Computational Linguistics Final Project: Transcribing with Neural Models

Alex Godfrey (ajg367), Sahib Manjal (ssm237), Sahil Hosalli (sh2276)

Google Drive Link:
https://drive.google.com/drive/folders/1ygmjfwjVj8OS6nXvY50ehbc-bbVKUG2b?usp=sharing
This drive folder (shared with Prof. Rooth and TA Lisa's Cornell Email Accounts) contains all files relevant to the project, and we link to files in this writeup where relevant.

Introduction

Neural models rose to dominance in the field of speech-to-text transcription through a rapid evolution beginning in the early 2010s, replacing traditional systems based on hidden Markov models (HMMs) and Gaussian mixture models (GMMs). The shift began with the introduction of deep neural networks (DNNs) for acoustic modeling, which significantly improved recognition accuracy by learning hierarchical representations of speech. This was followed by the development of recurrent neural networks (RNNs), particularly long short-term memory (LSTM) networks, which could capture temporal dependencies in audio more effectively than HMMs. The launch of end-to-end architectures, such as sequence-to-sequence models with attention and later connectionist temporal classification (CTC), eliminated the need for separate pronunciation and language models. Transformer-based models and self-supervised pre-training methods, like wav2vec 2.0, further improved performance and enabled low-resource adaptation, cementing neural approaches as the state of the art across research and commercial applications.

OpenAI's Whisper model, released in 2022, demonstrated the power of large-scale multilingual and multitask training, achieving robust performance across diverse accents and noisy conditions. CMU's OWSM (One Whisper Style Model) further extends these ideas by supporting simultaneous language identification and multilingual transcription and translation using a unified architecture optimized for real-time inference.

Modern neural speech-to-text models can achieve word error rates (WER) as low as 2–5% on clean, read speech (like the LibriSpeech test-clean benchmark), often surpassing human transcription accuracy under ideal conditions. The goal of this project is to explore how good these models are at the task of recognizing the speech in the `.wav` files produced by past and current students of this class, and to compare the transcriptions to the older methods used by Professor Rooth.

Approach 1: OWSM in Google Colab (Link)

To transcribe the .wav files from years 2021 and 2023, we first used OWSM (Open Whisper-style Speech Models), a neural speech recognition model developed by CMU's WAVLab. We began with an open-source Colab notebook provided by the CMU WAVLab team (link), which demonstrated how to load the model and transcribe single .wav files using ESPnet2's Speech2Text interface.

Modifications:

We adapted this base notebook to handle batch transcription of a large number of .wav files by:

- Recursively loading all files in a given Drive directory
- Converting audio to 16kHz mono format using librosa
- Using beam search decoding and retaining the top 5 hypotheses (N-best list) per file
- Formatting outputs into a standardized ID system (aiden-a-1-1, aiden-a-1-2, etc.)
- Saving results into a CSV file and later a text format suitable for downstream evaluation with our ps6 code.

Compute & Runtime

- ~1B parameter, transformer-based neural network model (owsm_v3.1_ebf)
- 20 minutes on a Colab GPU instance per ~350 wav files of data
- Nearly 100% of available GPU VRAM used (15 GB available), with peak memory observed during model loading

Note that most of the compute time was spent decoding each file using beam search, and GPU acceleration significantly reduced latency compared to CPU-only inference (see Approach 2).

For this project, we used OWSM in ASR mode, specifying <eng> (input language) and <asr> (automatic speech recognition task) as control symbols. Each .wav file was passed to the model using the Speech2Text class from the ESPnet2 library, with beam search decoding enabled and the top 5 transcription hypotheses retained. The transcriptions were saved in a standardized format (name-a-1-1, name-a-1-2, etc.) and passed into our logic-based evaluation pipeline.

Approach 2: OWSM Locally

We mimic the same approach as Approach 1, we now discuss the compute and runtime results of it below.

When running this file locally on a Mac, we had to patch ESPnet itself (what OWSM uses) to allow for MPS (macOS GPU API on Apple ARM Silicon) and to use float32 (MPS is incompatible with float64). However, there is a current issue with PyTorch where torch.stft() does not work, so we partially run on MPS and then fallback to CPU when we get this error.

Compute Comparison

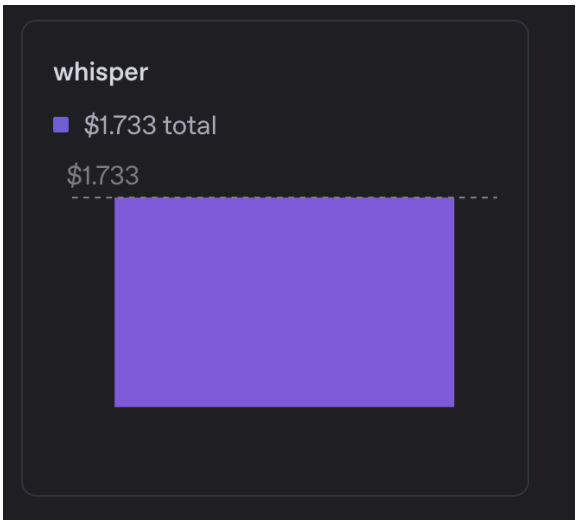
- Purely CPU: 100 seconds per wav file, so 10 hours per ~350 wav files of data
- MPS with fallback CPU: 10 seconds per wav file, so 1 hour per ~350 wav files of data.

This is a 300% decrease in performance compared to Google Colab but it is still computationally inexpensive enough to run locally on this scale. We expect that once the PyTorch issue is resolved, the Mac MPS will be equivalent to, if not better, than the Google Colab. The current estimated MPS usage, so percentage of runtime operations done on it, is around 10% (found from profiling the time spent in each function before and after MPS addition). Since we are running on a relatively small amount of audio files, we exclusively use Google Colab's free but limited runtime T4 GPU.

Approach 3: OpenAI Whisper API

OpenAI is all the buzz right now, so we wanted to also just use their Whisper API to see how well things got transcribed. In about a 100 line Python script, we could iteratively call the API on the .wavs from '21, '23, and in about an hour (one transcript takes about 1 second, but we kept getting rate limited), we had complete transcripts for each year.

OpenAI doesn't explicitly expose an "n-best" functionality in their API (according to this Github thread, you have to run the model locally to beam search like we did with OWSM), so to get the top five transcripts we increasingly varied the temperature to get possibly different transcriptions. Temperature in transformer-based models refers to the randomness with which output tokens are chosen, with a temperature of 0 resulting in an entirely deterministic result. Note that most of the transcripts were the same even as the temperature increased, indicating that there were extremely high logit values/probabilities of single output tokens. Here was the cost of using the whisper API for ~700 wav files (3500 transcriptions):



Results

Now it's time to see the parsing coverage based on OWSM vs Whisper vs Ground Truth entered. We repurposed the PS6 problem 1 into this script that parsed the n-best transcriptions from OWSM and Whisper, and also ran on the ground truths provided. You can see all nine .c2 files here produced from OWSM, Whisper, or just the human-entered transcripts (ground truth) for '21, '23, and '25.

2021

Ground Truth: 96.3% parse coverage

```
(base) alex@dhcp-vl2041-34824 ps6 % python find-parseable-transcripts.py
parsing: 349it [00:42,  8.31it/s]

coverage = 96.3%  (336/349)
```

OWSM: 86.8% parse coverage

```
(base) alex@dhcp-vl2041-34824 ps6 % python find-parseable-transcripts.py
parsing: 1740it [00:41, 41.79it/s]

coverage = 86.8%  (302/348)
```

Note that 1 audio file was not transcribed by OWSM due to GPU memory allocation errors

Whisper: 68.5% parse coverage

```
(base) alex@dhcp-vl2041-34824 ps6 % python find-parseable-transcripts.py
parsing: 1745it [00:28, 61.32it/s]

coverage = 68.5%  (239/349)
```

2023

Ground Truth: 74.3% parse coverage

```
(base) alex@dhcp-vl2041-34824 ps6 % python find-parseable-transcripts.py
parsing: 339it [00:07, 43.19it/s]

coverage = 74.3%  (252/339)
```

OWSM: 70.7% parse coverage

```
(base) alex@dhcp-vl2041-34824 ps6 % python find-parseable-transcripts.py
parsing: 1640it [00:09, 173.28it/s]

coverage = 70.7%  (232/328)
```

Whisper: 59.8% parse coverage

```
(base) alex@dhcp-vl2041-34824 ps6 % python find-parseable-transcripts.py
parsing: 1640it [00:06, 248.13it/s]

coverage = 59.8%  (196/328)
```

2025

Ground Truth: 28.5% parse coverage

```
(base) alex@dhcp-vl2041-5784 ps6 % python find-parseable-transcripts.py
parsing: 338it [00:39,  8.51it/s]

coverage = 28.5%  (93/326)
```

OWSM: 28.9% parse coverage

```
(base) alex@dhcp-vl2041-5784 ps6 % python find-parseable-transcripts.py
parsing: 1245it [00:34, 35.91it/s]

coverage = 28.9%  (72/249)
```

Whisper: 2.0% parse coverage

```
(base) alex@dhcp-vl2041-5784 ps6 % python find-parseable-transcripts.py
parsing: 1245it [00:01, 953.30it/s]

coverage = 2.0%  (5/249)
```

In terms of interpreting these results, the base Kaldi implementation we did in PS6 got 60% coverage. The ground truth of course has much higher coverage; in theory it should be 100% but not every submitted sentence was parseable. So the goal is to be as close to the ground truth percentage as possible.

In terms of 2025 data, basically as a class we did a poor job of having correct transcripts in the first place. The ground truth only scored 28.5%. The OWSM is probably slightly higher because the 2025 list we got was incomplete, some students didn't submit all their files correctly so that's probably the reason for the small discrepancy.

Surprisingly, Whisper was quite disappointing. This is probably because it's 1.5 years old, and also because the API doesn't expose the beam search functionality, so we have a weaker n-best implementation compared to that of Kaldi or OWSM. OWSM, however, performed exceedingly well, both when you compare it to Kaldi and the ground truth. There are also newer variants of Whisper, such as Whisper Turbo, as well as other OpenAI models, including GPT-4o (as an omnimodal LLM, accepts audio among many other modalities as input and output), which may have better performance but do not have as much or any API support at the moment.

Conclusion

In this project, we explored three state-of-the-art neural approaches to speech-to-text transcription—CMU's OWSM (in both cloud and local environments) and OpenAI's Whisper API—and compared them to the ground truth transcripts of previous student speech data. Our findings show that OWSM, particularly when run in Google Colab with GPU acceleration and beam search decoding, performs impressively well, coming close to the parseability of human-entered ground truth. This supports the promise of modern end-to-end neural models that integrate acoustic, pronunciation, and language modeling into a unified architecture. Despite the compute limitations of running OWSM locally, we still achieved solid performance with only moderate slowdown, which opens up possibilities for high-quality offline transcription without cloud dependency.

Whisper, despite its popularity and robustness in noisy and multilingual settings, underperformed in our specific evaluation task. Its lower parse coverage appears to stem from both its age and the constraints of its API, which lacks direct support for beam search or n-best outputs—features that proved critical in achieving high-quality transcriptions with OWSM. Ultimately, this project not only showcases the rapid evolution and ongoing advantages of neural transcription models but also illustrates how nuanced implementation choices (such as decoding strategy and model access) can significantly impact downstream accuracy in structured linguistic tasks. Our results suggest that neural models currently outperform traditional approaches for ASR tasks, and improvement is likely to continue, with these models likely matching or even outperforming humans at such tasks in the near future.