Assignment11-NLP

November 15, 2025

1 Assignment 11: Text processing - NLP

2 Automatic Speech Recognition

Speech Recognition, also known as Automatic Speech Recognition (ASR), is a technology that converts spoken language into written text. It plays a crucial role in bridging the gap between human speech and computer understanding, enabling applications like virtual assistants, real-time transcription, and voice-controlled systems. ASR systems typically process audio signals by extracting features such as spectrograms, then using Artificial Intelligence-based model sto decode these signals into meaningful word sequences. In the context of natural language processing (NLP), ASR provides the first step in enabling machines to understand and respond to spoken language.

3 Dataset

LibriSpeech is a widely used speech recognition dataset designed for training and evaluating automatic speech recognition (ASR) systems. It was introduced by Vassil Panayotov et al. in 2015 and has since become one of the standard benchmarks in the ASR research community.

LibriSpeech is a publicly available speech recognition dataset consisting of approximately 1,000 hours of English speech derived from audiobooks that are part of the LibriVox project. It was created to support training and evaluation of automatic speech recognition (ASR) systems. The audio is sampled at 16 kHz and is accompanied by accurate, time-aligned transcripts. LibriSpeech includes multiple subsets such as "train-clean," "train-other," "dev-clean," and "test-clean," allowing researchers to benchmark models under varying levels of audio quality and complexity. Its accessibility, size, and quality have made it one of the most widely used datasets in speech and NLP research.

Note on Dataset Usage:

The corresponding dataset used in this assignment is a subset of the LibriSpeech corpus and has been specifically selected and uploaded on Canvas. To ensure consistency in evaluation and reproducibility, you must not use any external data or download additional LibriSpeech files. Only use the audio and transcript files provided in the assignment folder on Canvas.

4 Step-by-Step Guide: From Audio to Clean Text

5 Step 1: Read the Audio Files

Assume the corresponding dataset includes:

- Several .flac audio files (e.g., 84-121123-0000.flac)
- A transcript file (e.g., transcripts.txt)

[ ]:
```
!pip install openai-whisper
```

```
Collecting openai-whisper
  Downloading openai-whisper-20240930.tar.gz (800 kB)
                           0.0/800.5

kB ? eta -:--:--

286.7/800.5 kB 8.5 MB/s eta 0:00:01
                     800.5/800.5 kB
11.6 MB/s eta 0:00:00
  Installing build dependencies … done
  Getting requirements to build wheel … done
  Preparing metadata (pyproject.toml) … done
Requirement already satisfied: numba in /usr/local/lib/python3.11/dist-packages
(from openai-whisper) (0.60.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(from openai-whisper) (2.0.2)
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages
(from openai-whisper) (2.6.0+cu124)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(from openai-whisper) (4.67.1)
Requirement already satisfied: more-itertools in /usr/local/lib/python3.11/dist-
packages (from openai-whisper) (10.7.0)
Requirement already satisfied: tiktoken in /usr/local/lib/python3.11/dist-
packages (from openai-whisper) (0.9.0)
Requirement already satisfied: triton>=2.0.0 in /usr/local/lib/python3.11/dist-
packages (from openai-whisper) (3.2.0)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in
/usr/local/lib/python3.11/dist-packages (from numba->openai-whisper) (0.43.0)
Requirement already satisfied: regex>=2022.1.18 in
/usr/local/lib/python3.11/dist-packages (from tiktoken->openai-whisper)
(2024.11.6)
Requirement already satisfied: requests>=2.26.0 in
/usr/local/lib/python3.11/dist-packages (from tiktoken->openai-whisper) (2.32.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from torch->openai-whisper) (3.18.0)
Requirement already satisfied: typing-extensions>=4.10.0 in
/usr/local/lib/python3.11/dist-packages (from torch->openai-whisper) (4.13.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-
```

```
packages (from torch->openai-whisper) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages
(from torch->openai-whisper) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages
(from torch->openai-whisper) (2025.3.2)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch->openai-whisper)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch->openai-whisper)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch->openai-whisper)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch->openai-whisper)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch->openai-whisper)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch->openai-whisper)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch->openai-whisper)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch->openai-whisper)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch->openai-whisper)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in
/usr/local/lib/python3.11/dist-packages (from torch->openai-whisper) (0.6.2)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in
/usr/local/lib/python3.11/dist-packages (from torch->openai-whisper) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch->openai-whisper) (12.4.127)
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch->openai-whisper)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-
packages (from torch->openai-whisper) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch->openai-
whisper) (1.3.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from
```

```
requests>=2.26.0->tiktoken->openai-whisper) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests>=2.26.0->tiktoken->openai-whisper) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from
requests>=2.26.0->tiktoken->openai-whisper) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from
requests>=2.26.0->tiktoken->openai-whisper) (2025.4.26)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from jinja2->torch->openai-whisper)
(3.0.2)
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4
MB)
                         363.4/363.4 MB
3.7 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (13.8 MB)
                         13.8/13.8 MB
63.0 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (24.6 MB)
                         24.6/24.6 MB
34.9 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (883 kB)
                         883.7/883.7 kB
41.9 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl
(664.8 MB)
                         664.8/664.8 MB
1.4 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl
(211.5 MB)
                         211.5/211.5 MB
4.5 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-
manylinux2014_x86_64.whl (56.3 MB)
                         56.3/56.3 MB
15.1 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl (127.9 MB)
                         127.9/127.9 MB
6.8 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-
manylinux2014_x86_64.whl (207.5 MB)
                         207.5/207.5 MB
5.3 MB/s eta 0:00:00
```

```
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (21.1 MB)
                          21.1/21.1 MB
54.5 MB/s eta 0:00:00
Building wheels for collected packages: openai-whisper
  Building wheel for openai-whisper (pyproject.toml) … done
  Created wheel for openai-whisper: filename=openai_whisper-20240930-py3-none-
any.whl size=803405
sha256=35b0d9406cc82f6ba92bd4ad944d0a9c1ddd713384f08ac54457893d0392f7ae
  Stored in directory: /root/.cache/pip/wheels/2f/f2/ce/6eb23db4091d026238ce7670
3bd66da60b969d70bcc81d5d3a
Successfully built openai-whisper
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12,
nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-
cuda-cupti-cu12, nvidia-cublas-cu12, nvidia-cusparse-cu12, nvidia-cudnn-cu12,
nvidia-cusolver-cu12, openai-whisper
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
```

```
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-
cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtime-cu12-12.4.127
nvidia-cudnn-cu12-9.1.0.70 nvidia-cufft-cu12-11.2.1.3 nvidia-curand-
cu12-10.3.5.147 nvidia-cusolver-cu12-11.6.1.9 nvidia-cusparse-cu12-12.3.1.170
nvidia-nvjitlink-cu12-12.4.127 openai-whisper-20240930
```

Question 1: Using Whisper, load Audio and Transcribe. In addition, can you answer the following questions:

- What is the sampling rate of the provided audio files, and why is it important for speech recognition?
- What is the duration of each audio file?
- What is the bit depth of the audio files, and how does it affect the quality of speech recognition?
- What is the file size of each audio file, and how might the size relate to audio quality or length?
- For each audio, plot it over time.
- List item

For every question, be sure to anlayze and discuss your response.

Note: For answering this questions, python coding is needed. You can use libraries, such as librosa, or pydub. For more information, please refer to:

Librosa: https://librosa.org/doc/latest/index.html

pydub: https://github.com/jiaaro/pydub

You can refer to tutorials at:

https://youtu.be/B31RiiRt_TE?si=76VKAir9xwG8hH2V

https://youtu.be/vJ_WL9aYfNI?si=Hx8OFbHfqJn07uQa

6 Step 2: Text Preprocessing

Question 2: Create a pipeline to clean texts. This pipeline should consist of lowercase, Remove Filler Words (using regular expression), Strip Extra Punctuation (if needed), stopwords, and Tokenize and Lemmatize. Be sure to use SpaCy for answering this question. * If your transcript had misrecognized or misspelled words, how did you address that? Could spell-checking or correction be integrated into your pipeline?

- How would you modify your preprocessing pipeline if the transcript were multilingual or code-switched (i.e., contained multiple languages)?

For every question, be sure to anlayze and discuss your response.

Question 3: Using EDA techniques, answer the following questions:

- **Basic EDA**

1. Visualize the top 20 most frequent words in the transcriptions. What do you observe?
2. Are there words that appear only once (hapax legomena)? What might they indicate?

- **Audio-Specific EDA**

1. Plot a waveform or spectrogram of one audio file. What do you observe in terms of intensity or frequency distribution?
2. Is there a pattern in speaking speed (e.g., words per second)? Does this vary a lot across files?
3. Are there common filler words or disfluencies in the transcripts (e.g., "uh", "um", "you know")? Count and analyze.

Question 4: using feature extraction technqiues:

- **Text-Based Feature Extraction**

1. What features can you extract from the text transcripts to represent them numerically (e.g., TF-IDF, bag-of-words, n-grams)? Use two techniques and compare your results.

2. Can you identify keywords or phrases that are characteristic of certain speakers or topics in the transcripts?

- **Audio-Based Feature Extraction**

1. What audio features could be extracted using MFCCs?
2. Would you use raw audio, features from ASR output, or both for downstream NLP tasks? Justify your choice.

For every question, be sure to anlayze and discuss your response.

Note: What is MFCC?

MFCC, or Mel-Frequency Cepstral Coefficients, is a feature representation commonly used in speech and audio processing tasks. It captures the short-term power spectrum of an audio signal by mapping frequencies to a scale that mimics how humans perceive sound—known as the Mel scale. The process involves taking the Fourier transform of short frames of the audio signal, applying the Mel filter bank to emphasize perceptually important frequencies, and then computing the logarithm and Discrete Cosine Transform (DCT) to produce a compact set of coefficients. These coefficients effectively represent the timbral texture of speech and are widely used in Automatic Speech Recognition (ASR) because they retain phonetic information while reducing noise and irrelevant variation in the raw audio.

7 Step 3: Evaluation

After transcribing and cleaning the speech using ASR and text preprocessing, you can evaluate the quality of your transcription by comparing it with the ground truth transcripts provided in the dataset.

This is useful to:

- Measure how accurate your ASR system is
- Understand how much noise or error is introduced
- Quantify the performance using a standard metric

Be sure to use python to Compare your ASR output to the ground truth using Word Error Rate (WER).

Question 5: Word Error Rate (WER) is a standard metric in ASR that tells you how different your ASR output is from the reference transcript. Be sure to anlayze and discuss your response.

Note: You can calculate this metric using jiwer tool.

More details can be found at:

1. https://pypi.org/project/jiwer/
2. https://github.com/jitsi/jiwer