Spoken Language Identification Using xVectors

Lohesh M - Esvar Ram Kumar P - Rathna Sabapathy P

Objective

Develop a real-time system to identify the spoken language in audio samples by converting signal features like MFCCs into fixed-dimensional embeddings (xVectors) via a neural network (e.g., TDNN). The system should robustly handle diverse conditions such as noise, varying accents, and device variability.

Motivation

Why can't we directly compare the acoustic features of the signal and features of specific languages? It is because features like MFCCs are affected by noise, recording devices and transmission channels, same speaker's voice may have different mood so there exists variability. So direct comparison of these is not advisable and computationally expensive.

Therefore, we represent these features as a fixed dimensional vector (xVector), variability is reduced, comparisons are simplified, and temporal context across frames is captured. Moreover, this approach reduces computational overhead compared to frame-level processing by collapsing variable-length inputs into fixed-length embeddings. Overall, the spoken language identification has various application and requires real time and accurate results. xVector performance in tasks like speaker verification and identifications inspires their application in language identification.

Methodology

Dataset Collection and Preprocessing

1) Collect language-specific audio files with corresponding labels.
2) Perform preprocessing, such as noise reduction and normalization, to ensure quality.

Feature Extraction

1) Extract features from the audio files (e.g., MFCCs).

xVector Extraction (available in kaldi toolkit)

1) Pass extracted features through a Time Delay Neural Network (TDNN).
2) Use a statistics pooling layer to aggregate frame-level embeddings into a single fixed-dimensional vector by computing mean and standard deviation.
3) This pooling layer establishes long-term temporal context and provides robust embeddings.

Dataset Splitting and Model Training

1) Split the dataset into train and test sets, And train the model to extract embeddings.

Classification

1) Use methods like PDLA (Probabilistic Discriminative Linear Analysis) or categorical cross-entropy for optimization.

Evaluation Metrics

1) Measure performance using F1 Score and Accuracy. And check it with test data