LAB EVAL 1
UCS749: Conversational AI: Speech Processing and Synthesis

B.V. Raghav

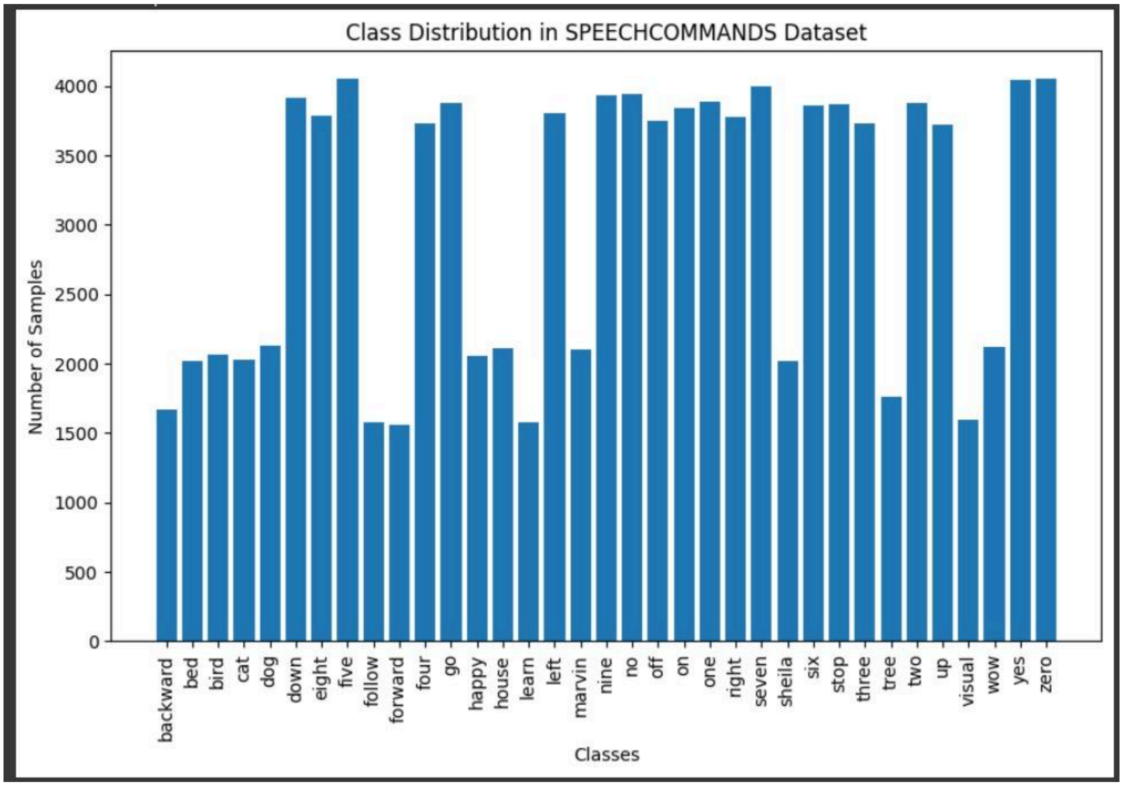
[2024-09-11 Wed]

Summary of Paper

The document details the Speech Commands dataset created for evaluating keyword spotting systems. It outlines the necessity for a dataset specialized for speech recognition tasks that are distinct from full-sentence recognition. The dataset aims to facilitate development and testing of models that efficiently recognize specific words or phrases, crucial for voice-activated interfaces. It also describes the dataset's structure, the process of data collection, verification, and the challenges of constructing a reliable keyword spotting dataset.

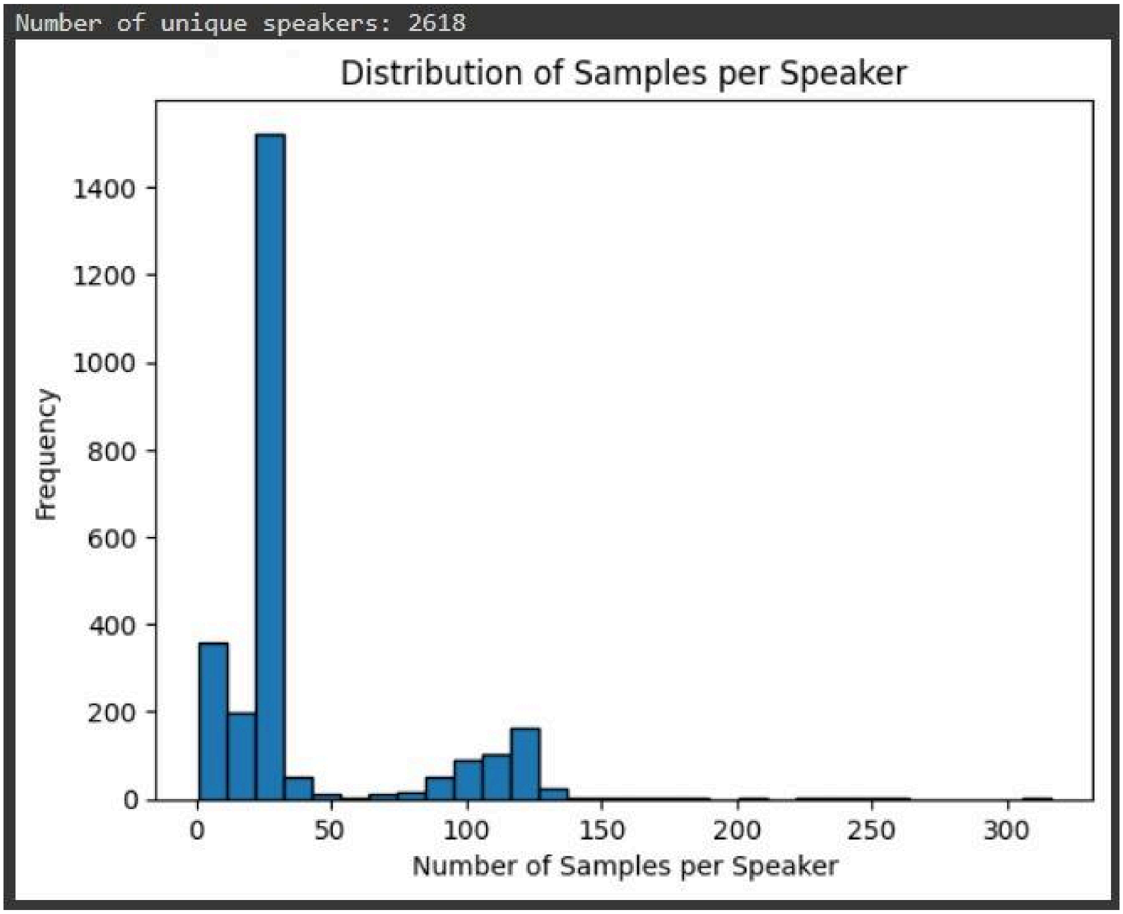
Statistical Analysis

1. **Class distribution of SPEECHCOMMANDS dataset:**



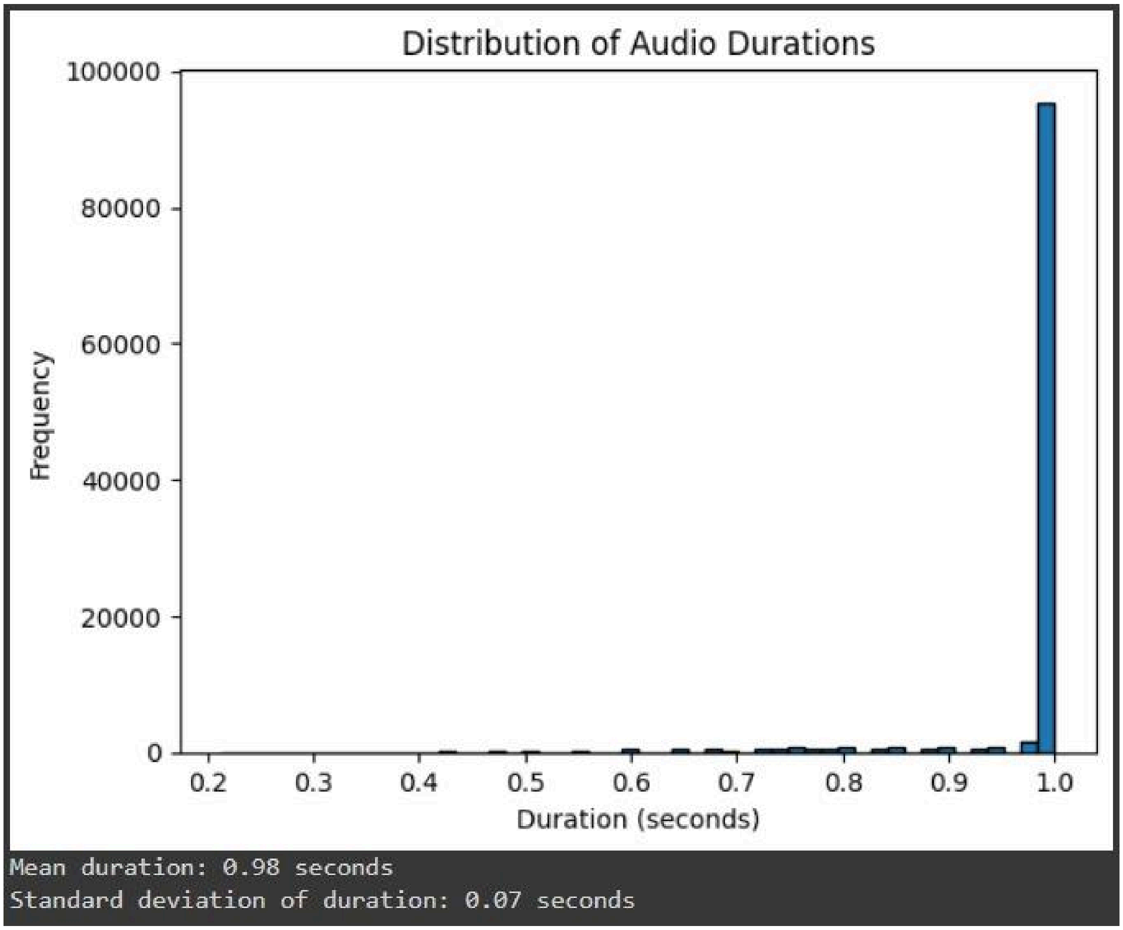
X axis : Classes   Y axis : Number of Samples

2. Distribution of Samples per Speaker:



X axis : Number of Samples per Speaker Y axis : Frequency

3. **Distribution of Audio Durations:**



X axis : Duration(seconds) Y axis : Frequency

1. Clarity of Thought Process and Presentation:

The goal is to record around 30 samples of each spoken command in a consistent, structured manner. The thought process involves:

- Designing a simple pipeline to record commands using a microphone.
- Creating a system that associates each recording with a command label and unique user ID.
- Saving the recordings in a structured dataset format similar to *SPEECHCOMMANDS*.

2. Data Processing Skills:

Data processing involves handling audio files, ensuring they are properly labeled, and stored in an appropriate format (e.g., WAV files). Skills demonstrated include:

- Capturing audio at the correct sample rate (16 kHz for consistency with speech datasets).
- Preprocessing audio to ensure consistent length (typically one second per command).
- Organizing the recordings into folders labeled by command.

Python libraries like `pyaudio` or `sounddevice` are used to record audio, and `os` and `wave` help with file management.

3. Model Fine Tuning/Training Skills:

Training a model for keyword spotting involves building a model (e.g., CNN-based) and fine-tuning it using the collected dataset. It also includes:

- Splitting the dataset into training and testing sets.
- Using a suitable loss function (like CrossEntropy) and optimizer (like Adam).
- Adjusting hyperparameters like learning rate, batch size, and number of epochs.

Transfer learning can be applied to leverage pre-trained models on similar tasks, speeding up the training process and improving accuracy.

4. Details of Progress:

Problems Encountered:

- **Microphone latency or synchronization issues:** Adjusted the recording intervals and set up automatic trimming of excess silence using audio processing libraries like `librosa`.
- **Consistency in labeling:** Used automated label assignment based on file names and directory structures.

Solutions:

- Introduced a simple graphical user interface (GUI) to simplify the recording process.
- Implemented a verification step for recording quality to reduce errors.

5. How Adaptable is the Pipeline?

The pipeline is adaptable in that:

- A new user can follow the same steps to record their voice, with the dataset structure automatically handling user IDs and labels.

- You can specify new commands by simply updating a list, and the script will handle the rest.
- Synchronizing with a timer makes it easy for anyone to use the system without complex setups.

6. How Scalable is the Approach?

Scaling the dataset to multiple voices is straightforward:

- The system supports multiple user IDs and can easily incorporate additional speakers.
- New commands can be added without restructuring the dataset by simply recording new samples and assigning them appropriate labels.
- You can crowdsource data collection using a similar pipeline by sharing the recording script, making it highly scalable.

7. Strengths and Shortcomings:

Strengths:

- The system is easy to use and can be adapted by new users without technical expertise.
- The dataset structure is modeled after a standard like *SPEECHCOMMANDS*, making it compatible with many speech recognition models.
- It's scalable and adaptable for larger datasets.

Shortcomings:

- It requires manual intervention for each recording session, which could be automated further.
- Background noise may affect recordings, requiring additional noise filtering or preprocessing.

This approach provides a simple, scalable way to create a speech dataset, easily adaptable for various users and commands, with room for improvements in automation and noise handling.

Results:

Before fine tuning:

Top one error : 88%

After fine tuning:

Top one error: 50%