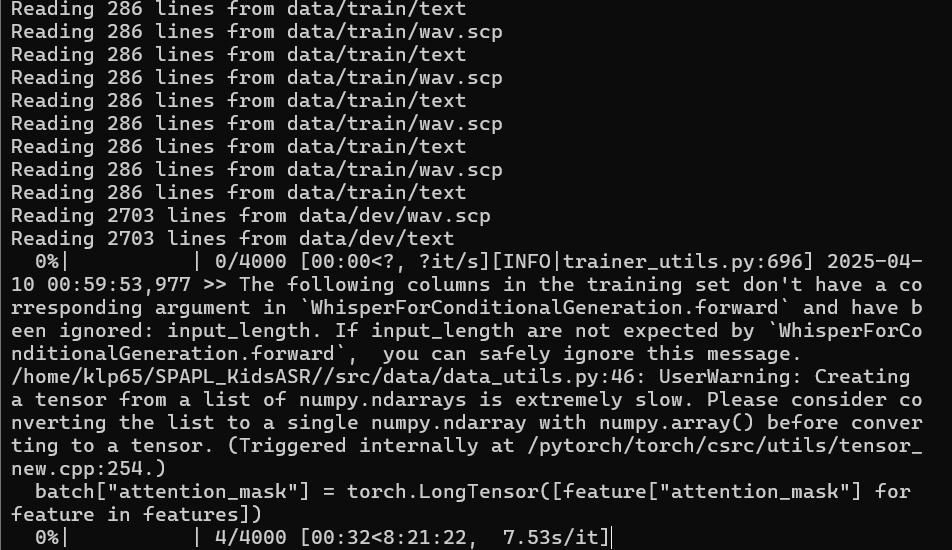
Gradient checkpointing changed to False



Training Success!

**Testing Stage 2 (All DA – not PIF, No PEFT):**

Works!

****

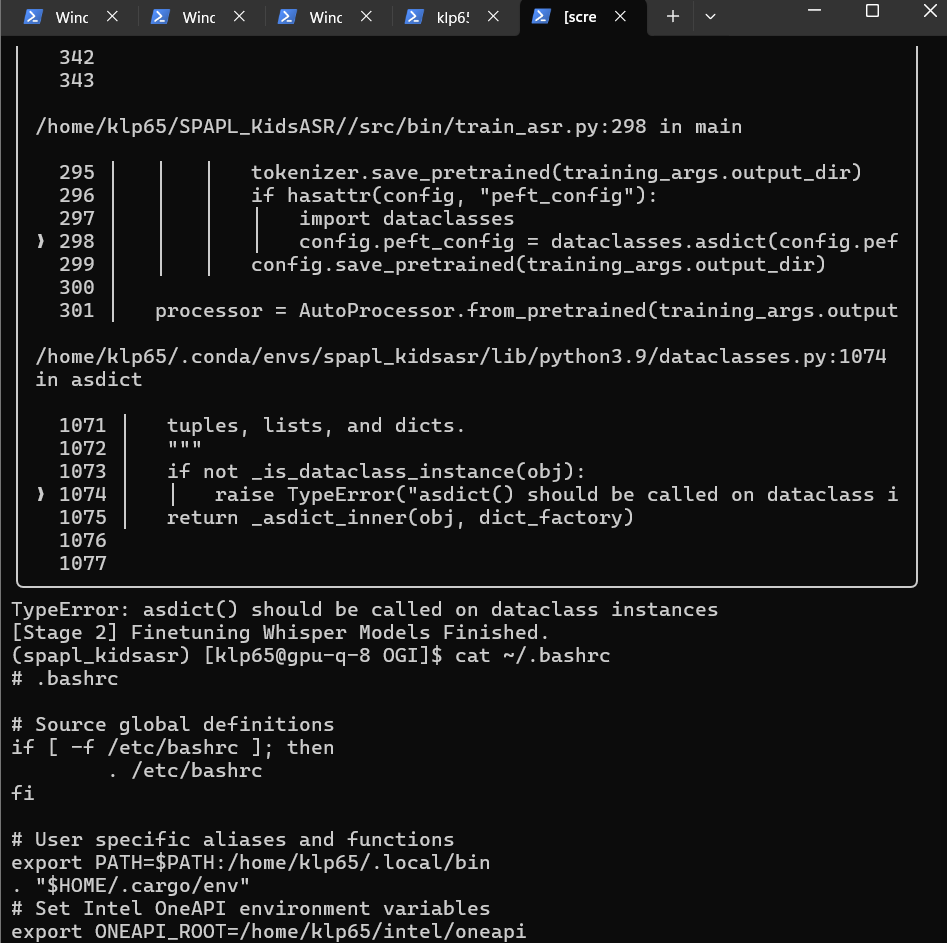
**Testing Stage 2 (All DA – PIF + PP, No PEFT):**

Error. May not want to explore this.

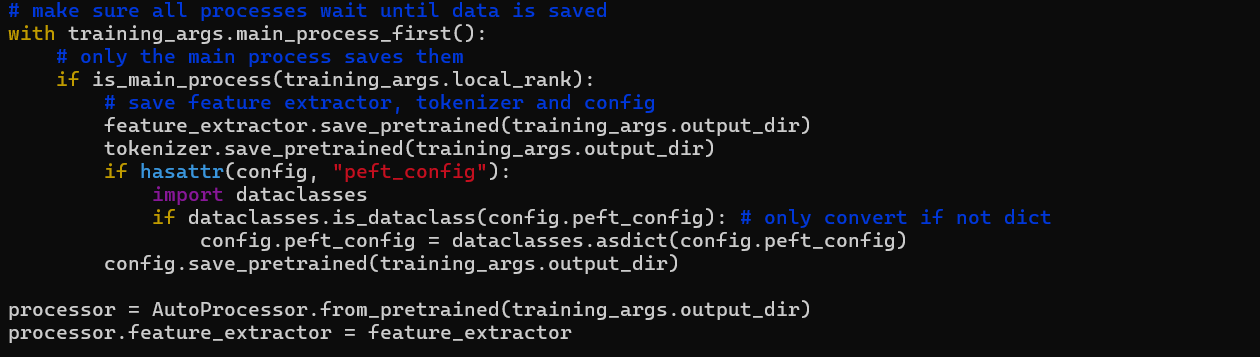
****

**Testing Stage 2 (No DA, lora adaptor)**

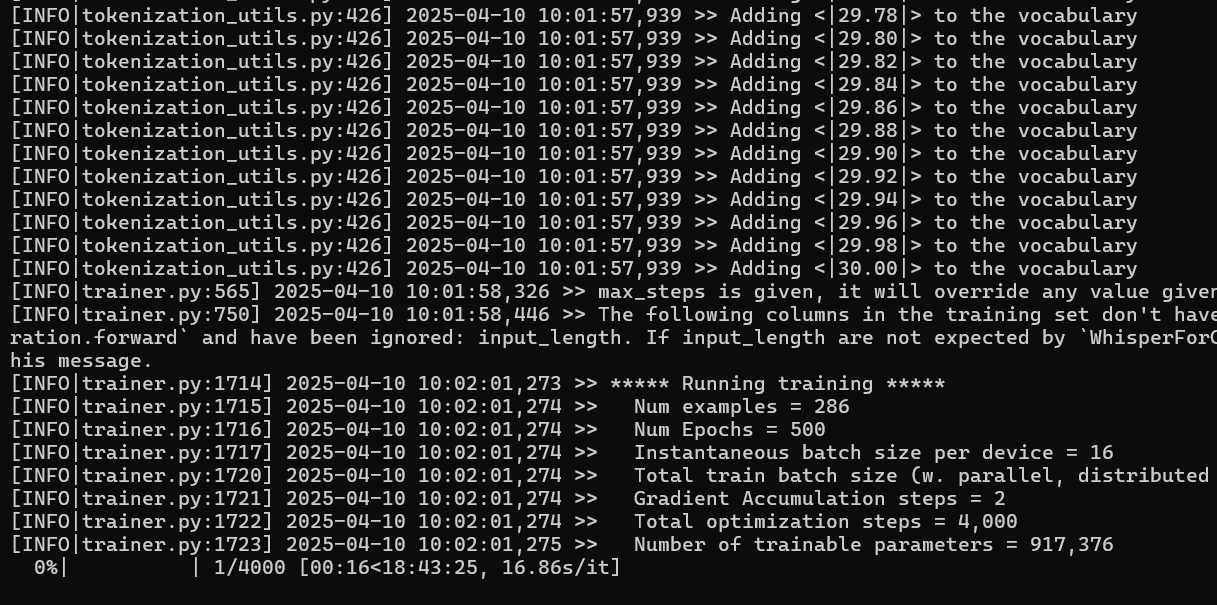
Error:



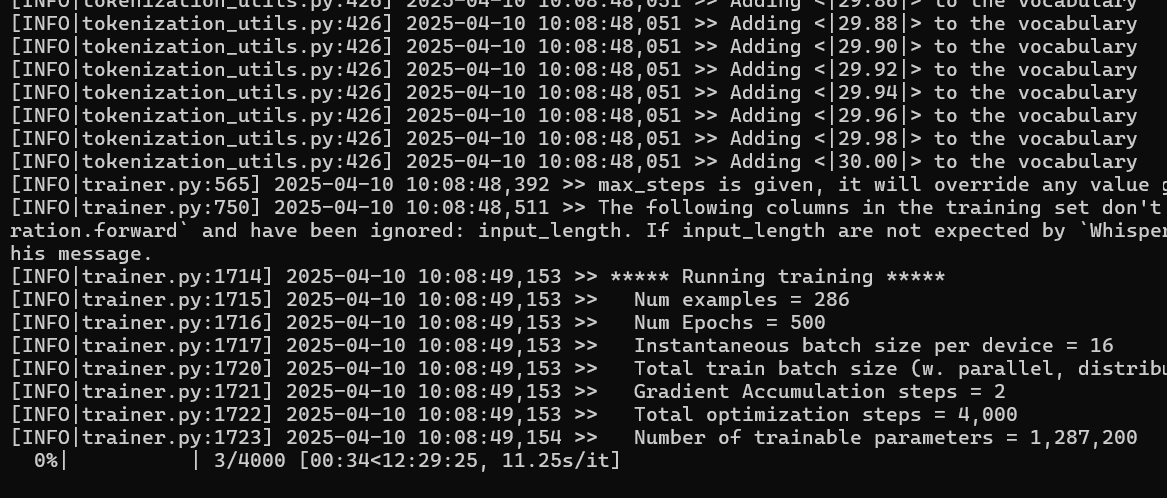
Resolve by:

****

Works!

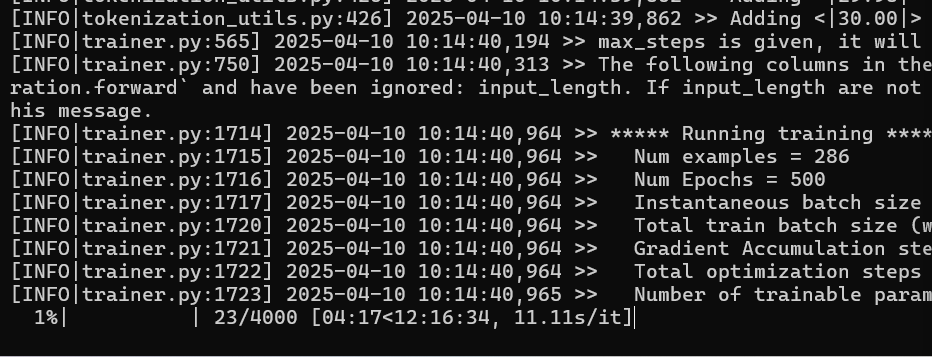
****

**Testing Stage 3 (No DA, adaptor)**

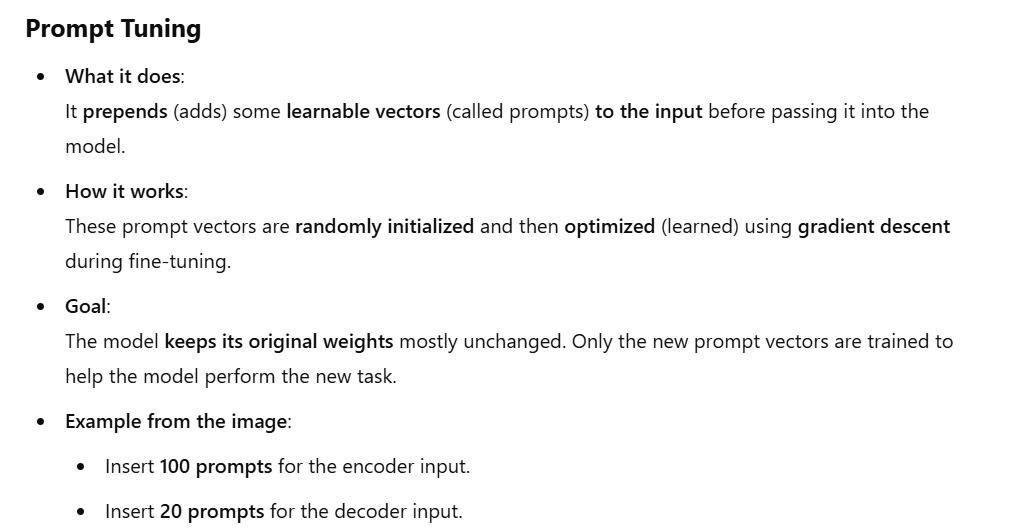
****

Works!

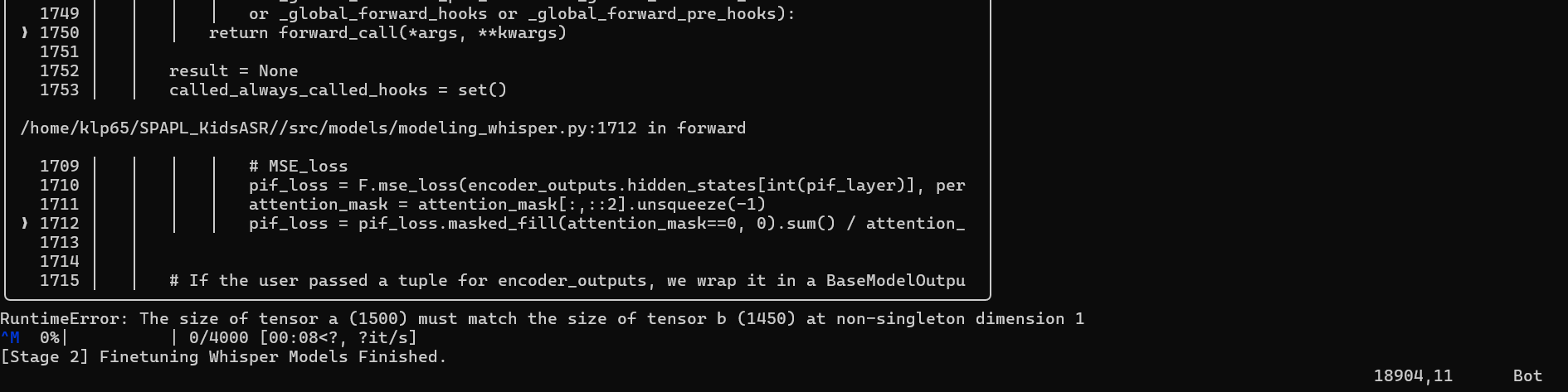
**Testing Stage 3 (No DA, prompt tuning)**

****

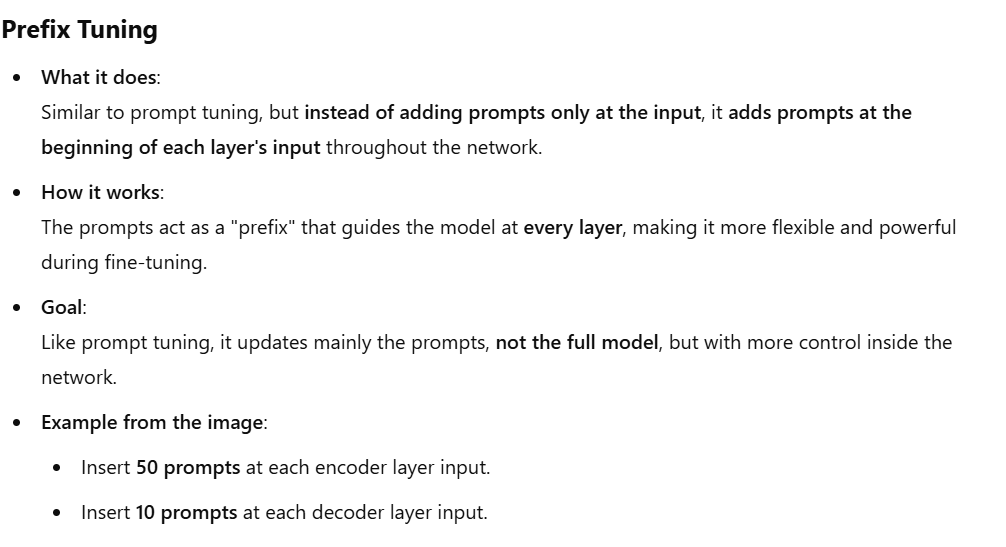
Works!



**Testing Stage 3 (No DA, prefix tuning)**

****

Error. (Read up on prefix tuning)



**Unsolved Error**

**Stage 3 Untested**

(same code as Stage 1 except that takes model from checkpoint)

**Testing MyST corpus code**

Train and development data:

    name: 'train'

    scp\_path: data/train\_filter/wav.scp

    text\_label: data/train\_filter/text

    name: 'development'

    scp\_path: data/select\_valid/wav.scp

text\_label: data/select\_valid/text

Test data:

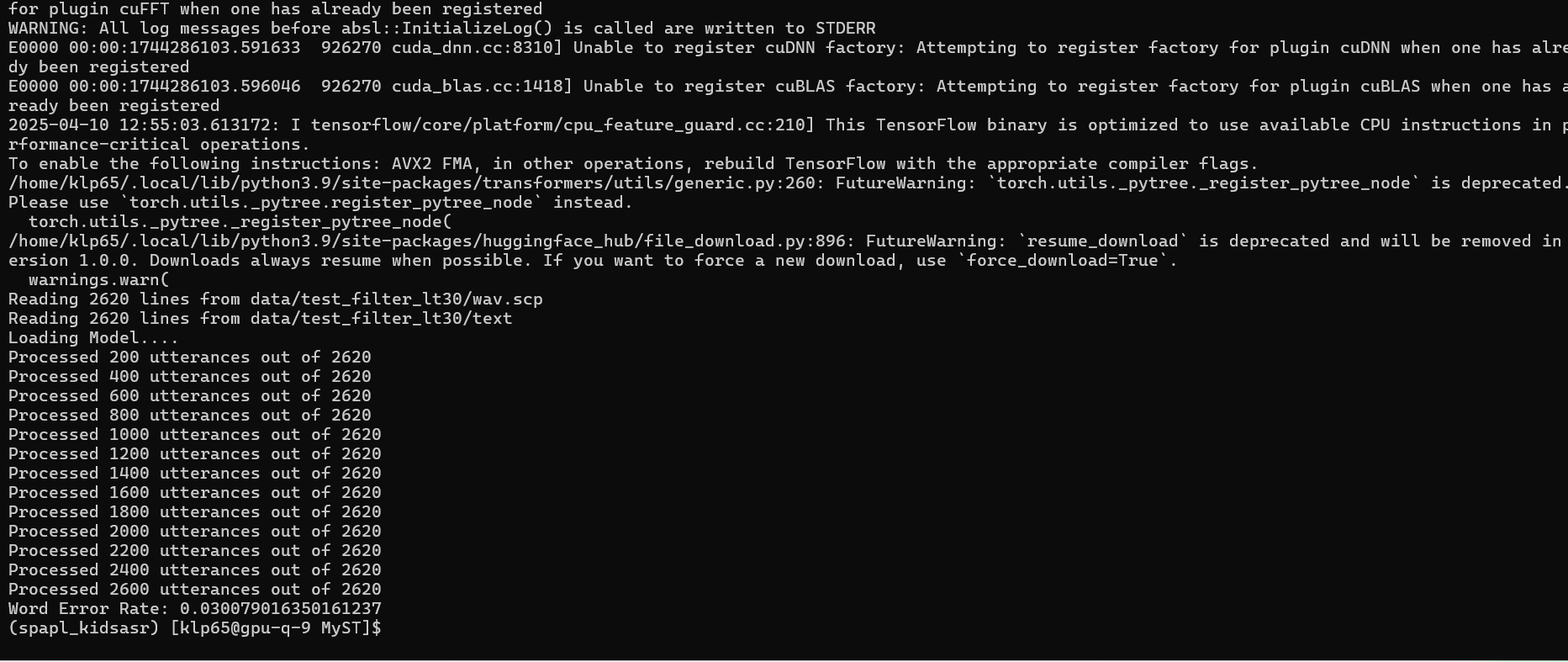
data/test\_filter\_lt30

data/test\_filter\_gt30

data/development\_filter

**Testing Stage 1**

*cat exp/whisper\_zero\_shot/small.en/test\_filter\_lt30/decode.log for WER (sclite not working)*



**Testing Stage 2: Full FT, No DA**

Works!

**Testing Stage 2: Full FT, All DA**

Works!

**Testing Stage 2: LoRA, No DA**

Works!

**Testing Stage 2: Adaptor, No DA**

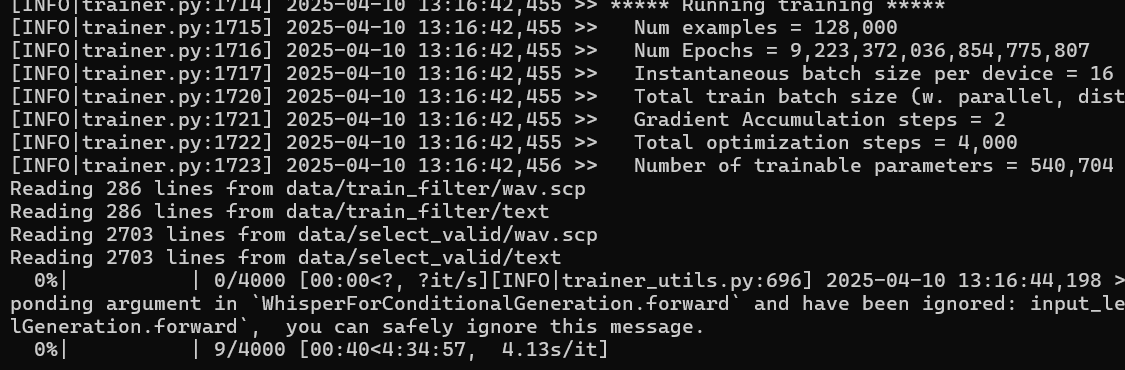
Works!

**Testing Stage 2: Prompt tuning, No DA**

Works!

**Testing Stage 2: Prefix tuning, No DA**

**Works (unlike above)**

****

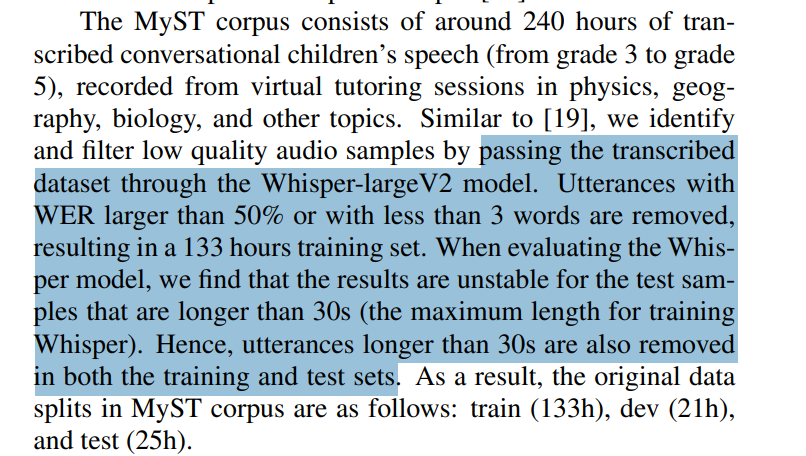
**Testing Stage 2: Freeze Encoder, No DA**

Works!

**Testing Stage 2: Freeze Decoder, No DA**

Works!

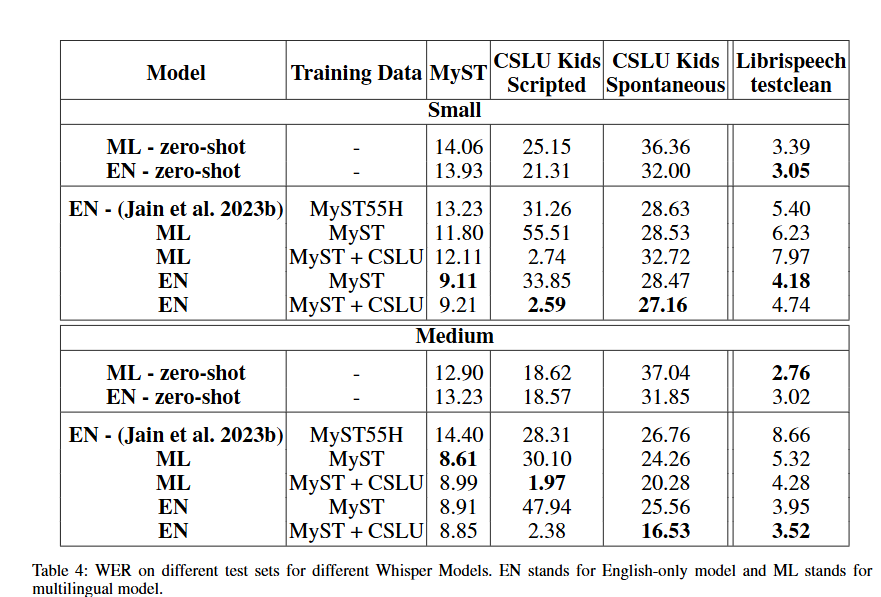
**Data Preprocessing:**



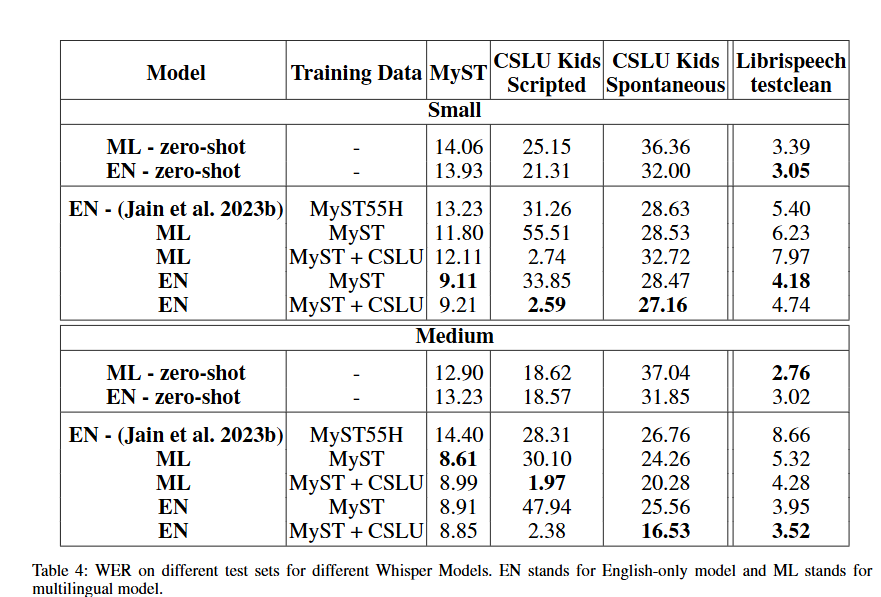
<https://ojs.aaai.org/index.php/AIES/article/view/31618/33785>

paper for data preprocessing.

**To Replicate:**

****

After finetuning:

****

**Read this paper:** <https://ojs.aaai.org/index.php/AIES/article/view/31618/33785>

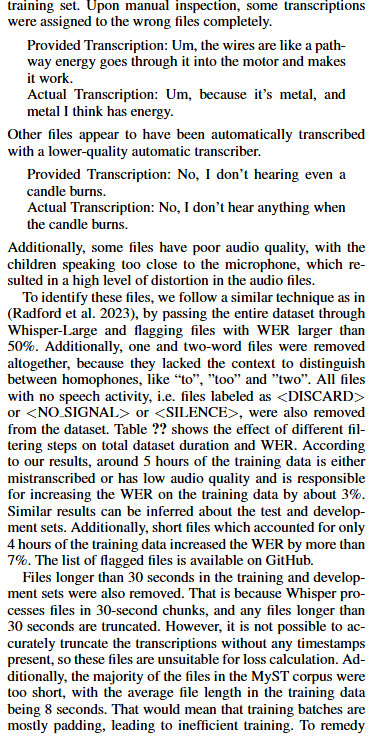
and think about what to do for data preprocessing and combination.

MyST is the largest publicly available children’s speech corpus. A recent study (Jain et al. 2023b) has attempted to adapt Whisper to the MyST corpus. They found that the quality of audio files as well as transcriptions in the MyST corpus varies, and were able to extract 65 hours of well-transcribed speech from the 197 hours of transcribed speech provided in MyST. We expand upon their work by outlining a more efficient data preprocessing scheme and extracting a total of 179.2 hours, which we show improves the performance of Whisper.

Train-test-split for MyST should be maintained to ensure no overlap in speakers.

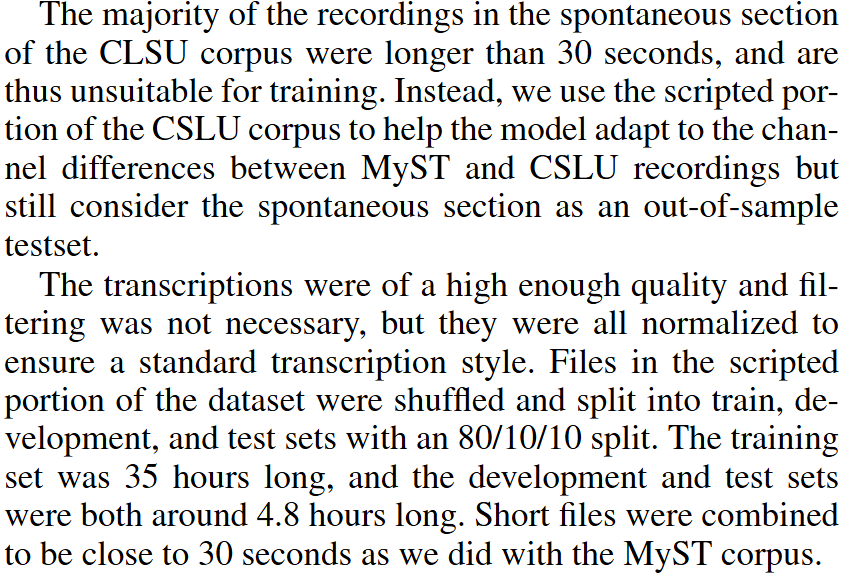
Improving the WER on the spontaneous part of the CSLU Kids dataset (Shobaki, Hosom, and Cole 2000) from 32.00% to 27.16% w/o explicitly including the dataset.

MyST Preprocessing:

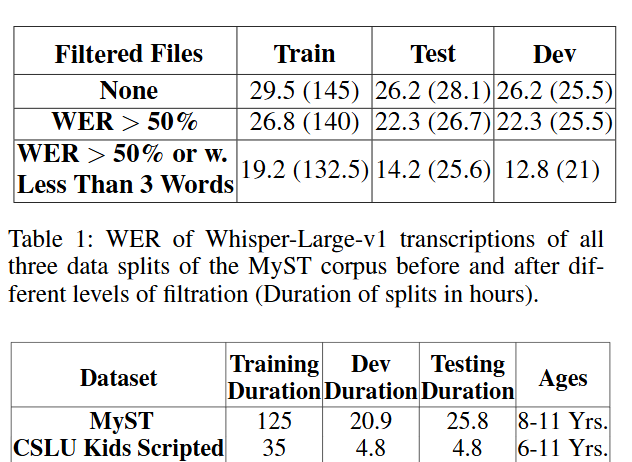


Consequently, all the text was mapped to be lowercase and further normalized using theWhisperNormalizer1 Python package, which mapped tokens like ”you’re” to a standard ”you are”, as well as mapping all digit numbers to be spelled out.

CSLU preprocessing:



To replicate:



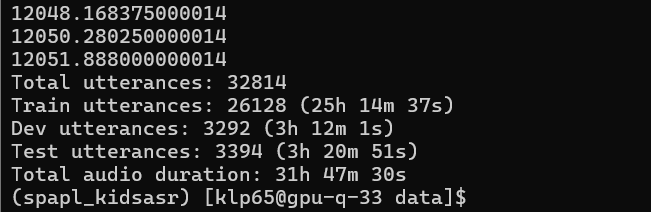
To show the impact of WER, can calculate combined duration of .wav files after training.

**Exploring OGI corpora on Speech System**

Corpus is composed of both prompted and spontaneous speech from 1100 children from kindergarten (age 5) through grade 10 (age 15 to 16).

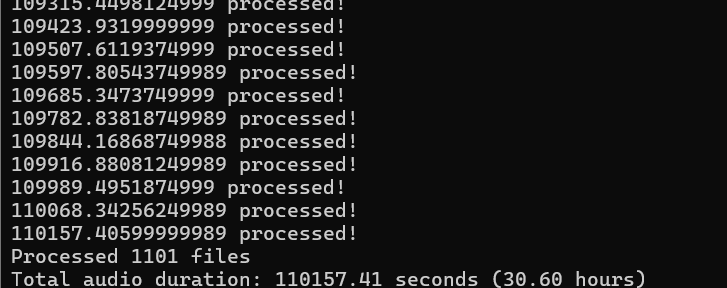
Scripted:

The subject then repeated the prompt, which was recorded via a head-mounted microphone and digitized at 16 bits and 16 kHz using a SoundBlaster 16 PnP sound card. The recorded utterance was then played back to the subject and the data-collection supervisor. If the recording was deemed unacceptable, the prompt was repeated.

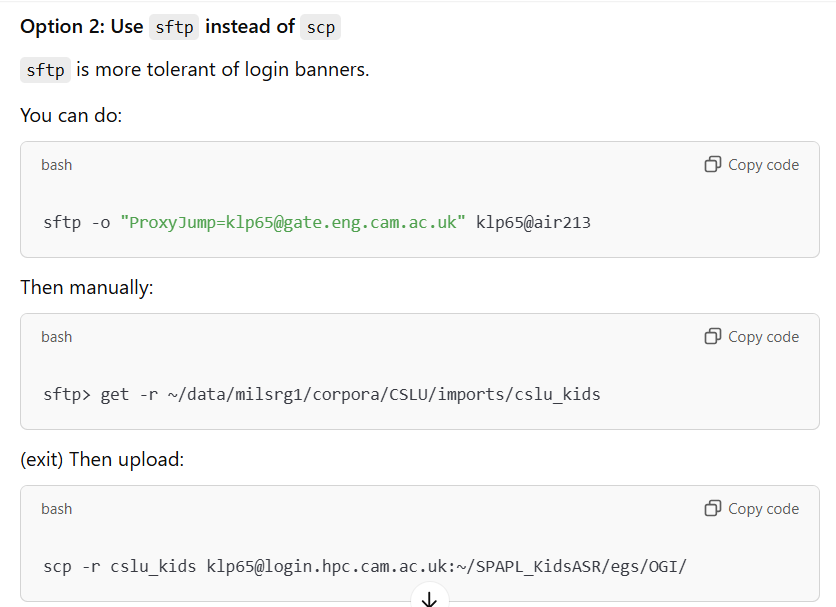


Spontaneous:

After the prompted speech phase of the collection was completed, the experimenter asked the subject a series of questions intended to elicit spontaneous speech (i.e. “Tell me about your favorite movie”). The total amount of speech recorded per subject was approximately 8-10 minutes. Some additional biographical information was collected about each subject, including age, gender, languages spoken, and any physical maladies that could affect their speech.



Copying data over:

****

sftp -o "ProxyJump=klp65@gate.eng.cam.ac.uk" klp65@air213

sftp> get -r /data/milsrg1/corpora/CSLU/imports/cslu\_kids

scp -r cslu\_kids [klp65@login.hpc.cam.ac.uk:~/SPAPL\_KidsASR/egs/OGI/](mailto:klp65@login.hpc.cam.ac.uk:~/SPAPL_KidsASR/egs/OGI/)

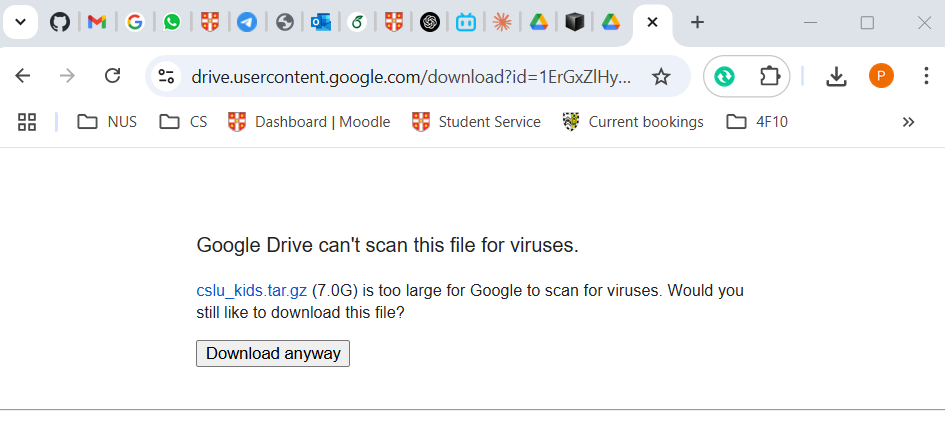
(keeps getting same error)

Final solution:

Upload to google drive and get link:



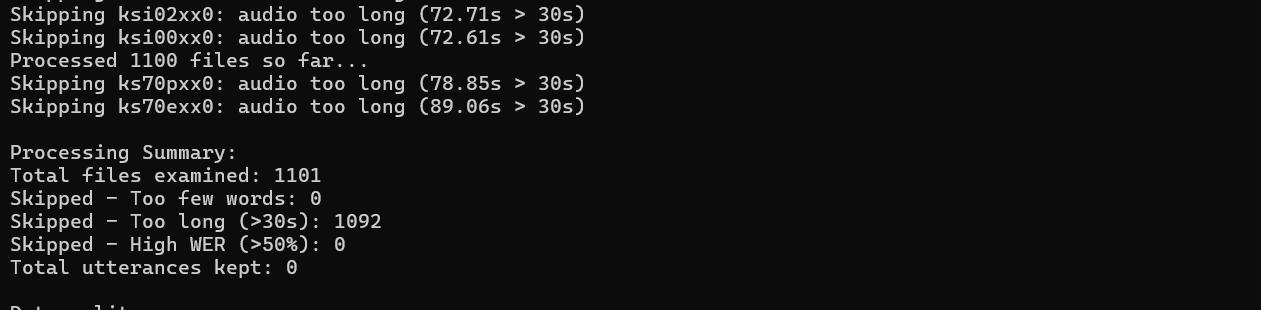
Download:



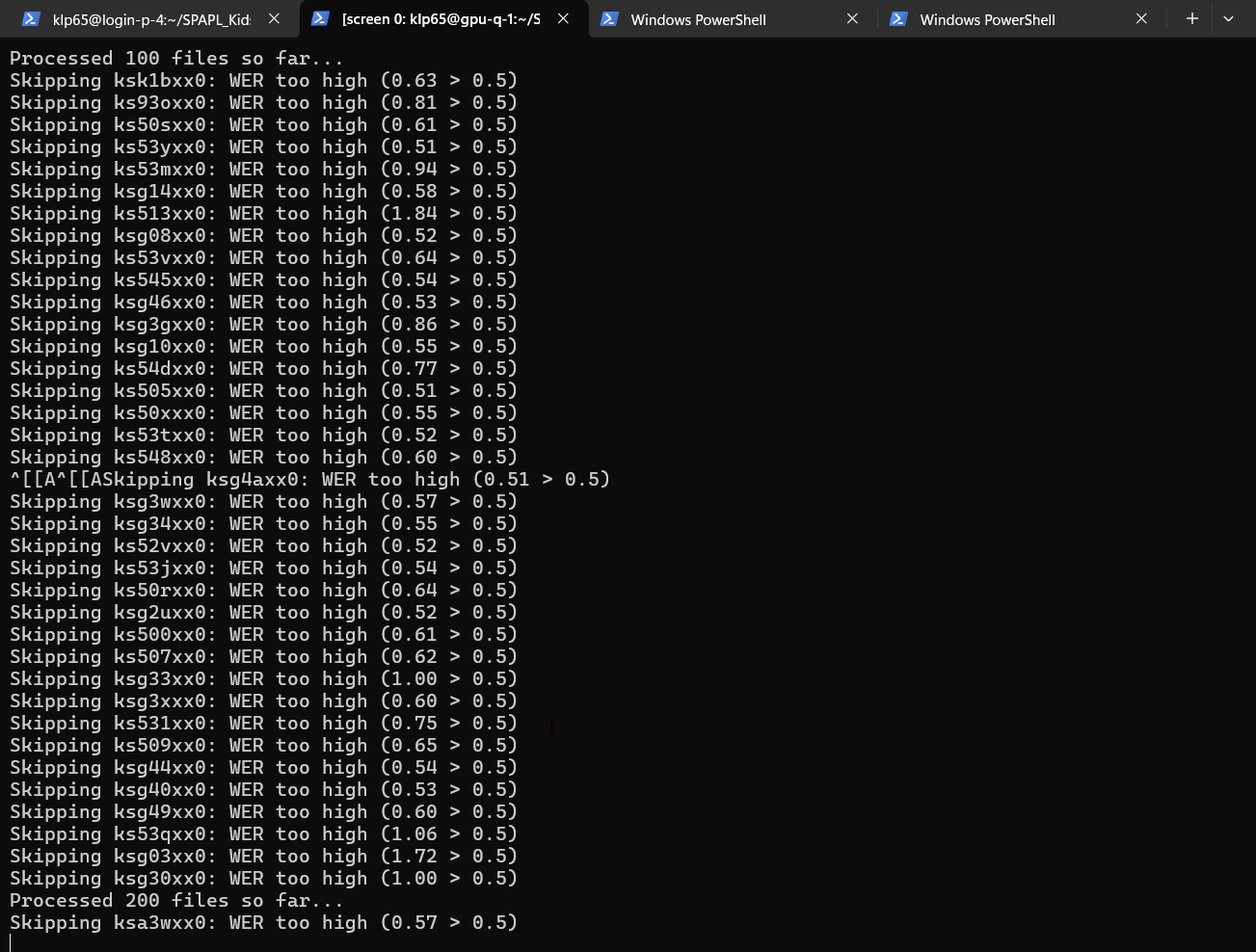
wget --load-cookies /tmp/cookies.txt "https://drive.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cookies.txt --keep-session-cookies --no-check-certificate 'https://drive.google.com/uc?export=download&id=1ErGxZlHyk9VsSwHNS2N4B3j-nhFHyZWx' -O- | sed -rn 's/.\*confirm=([0-9A-Za-z\_]+).\*/\1\n/p')&id=1ErGxZlHyk9VsSwHNS2N4B3j-nhFHyZWx" -O cslu\_kids.tar.gz && rm -rf /tmp/cookies.txt

**For OGI spontaneous:**

For OGI spontaneous (with data preprocessing):



Without the 30s limitation:



37% is deemed poor quality (WER > 0.5)

For the large ratio of poor quality data (w.r.t total hours in dataset) and all utterances > 30s, we might not want to use this dataset in training/testing.

For OGI spont dev (before training, without any data preprocessing):

Word Error Rate: 0.3737906110002635

**For OGI scripted:**

For OGI scripted (with data preprocessing):

**save data in /home/klp65/rds/hpc-work**

testing spontaneous speech and scripted speech separately first, see if can combine them?

For the paper, it only uses the scripted speech.

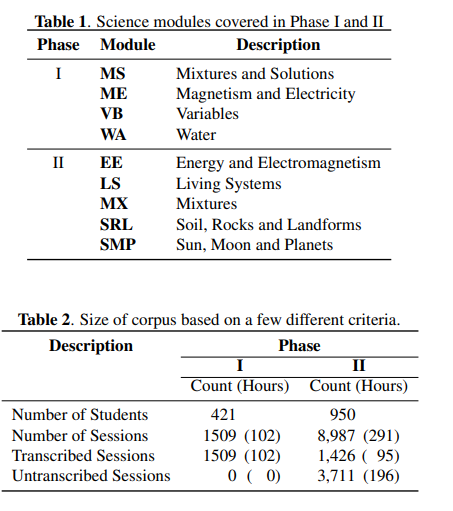
**Exploring MyST corpora on Speech System**

MyST children’s conversational speech corpus consists of spoken dialog between 3rd, 4th and 5th grade students (8-11 years old).

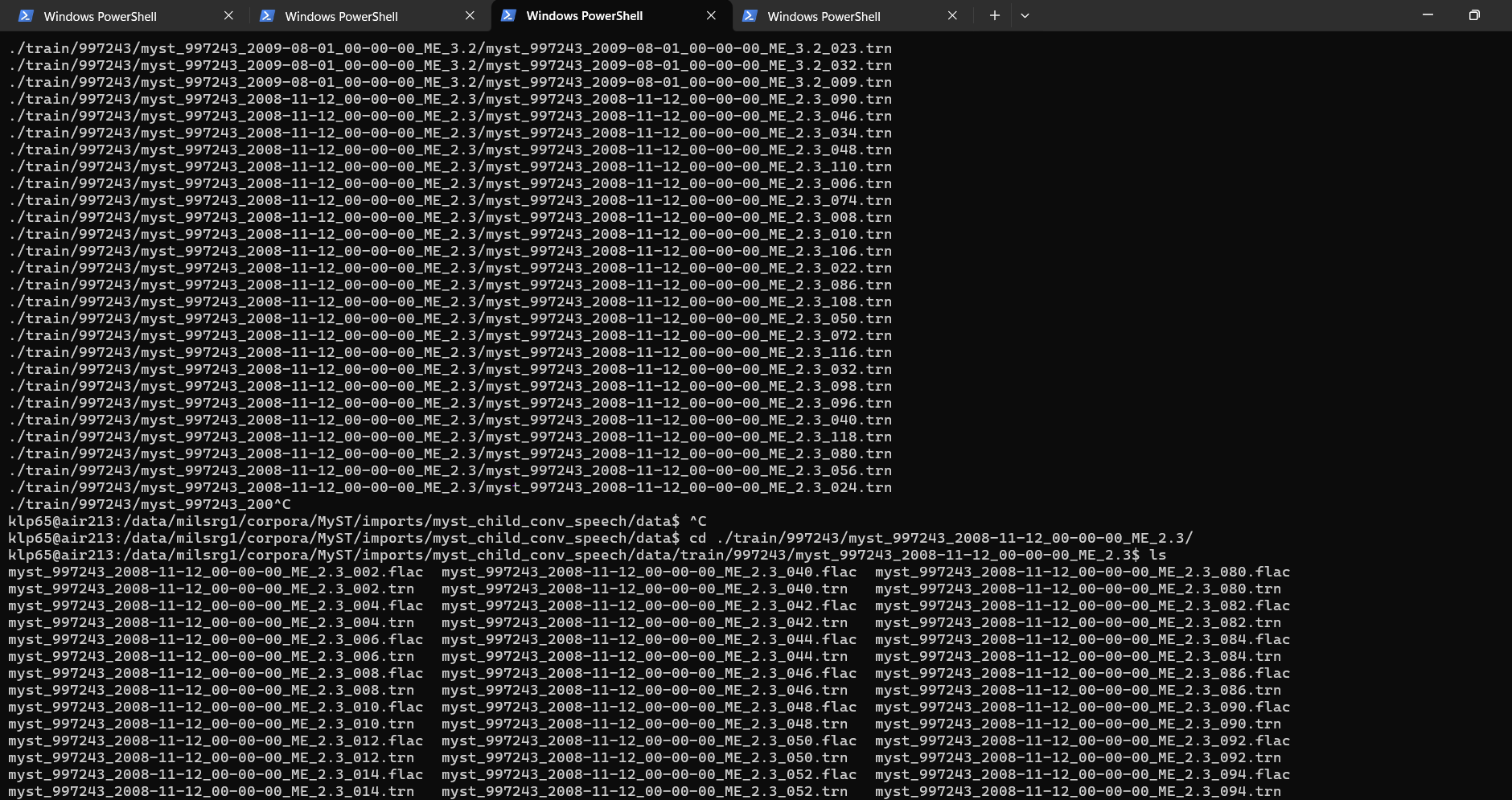
As part of the study, students engaged in spoken dialog with a virtual science tutor — a lifelike computer character that produced accurate lip and tongue movement synchronized with speech produced by a voice talent. Analyses of the spoken dialog sessions indicated that, during a dialog of about 15 minutes, tutors and students produced about the same amount of speech, around 5 minutes each. This approach was used to develop over 100 tutorial dialog sessions, of about 15 minutes each.

Phase II transcription is cheaper and simpler transcription, may not be as good.

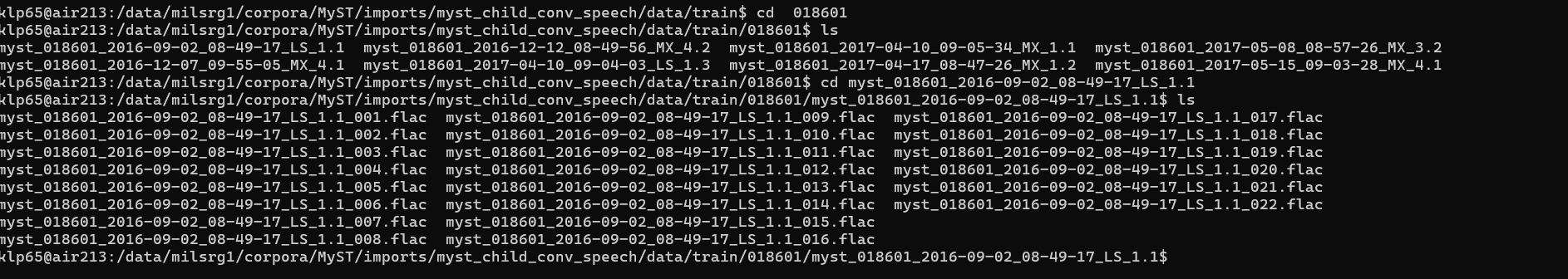
Dropbox download link: <https://www.dropbox.com/scl/fi/v3tybtxbljn3ke9xnizge/myst_child_conv_speech.tar.gz?rlkey=yx6hyet1lfc6mwzwt13ubib5h&st=2c5svqvz&dl=1>



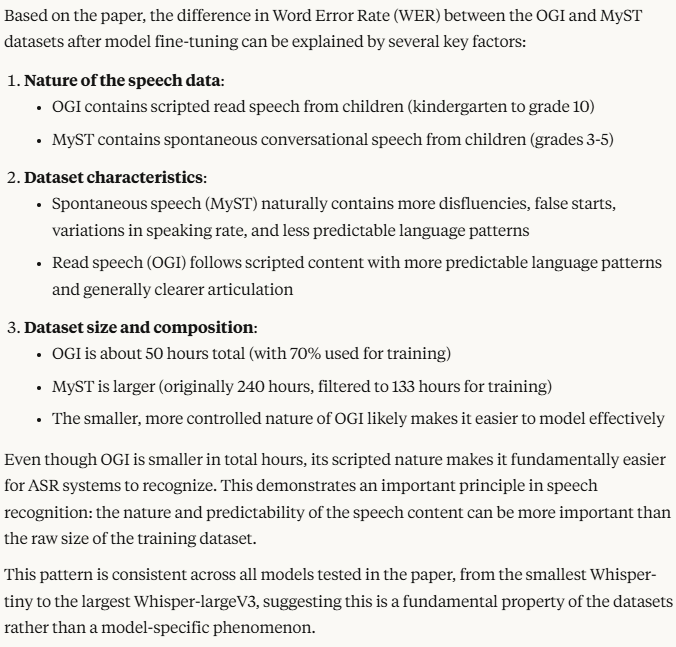
Seems to be transcripts in some:



No transcripts in other:



Are these the untranscribed sessions above?

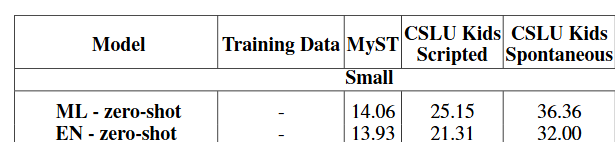


Might not want to mix.

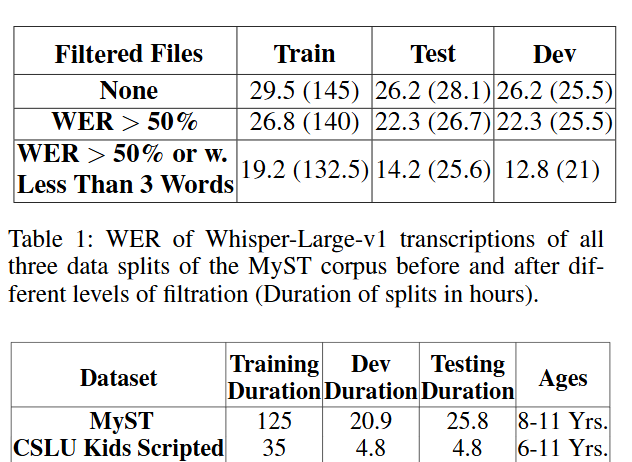
Test PEFT + DA for whatever configuration (mixed/not mixed) that gives better results.

To-dos:

1. Get MyST data filtering settled.
   1. **Did not handle numbers to word conversion in whispernormalizer. (can include that in decode asr.py instead)**
2. Calculate the train test dev duration of the MyST filtered files
3. Get zero-shot WER for whisper small on CSLU Kids Scripted test set and spontaneous all



1. Get zero-shot WER for whisper small on MyST
2. Do zero-shot WER for MyST different filtering techniques + calculate train test dev duration (not very important)



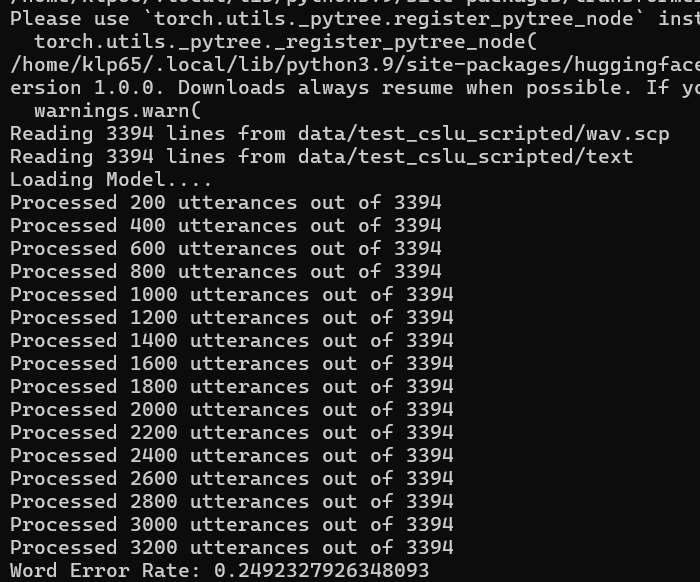
1. Apply full finetuning of the model with MyST
2. Apply full finetuning of the model with MyST + CSLU Scripted



1. Apply PEFT and data augmentation with MyST only
   1. Apply PEFT first (may not have enough for full finetuning)
   2. After finding best PEFT, apply DA and add CSLU Kids Scripted

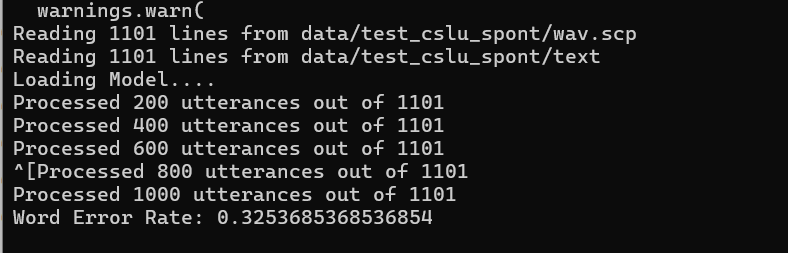
|  |  |  |  |
| --- | --- | --- | --- |
| Training Data | MyST | CSLU Kids Scripted | CSLU Kids Spontaneous |
| - | 12.8588994774% | 24.923279263% | 32.5368% |
| MyST |  |  |  |
| MyST + CSLU Scripted |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Zero Shot Testing**

CSLU scripted 

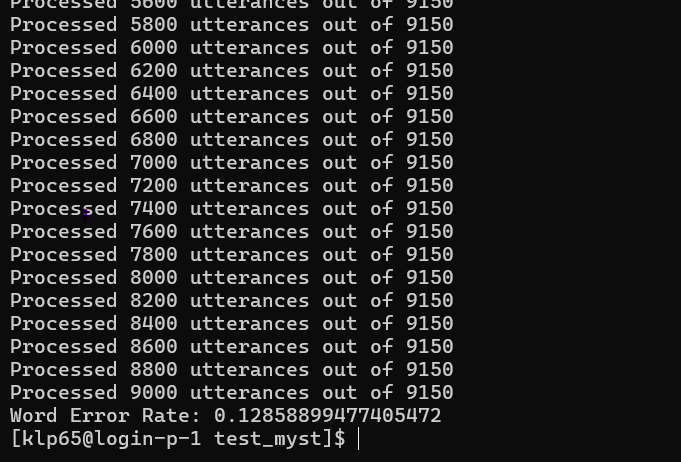
**WER: 24.9%**

CSLU Spontaneous

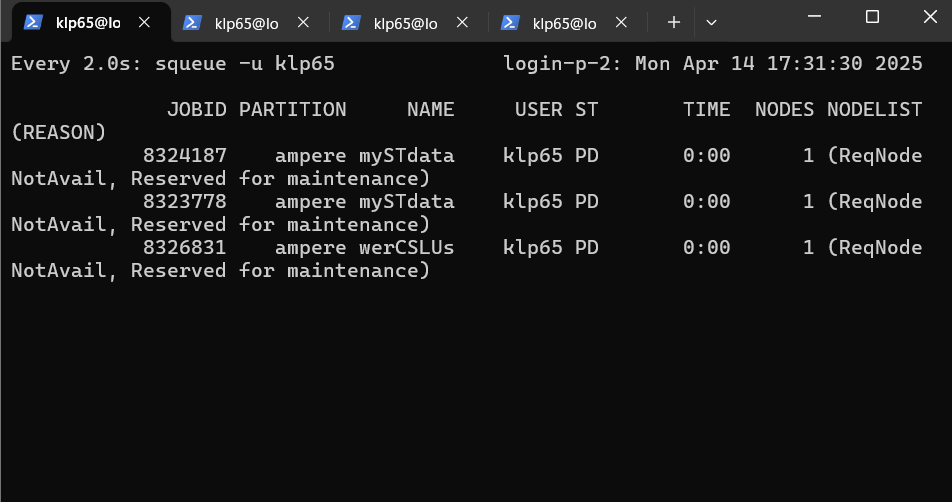


32.5%

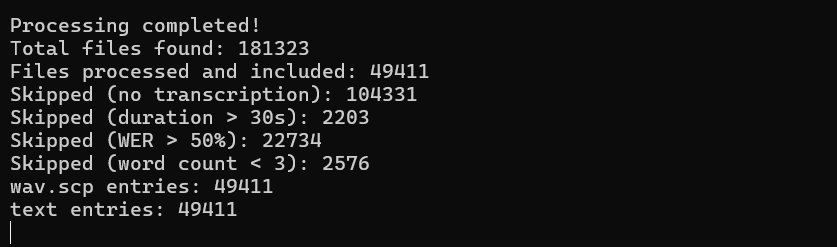
MyST



12.8%



myST train done



**Reducing Training Time:**

Current prompt\_tuning estimated: 3 hours

Number of trainable parameters = 92,160

Fp16 works!

Attempting to combine wav files to 30s for train and dev sets for myST and cslu to quicken training even further.

Already processed and transferred combined wav files into new files for myST.

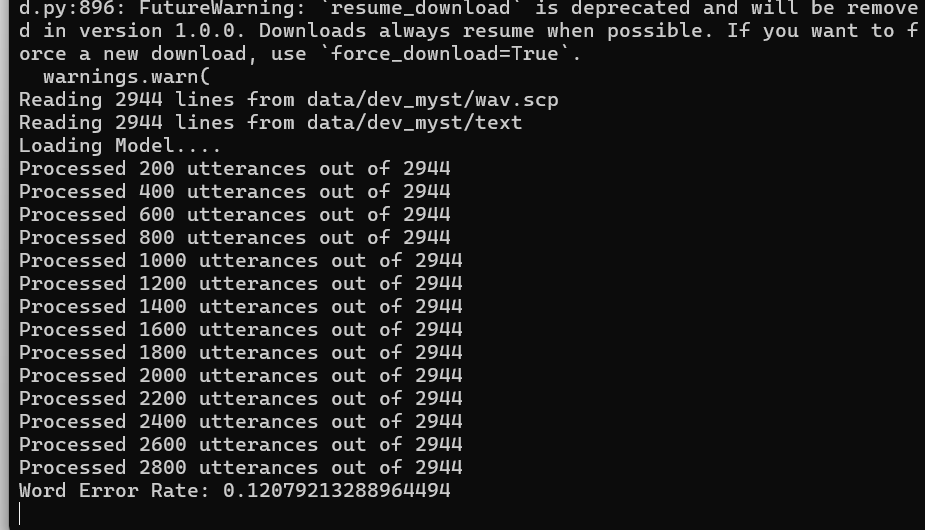
Combine 30s for cslu too.

Calculate the total duration of training, dev and test set after filtering

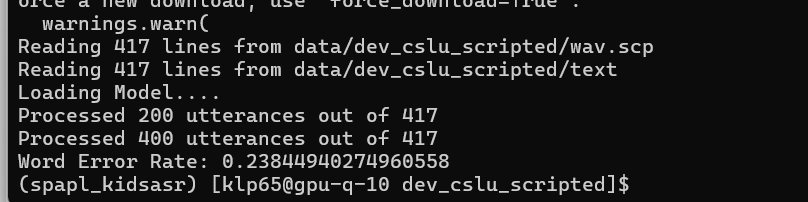
1. Refilter CSLU Scripted – if too little, consider not using WER too, just filter 30s
   1. Calculate duration and utterances of training, dev and test set for CSLU after filtering (cpu)
   2. redo num2words (cpu)
   3. redo 30s combine for dev and training (cpu)
   4. trf files into myST/data
   5. run decode on the FT model and zero-shot to check WER (gpu)
2. Test WER of MyST dev after 30s combine (gpu)

After that, send model finetuning.

Can use CPU hours to setup environment for LLM while waiting for GPU hours from Dudley

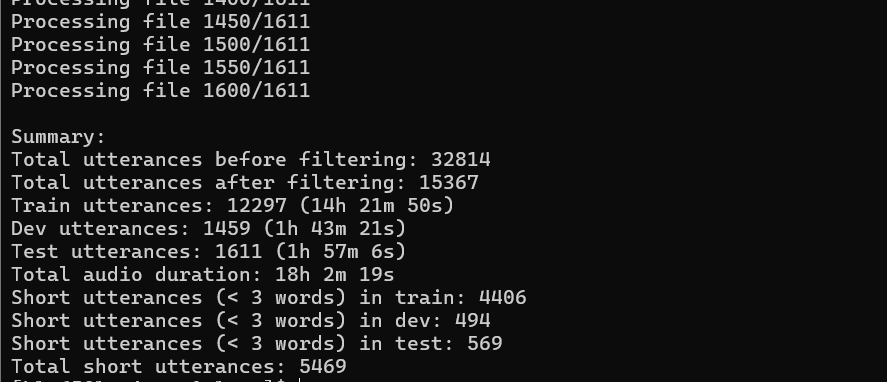
****

After combining 30s, seems that WER is maintained.

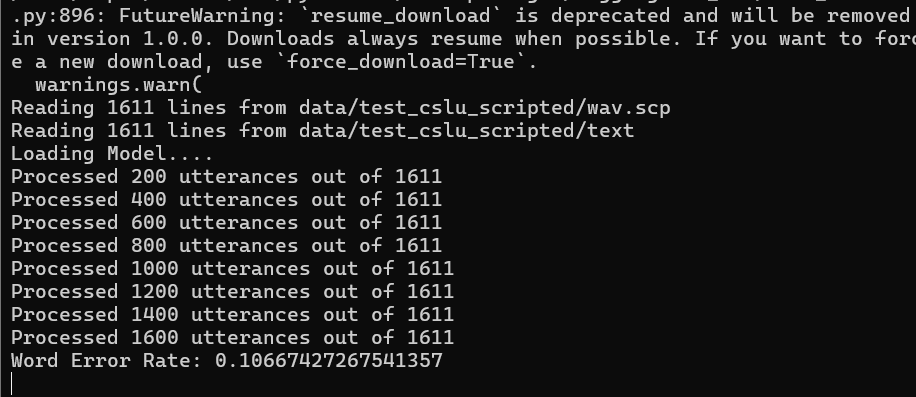


Same for CSLU scripted.

CSLU filtered:



CSLU test (zero shot):



After Prompt tuning:

On MyST: 11.76%

CSLU Spontaneous: 24.67%

Deleted model accidentally ☹ training is here: cat slurm-8410320.out **PLS REMEMBER TO TAKE SNAPSHOT**

**To monitor loss:**

Navigate to runs/run\_name/

/home/klp65/.conda/envs/spapl\_kidsasr/bin/tensorboard --logdir=./

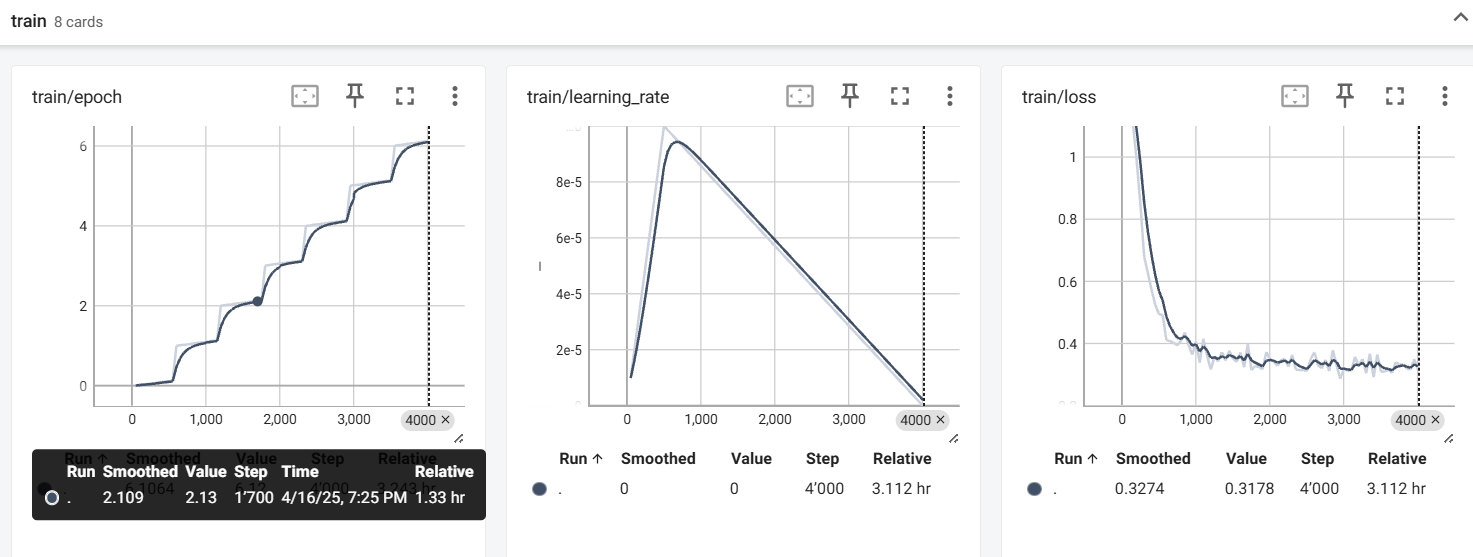
On local machine: ssh -L 6006:localhost:6006 [klp65@login-q-2.hpc.cam.ac.uk](mailto:klp65@login-q-2.hpc.cam.ac.uk) (depending on login node)

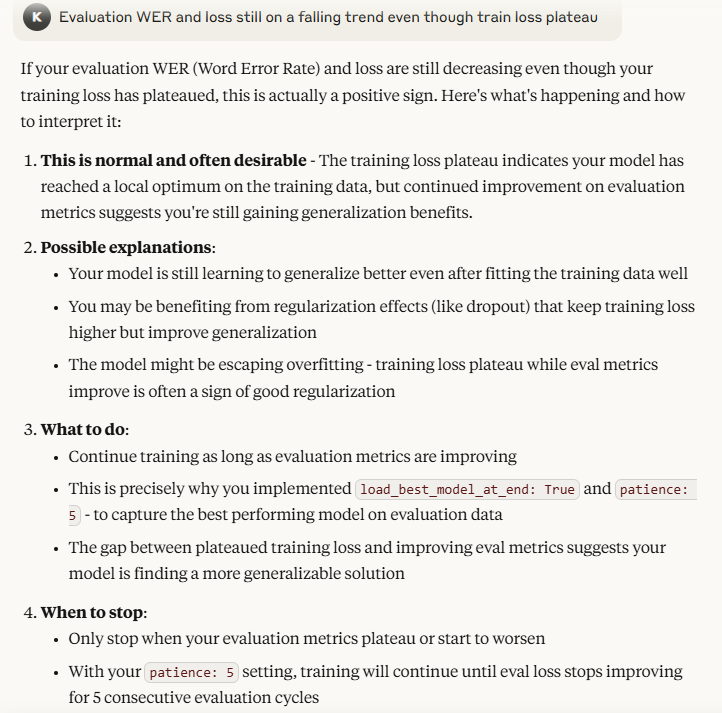
Access port 6006 on local machine.

**Parameter Tuning**

1. **Prompt Tuning**

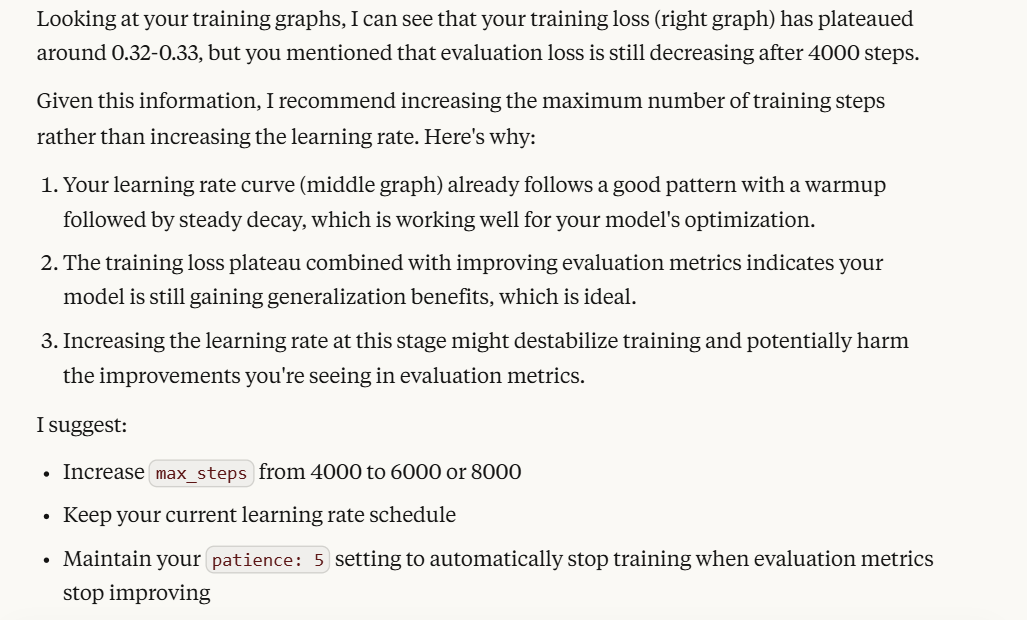
Evaluation WER and loss still on a falling trend even though train loss plateau





Can increase the number of eval steps (1000 steps per evaluation cycle now) – meaning patience is not useful since only 4000 steps in total. Also difficult to observe trend with just 4 datapoints.

Can increase the number of training steps too.



Two changes:

1. Decrease evaluation steps to possibly 100 while maintaining patience = 5
2. Step max\_steps to 8000

Bonus:

1. Try to upgrade transformers, to check if it supports gradient checkpointing
2. If it does that can enable gradient checkpointing to save memory and double batch size from 16 to 32.

Why your results might be poorer:

1. **Effective batch size**: Using 2 GPUs likely doubled their effective batch size compared to your setup. Larger batch sizes can lead to more stable gradient updates and better convergence, especially for speech recognition tasks.
2. **Optimization advantages:** Multi-GPU training with proper synchronization can sometimes result in better optimization dynamics due to gradient averaging across more diverse mini-batches processed in parallel.
3. **Implementation differences:** There might be subtle implementation details that differ:
   1. Different random seeds
   2. Slightly different preprocessing (whisper small instead of large v2)
   3. Additional regularization techniques

Might not want to do full fine-tuning, can perhaps stick to PEFT + encoder/decoder

Take the best then do DA, then add in CSLU

Transformer implementation: <https://github.com/pkailin/whisper-lm-transformers>

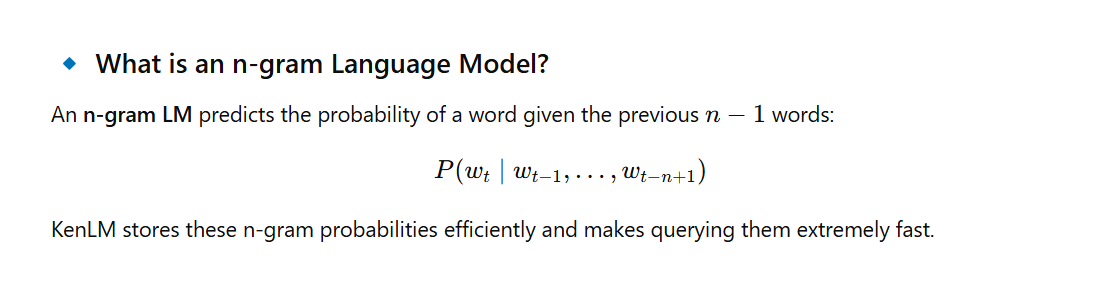
Non-transformer implementation: <https://github.com/pkailin/whisper-lm>

Whisper model + LM:   
test out: <https://github.com/hitz-zentroa/whisper-lm-transformers> (implementation with LLM or KenLM + whisper)

Paper: <https://arxiv.org/pdf/2503.23542>

LLMs might not be worth to finetune at this stage, might want to just integrate: zero-shot LLM and fine-tuned LM.

But KenLM is too simple, might not perform well too:

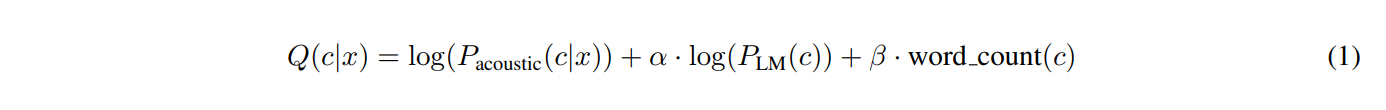


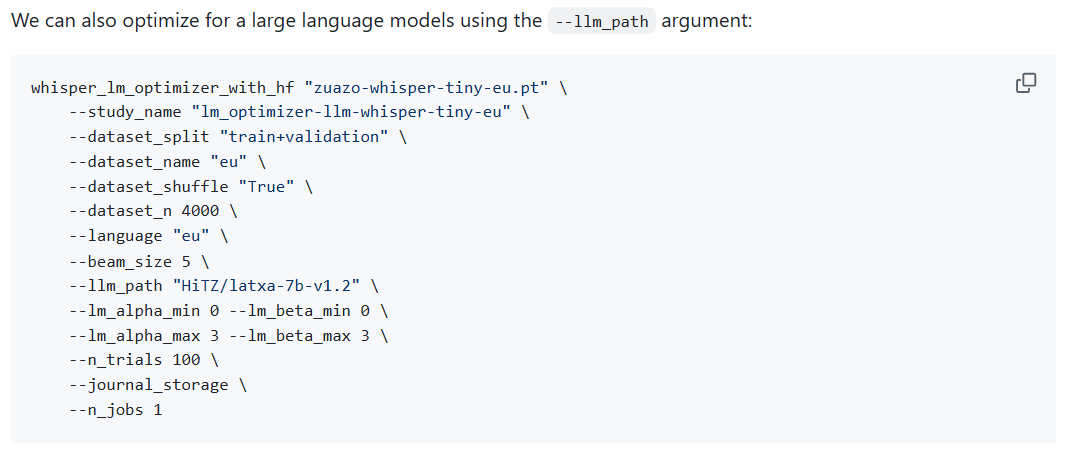
Can use LLM with less parameters:

Whisper small 244M

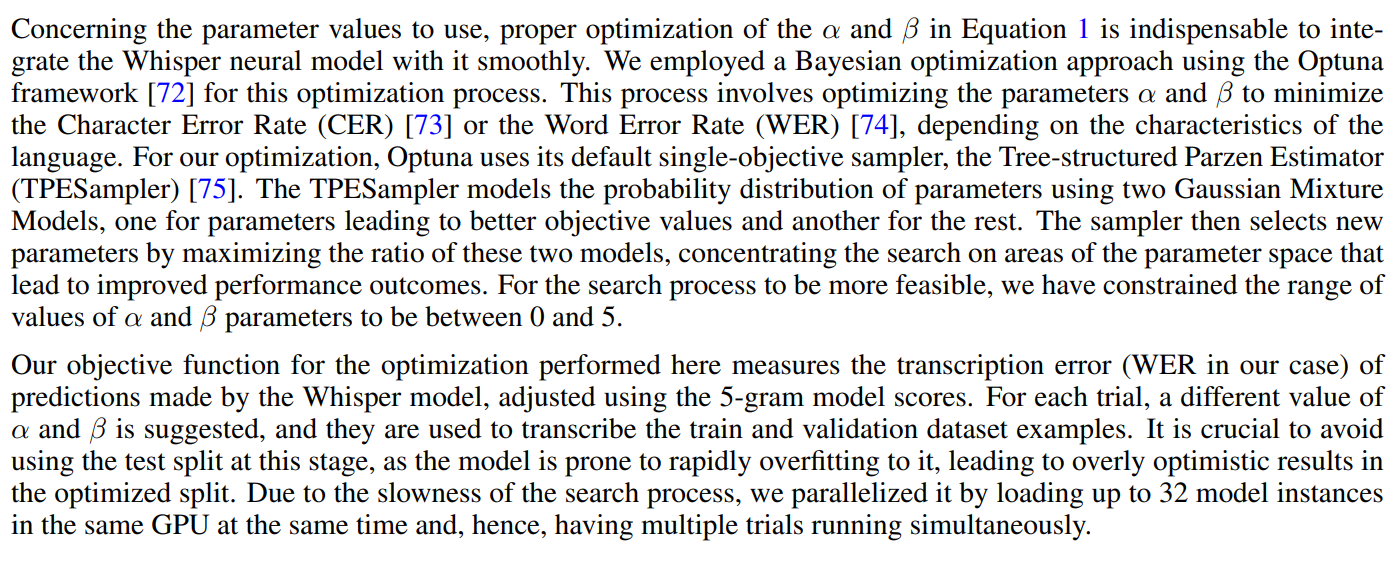
<https://huggingface.co/google-t5>

T5 models: T5-Base is the checkpoint with 220 million parameters. N-best paper by Kate uses T5 base model.





Alpha and beta are tuned based on the procedure below:



Read section 3.2 of the paper.

Try this out with tuned whisper + T5 base LLM in HPC.

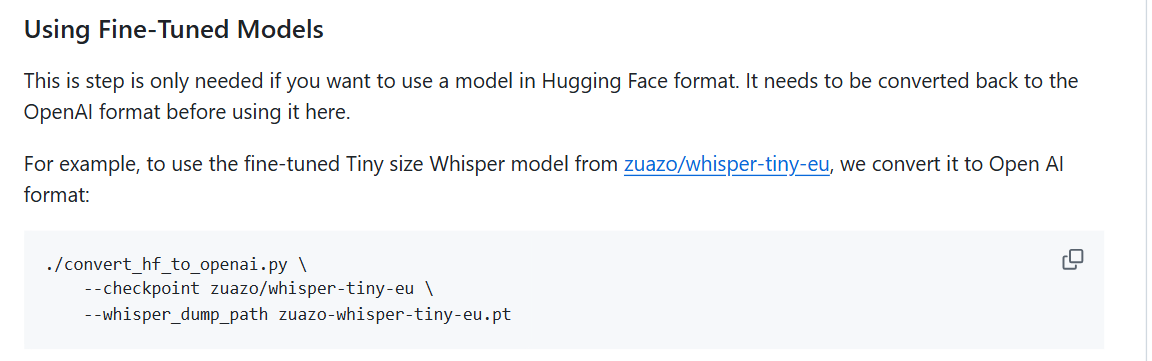
Need to figure out the architecture behind the code and type in report too.

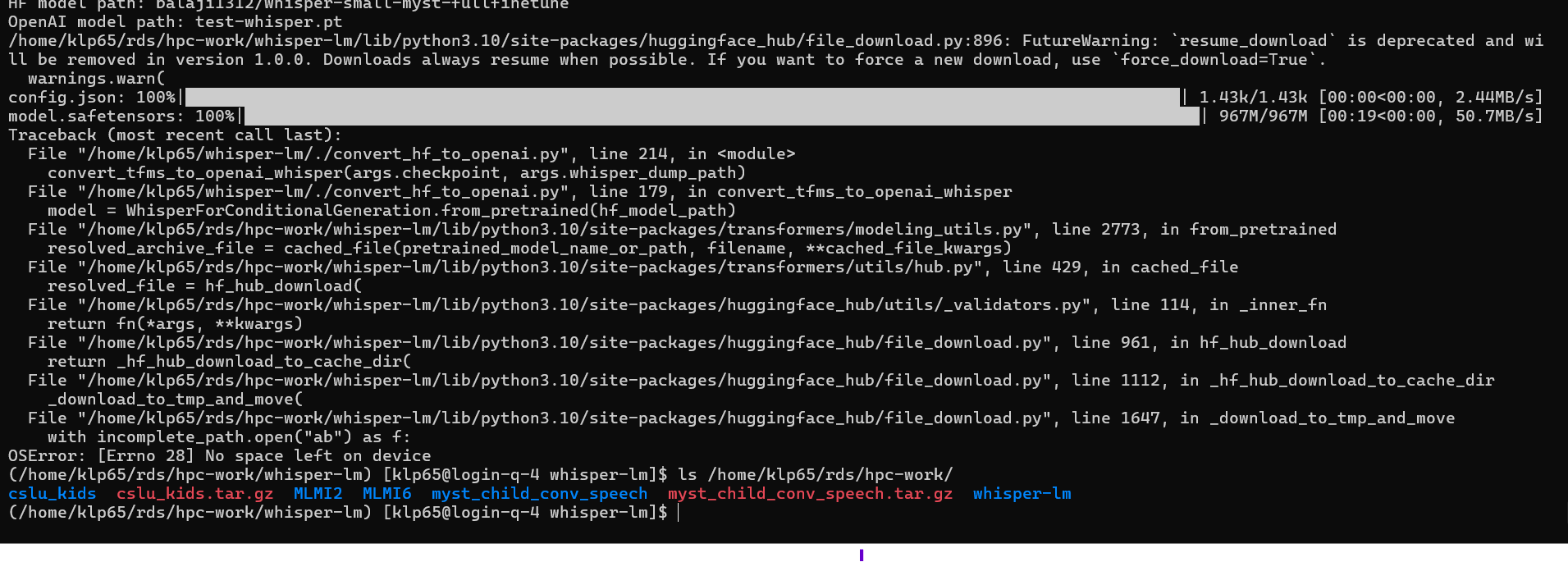
**(whisper lm with transformer setup)**

Python 3.10

conda create --prefix /home/klp65/rds/hpc-work/envs/whisper-lm-env python=3.10 -n whisper-lm

To activate environment: conda activate /home/klp65/rds/hpc-work/whisper-lm-env



****

No space.

shifted all repos to /rds

./convert\_hf\_to\_openai.py --checkpoint balaji1312/whisper-small-myst-fullfinetune --whisper\_dump\_path test-whisper.pt

./whisper\_evaluate\_external.py test-whisper.pt --beam\_size 5 --llm\_path google-t5/t5-base --lm\_alpha 2.7332939 --lm\_beta 0.00178595

module load ffmpeg

conda install -c conda-forge 'ffmpeg>=4.2'

testing LLM github with custom datasets, T5 model and finetuned huggingface whisper model.

Finetuning T5 model: <https://github.com/Shivanandroy/T5-Finetuning-PyTorch>