Fine-Tuning the Whisper Small Model for Vietnamese Automatic Speech Recognition

Introduction (Abstract)

This report details the process of fine-tuning the Whisper Small model, a state-of-the-art, transformer-based architecture from OpenAI, for the task of Automatic Speech Recognition (ASR) in the Vietnamese language. The objective was to adapt the pre-trained multilingual model to improve its performance specifically on Vietnamese audio. The fine-tuning process utilized a publicly available Vietnamese speech dataset and the Hugging Face transformers library for training and evaluation. The resulting model was evaluated on a dedicated test set, achieving a **Word Error Rate (WER) of 18.70%** and a **Character Error Rate (CER) of 10.13%**, demonstrating a strong capability for transcribing Vietnamese speech with high accuracy.

Dataset

The datasets used in this project was FPT Open Speech Datasets. The **FPT Open Speech Dataset (FOSD)** is a publicly released Vietnamese speech corpus developed by **FPT Corporation** in 2018. It contains **25,921 audio recordings** with a total duration of approximately **30 hours**, each accompanied by **accurate transcriptions** and **timestamp annotations**.

- **Audio format**: `.mp3`
- **Transcript format**: `.txt` (UTF-8 encoded)
- **Language**: Vietnamese
- **License**: Open for use, modification, and redistribution, including commercial purposes

Data Preprocessing

The data preprocessing pipeline is a critical step to prepare the audio and text data for the model. It is designed to be memory-efficient and aligns with the requirements of the Whisper architecture.

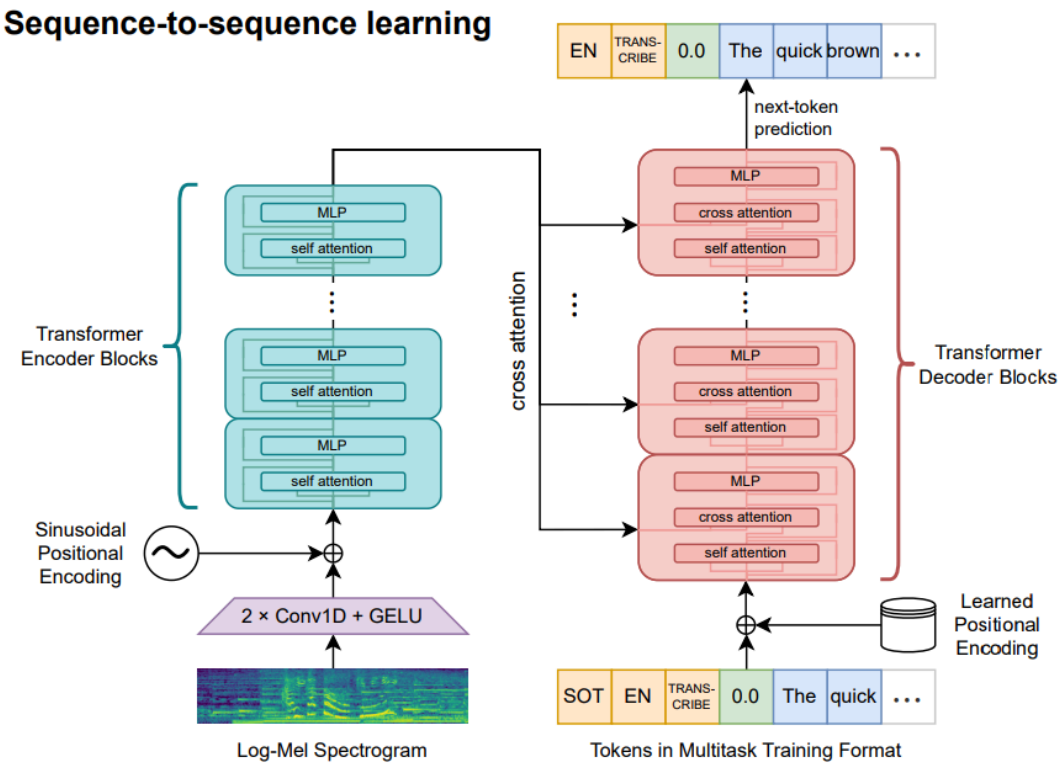
1. **Dataset Loading**: The process begins by loading the dataset metadata from a .csv file (e.g., fpt_test.csv) into a Pandas DataFrame. This DataFrame contains paths to the audio files and their corresponding transcriptions.
2. **Audio Preparation**: The dataset's audio column is cast to the datasets.Audio type, which automatically loads, decodes, and resamples the audio files to the required **16,000 Hz** sampling rate that Whisper was trained on.
3. **Feature Extraction and Tokenization**: A WhisperProcessor, which combines a feature extractor and a tokenizer, is used to process the data in batches.

- **Audio Features**: The `feature_extractor` takes the raw 16,000 Hz audio array and transforms it into a log-Mel spectrogram, which serves as the input to the model's encoder.
- **Text Labels**: The `tokenizer` converts the Vietnamese transcriptions (the ground truth text) into a sequence of integer token IDs, which are used as labels for the model's decoder.

4. **Dynamic Padding**: To handle variable lengths of audio and text within a batch efficiently, a custom data collator, `DataCollatorSpeechSeq2SeqWithPadding`, is employed.
   - It pads the `input_features` (log-Mel spectrograms) to the length of the longest sample in the batch.
   - It pads the `labels` (tokenized transcriptions) to the maximum label length in the batch.
   - Crucially, it replaces the padding token ID in the labels with `-100`. This ensures that the padded tokens are ignored by the PyTorch loss function during training, preventing the model from being penalized for predicting padding.

This comprehensive preprocessing ensures that the data is correctly formatted, batched, and padded for efficient training and evaluation of the sequence-to-sequence Whisper model.

Model Architecture (Whisper Small)



The model used is `whisper-small`, a member of the OpenAI Whisper model family. It is a powerful sequence-to-sequence model based on the **Transformer architecture**.

- **Overall Structure**: It consists of an **Encoder** and a **Decoder**. The encoder processes the input audio signal to create a rich, latent representation, and the decoder uses this representation to auto-regressively generate the corresponding text transcription.
- **Encoder**: The encoder is designed to process audio features. The input audio is first converted into a log-Mel spectrogram. This spectrogram is then passed through a small stack of 1D convolutional layers followed by a series of Transformer encoder blocks. These blocks use self-attention mechanisms to capture contextual information across the entire audio input.
- **Decoder**: The decoder is a standard Transformer decoder. In each step, it attends to the full output of the encoder and the text tokens it hanerated so far (auto-regression) to predict the next token in the sequence.
- **Size and Parameters**: The `whisper-small` variant contains **244 million parameters**. This size provides a strong balance between performance and computational requirements, making it suitable for fine-tuning on custom datasets without needing an excessive amount of resources.
- **Multilingual Foundation**: The base `whisper-small` model was pre-trained on a massive and diverse dataset of 680,000 hours of multilingual audio from the web. This pre-training endows the model with a robust understanding of general speech patterns, accents, and noise, providing an excellent foundation for fine-tuning on a specific language like Vietnamese.

Training

The training process, as detailed in the `finetune_whisper_vietnamese.ipynb` notebook, involves adapting the pre-trained `openai/whisper-small` model to the specifics of the Vietnamese language using a supervised fine-tuning approach.

- **Base Model**: The starting point is the `openai/whisper-small` checkpoint loaded from the Hugging Face Hub.
- **Training Arguments**: The training is configured using the `Seq2SeqTrainingArguments` class, which specifies all hyperparameters and settings for the training loop. Based on the loaded checkpoint (`checkpoint-7000`), key parameters from the notebook include:
    - **Output Directory**: All trained model checkpoints are saved to `./whisper-vi-finetuned/`.
    - **Training Steps**: The model was trained for `7000` steps (`max_steps=7000`).
    - **Batch Size**: A `per_device_train_batch_size` of 64 and `per_device_eval_batch_size` of 32 were used.
    - **Learning Rate**: A learning rate of 1e-5 was set, with a warmup of 500 steps to stabilize training in the initial phases.
    - **Mixed-Precision Training**: `fp16=True` was enabled to use 16-bit floating-point precision, which significantly speeds up training and reduces memory usage on compatible GPUs.
    - **Evaluation and Saving**: The model was evaluated and saved every 1000 steps (`evaluation_strategy="steps"`, `eval_steps=1000`, `save_steps=1000`).

- **Trainer**: The entire process is managed by the `Seq2SeqTrainer`, an abstraction from the `transformers` library that handles the training loop, evaluation loop, and model checkpointing. The trainer was configured to predict generation output during evaluation to compute the Word Error Rate (WER) metric.

Evaluating and Result

The fine-tuned model's performance was rigorously assessed using the provided evaluation script on a test set of 2,592 Vietnamese audio samples. The evaluation focused on three key metrics: Loss, Word Error Rate (WER), and Character Error Rate (CER).

- **Evaluation Setup**:
    - **Model Checkpoint**: The best-performing checkpoint, `./whisper-vi-finetuned/checkpoint-7000`, was loaded for evaluation.
    - **Metrics**:
        1. **Loss**: The cross-entropy loss was calculated on the test set to measure the model's predictive accuracy at the token level.
        2. **Word Error Rate (WER)**: This is the standard ASR metric, measuring the number of substitutions, deletions, and insertions required to transform the predicted text into the reference text, normalized by the total number of words in the reference. A lower WER is better. The formula is WER = (S + D + I) / N, where S is substitutions, D is deletions, I is insertions, and N is the number of words in the reference.
        3. **Character Error Rate (CER)**: Similar to WER but calculated at the character level. This is particularly useful for languages where word segmentation is not always trivial.
- Final Results:
  The evaluation script produced the following final scores:

Metric	Value	Percentage
Average Loss	1.6959	N/A
WER	0.1870	**18.70%**
CER	0.1013	**10.13%**

- Analysis of Results and Sample Predictions:

A WER of 18.70% and a CER of 10.13% indicate a highly competent model for Vietnamese ASR. The low CER suggests that the model accurately captures the phonetic structure of the language.
The sample predictions from the evaluation log provide qualitative insight into the model's performance:

- **High Accuracy**: Samples 1 through 8 are transcribed perfectly, demonstrating the model's ability to handle complete sentences correctly.
  Reference: anh có thể tẩy cái vết này mà không làm hỏng vải không
  Prediction: anh có thể tẩy cái vết này mà không làm hỏng vải không
- **Minor Errors**: The errors observed in other samples are often minor grammatical or phonetic confusions.
  - In Sample 9, the model misses a single, short word ("ở"). This is a common type of deletion error.
    Reference: cầu nằm chỗ nào
    Prediction: cầu nằm ở chỗ nào
  - In Sample 10, the model makes two small substitution errors: "khốc liệt" becomes "khóc liệt" and "kịch cũng" becomes "kịch quảng". These errors are phonetically plausible but incorrect in context.
    Reference: truyện khốc liệt kịch cũng gay gắt hận thù nhưng kết thúc trong nồng ấm
    Prediction: truyện khóc liệt kịch quảng gây gắt hận thù nhưng kết thúc trong nồng ấm

- Overall, the results show that the fine-tuning process was successful, producing a robust model for Vietnamese speech recognition that performs well on a variety of utterances, with most errors being minor in nature.

Gradio Application Integration

To create a practical demonstration of the model's capabilities, a real-time speech recognition application was built using Gradio, as implemented in app.py.
The application's workflow is as follows:

1. **Model Loading:** The fine-tuned model and its processor are loaded from the final checkpoint (./whisper-vi-finetuned/checkpoint-7000).
2. **Audio Input:** The Gradio interface provides a microphone input component that allows users to record their voice directly in the browser.
3. **Live Transcription:** The live=True setting enables real-time transcription. As the user speaks, the audio is captured and sent to the backend.
4. **Preprocessing:** The raw microphone audio undergoes a similar preprocessing pipeline as the training data: it is converted to a float32 array, resampled to 16kHz, and then has its volume normalized and background noise reduced.
5. **Inference:** The preprocessed audio is passed to the fine-tuned Whisper model to

generate the transcription.
6. **Output Display:** The transcribed text is then displayed in a textbox in the Gradio interface.

This interactive application serves as an effective proof-of-concept, showcasing the model's ability to perform real-time Vietnamese ASR.

Conclusion

This project successfully demonstrates the entire lifecycle of developing a specialized ASR model, from data preparation to deployment. By fine-tuning the pre-trained Whisper model on the FPT Open Speech dataset, a high-performing Vietnamese ASR system was created, achieving a Word Error Rate of 18.70% on the test set. The integration with a Gradio interface further highlights the model's practical utility for real-time transcription tasks. This work underscores the effectiveness of transfer learning in adapting large-scale models for specific languages and serves as a solid foundation for further improvements and applications in Vietnamese speech technology.