**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**KHOA/VIỆN ……………..…………..**

**---□**&**□---**



**BÁO CÁO SPEECH TECHNOLOGY**

**Đề tài: AUTOMATIC SPEECH RECOGNITION (ASR)**

***GV hướng dẫn: TS. Nguyễn Thị Thu Trang***  
  ***Nhóm thực hiện: Nhóm 12***

***Lớp: GenAI***

1. **Giới thiệu bài toán**

Bài toán nhận diện tiếng nói tự động (Automatic Speech Recognition – ASR) cho tiếng Việt là một trong những thách thức quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP). Mục tiêu của bài toán là xây dựng một hệ thống có khả năng chuyển đổi chính xác âm thanh tiếng Việt thành văn bản.

Việc phát triển hệ thống ASR cho tiếng Việt gặp nhiều khó khăn do đặc thù của ngôn ngữ như ngữ điệu phong phú, dấu thanh, sự đa dạng vùng miền, và tập hợp từ vựng có nhiều biến thể. Một hệ thống ASR hiệu quả không chỉ cần hiểu rõ đặc điểm ngôn ngữ mà còn phải xử lý tốt các yếu tố như tạp âm, tốc độ nói, hoặc giọng điệu của người nói.

1. **Dataset**

Vietnamese Language and Speech Processing 2020 Speech Recognition Dataset

Kích thước: Khoảng 100 giờ ghi âm tiếng Việt

Định dạng: Cặp file audio (.wav) và transcript (.txt) tương ứng

Đặc trưng của bộ dữ liệu:

* Đa dạng về người nói: Nhiều giọng nói khác nhau, Đa dạng về độ tuổi và giới tính, Bao gồm các vùng miền khác nhau của Việt Nam
* Đa dạng về nội dung: Tin tức, Hội thoại, Đọc văn bản, Phát biểu ...
* Chất lượng âm thanh: Môi trường ghi âm đa dạng, Có thể chứa tiếng ồn nền, Độ rõ của giọng nói khác nhau

1. **Phương pháp đánh giá**

Chỉ số đánh giá: Word Error Rate (WER)

Chỉ số chính để đánh giá hệ thống ASR là Word Error Rate (WER), được biểu diễn bằng phần trăm (%). WER đo lường mức độ chính xác của hệ thống trong việc chuyển đổi lời nói thành văn bản.

WER được tính theo công thức:

WER = (S + D + I) / N × 100%

Trong đó:

* S (Substitutions): số từ bị thay thế sai,
* D (Deletions): số từ bị thiếu so với bản gốc,
* I (Insertions): số từ bị thêm vào không đúng,
* N: tổng số từ trong bản ghi tham chiếu (reference transcript).

Giá trị WER càng thấp chứng tỏ hệ thống nhận diện càng chính xác. Hệ thống lý tưởng có WER bằng 0%, nghĩa là kết quả chuyển đổi khớp hoàn toàn với bản tham chiếu.

1. **Pipeline xử lý**
2. **Cài đặt môi trường**

transformers: Cho mô hình Whisper

datasets[audio]: Xử lý dữ liệu âm thanh

torchaudio: Xử lý âm thanh

deepfilternet: Lọc nhiễu

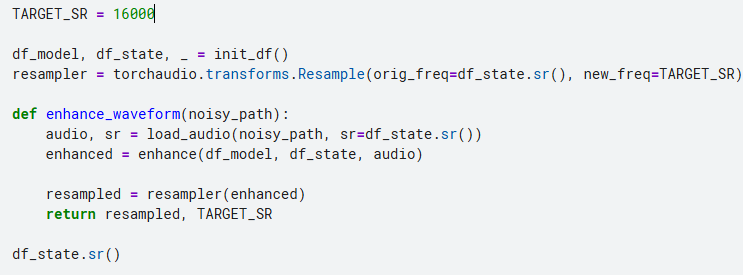
evaluate, jiwer: Đánh giá mô hình

bitsandbytes: Tối ưu bộ nhớ

PEFT: Cho fine-tuning hiệu quả

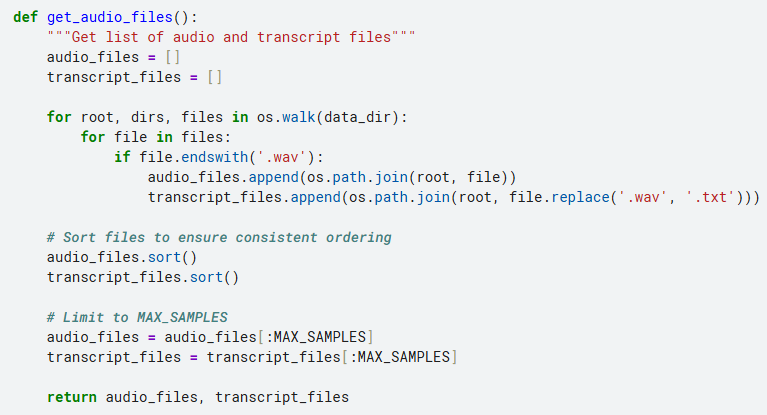
1. **Tiền xử lý dữ liệu**

Xử lý Âm thanh với DeepFilterNet:

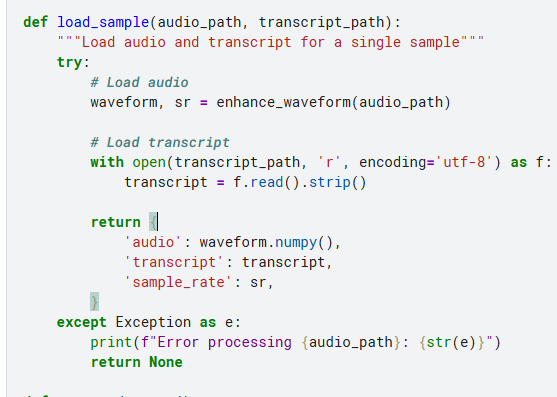


Sử dụng mô hình DeepFilterNet để lọc nhiễu

Tạo resampler để chuyển đổi tần số lấy mẫu, hàm enhance\_waveform đọc file âm thanh, sử dụng DeepfilterNet để lọc nhiễu rồi chuyển tần số lấy mẫu về 16khz

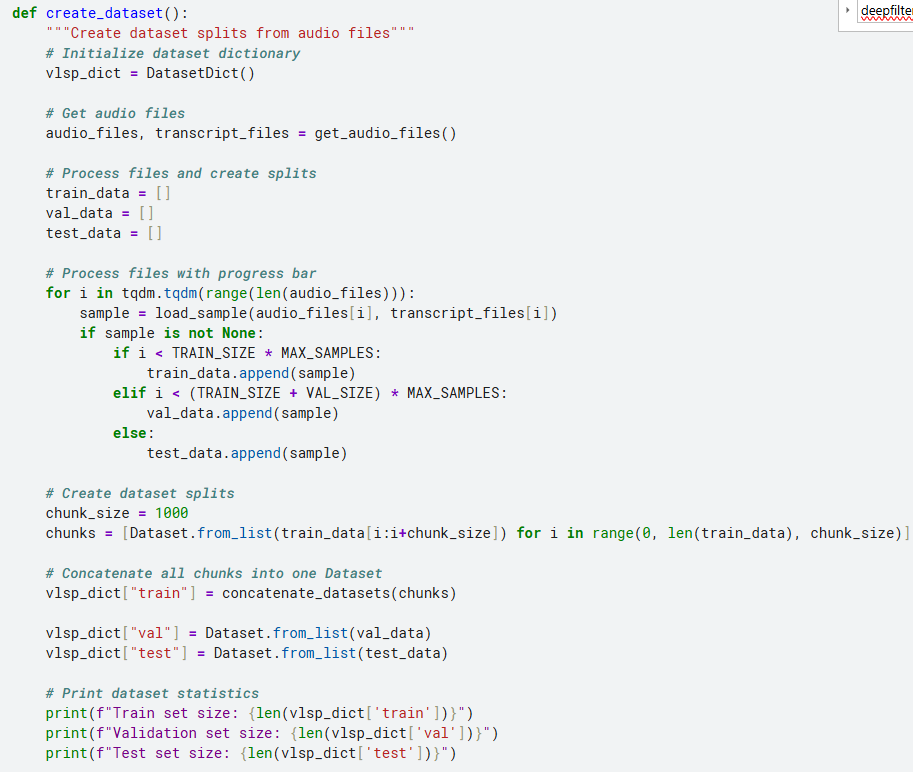
Tổ chức dữ liệu:  


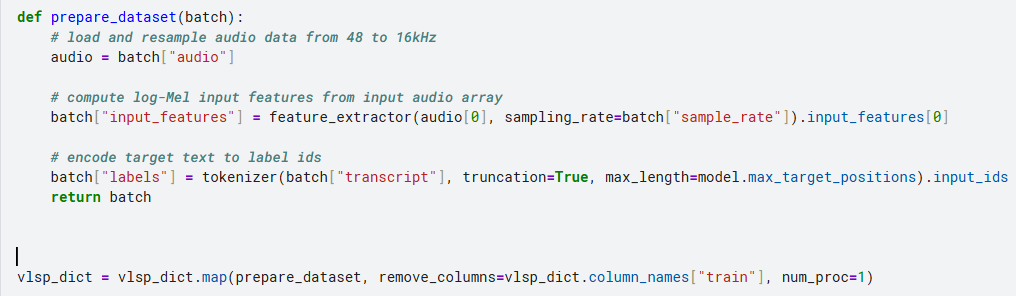
Tìm file .wav và .txt tương ứng, sắp xếp lại để đảm bảo thứ tự và giưới hạn số lượng theo MAX\_SAMPLES

Xử lý từng mẫu:  


Lọc nhiễu và chuẩn hóa tần số với hàm enhance\_waveform

Tạo dataset và chia tập dữ liệu:

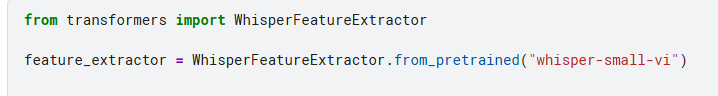
Chia dataset thành 3 phần gồm train, val, test theo tí lệ 7, 2, 1

Chuẩn bị Features cho mô hình   


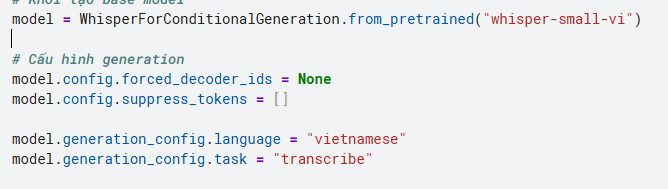
Chuyển đôi âm thanh thành đặc trưng log-mel, mã hóa transcipt thành chuỗi token ids

1. **Chuẩn bị mô hình**

Khởi tạo các thành phần xử lý

Feature extractor để chuyển đổi âm thanh thành đặc trưng log-mel

Tokenizer để chuyển đổi text thành token  
  
processor để kết hợp 2 phần, xử lý đầu vào và đầu ra

Khởi tạo mô hình:  
  
Cấu hình cho nhiệm vụ nhận dạng tiếng việt

Cấu hình Data Collector:

Gom các mẫu thành batch

Padding các sequence về cùng độ dài

Xử lý đặc biệt cho labels trong quá trình training

Đảm bảo format dữ liệu phù hợp với mô hình



1. **Huấn luyện mô hình**

Cấu hình Training Arguments:



Các tham số training quan trọng:

Batch size nhỏ (4) để tiết kiệm bộ nhớ

Gradient accumulation (2) để mô phỏng batch size lớn hơn

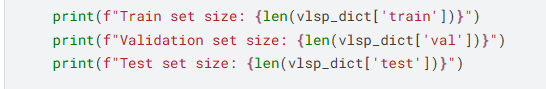
Learning rate thấp (1e-5) phù hợp cho fine-tuning

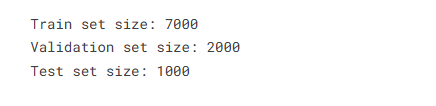
Sử dụng fp16 và gradient checkpointing để tối ưu bộ nhớ

Đánh giá và lưu mô hình theo epoch

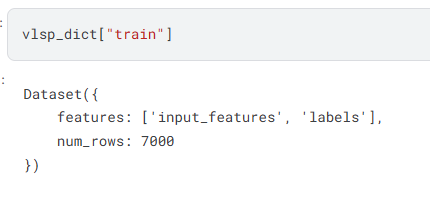
Sử dụng WER làm metric chính để đánh giá

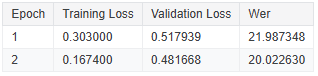
1. **Kết quả**











1. **Nhận xét và kinh nghiệm**
2. **Ưu điểm:**

Sử dụng DeepFilterNet để lọc nhiễu, cải thiện chất lượng âm thanh đầu vào

Áp dụng LoRA để fine-tune hiệu quả, tiết kiệm tài nguyên

Tối ưu bộ nhớ thông qua gradient checkpointing và fp16

1. **Khó khăn:**

Pipeline xử lý dữ liệu linh hoạt và có thể mở rộng

Giới hạn về tài nguyên tính toán, phải sử dụng batch size nhỏ

Cần cân bằng giữa hiệu suất và tài nguyên khi fine-tune mô hình lớn

1. **Hướng phát triển**

Thử nghiệm với các cấu hình LoRA khác nhau

Tăng kích thước dữ liệu training nếu có thêm tài nguyên

Thử nghiệm các kỹ thuật augmentation dữ liệu

Tối ưu hóa hyperparameters

Thử nghiệm với các phiên bản Whisper lớn hơn