School of Information Technology and Engineering at the ADA University

School of Engineering and Applied Science at the George Washington University

# AUTOMATIC SPEECH RECOGNITION FOR NUMERIC DATA IN AZERBAIJANI LANGUAGE

A Thesis
Presented to the Graduate Program of Computer Science and Data Analytics
of the School of Information Technology and Engineering
ADA University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science and Data Analytics
ADA University

By
Ulvi Aslanli

April, 2023

THESIS ACCEPTANCE

This Thesis by: Ulvi Aslanli
Entitled: *Automatic Speech Recognition for numeric data in Azerbaijani Language*

has been approved as meeting the requirement for the Degree of Master of Science in Computer Science and Data Analytics of the School of Information Technology and Engineering, ADA University.

Approved:

| | |
|---|---|
| (Adviser) | (Date) |
| (Program Director) | (Date) |
| (Dean) | (Date) |

ACADEMIC INTEGRITY STATEMENT

"I affirm that this is my own work, I attributed where I used the work of others, I did not facilitate academic dishonesty for myself or others, and I used only authorized resources for my Thesis, per the ADA University Academic Integrity requirements. If I failed to comply with this statement, I understand consequences will follow my actions. Consequences may range from failing the course to expulsion from the program/university and may include a transcript notation."

| (Full Name) | (Signature) | (Date: DD.MM.YY) |
|---|---|---|

# ABSTRACT

Automatic Speech Recognition (ASR) technology is essential in a variety of applications, such as voice search, virtual assistants, transcription services, and subtitling for people with hearing impairments. Despite its numerous applications, developing ASR systems for low-resource languages like Azerbaijani presents significant challenges due to the scarcity of available data, linguistic variations, and the unique phonetic properties of the language. This thesis specifically addresses the development of an ASR system for recognizing numeric data in Azerbaijani, a Turkic language spoken by approximately 50 million people worldwide. Numeric data recognition has critical practical applications in industries such as finance and transportation, where accurate and reliable recognition of numbers is essential.

One of the primary challenges in developing an ASR system for numeric data is the inherent lack of context available to help disambiguate similar-sounding numbers. Unlike general speech recognition, numeric data often appears in isolation or with limited accompanying information, making it more difficult to accurately recognize spoken numbers. This challenge is further exacerbated in low-resource languages like Azerbaijani.

The objective of this master's thesis is to develop an ASR system for numeric data in Azerbaijani by exploring various techniques and methodologies. We investigate the phonetic and linguistic properties of Azerbaijani relevant to numeric data recognition and analyze the existing resources for developing an ASR system. The study proposes a framework for ASR system development, experimenting with different feature extraction and modeling techniques, and evaluating the performance of the system using appropriate metrics.

In this research, we developed an ASR system for the Azerbaijani language using the Kaldi toolkit. The ASR model was trained using the classic Hidden Markov Model - Gaussian Mixture Model (HMM-GMM) architecture, employing both monophone and triphone models along with various feature extraction techniques such as Mel-Frequency Cepstral Coefficients (MFCC), Linear Predictive Coding (LPC), and Cepstral Mean and Variance Normalization (CMVN). The experimental results showed that the triphone models generally outperformed monophone models, and the combination of MFCC, LPC, and CMVN features provided the best performance among the tested feature extraction techniques. While performance varied across different datasets, our ASR system demonstrated promising potential for further improvements and adaptation to specific challenges presented by each dataset.

This thesis contributes to the development of ASR technology for low-resource languages, specifically Azerbaijani, in the domain of numeric data recognition. The results of this research have practical implications for industries that rely on accurate and reliable recognition of numeric data, such as financial services and transportation. As the dataset and ASR system improve, we anticipate that the impact on various applications, including voice assistants, transcription services, and speech analytics in Azerbaijani, will be significant. This study lays the foundation for further research and development of ASR systems for the Azerbaijani language, paving the way for improved and more robust ASR solutions.

# Contents

# List of Tables

# List of Figures

# LIST OF KEYWORDS

- ASR: Automatic Speech Recognition, the process of transcribing spoken language into text using computer algorithms.
- N-gram: A sequence of N items, typically words or letters, used in language modeling for statistical language modeling.
- Language Modeling: The task of estimating the probability distribution of sequences of words or other linguistic units in a language.
- Kaldi: A popular open-source toolkit for speech recognition, capable of performing speech recognition, speaker diarization, and other related tasks.
- SRILM: The SRI Language Modeling Toolkit, a suite of tools for building and evaluating statistical language models.
- GMM: Gaussian Mixture Model, a probabilistic model used to represent the distribution of a set of continuous variables as a mixture of Gaussian distributions.
- HMM: Hidden Markov Model, a statistical model used to describe sequences of observable events in terms of underlying unobservable states.
- Phoneme: The smallest unit of sound in a language, used as the building blocks for constructing words.
- Acoustic Model: The component of an ASR system responsible for converting audio input into a sequence of phonemes or other linguistic units.
- Language Model Smoothing: The process of adjusting the probability distribution of a language model to account for rare or unseen events in the training data.
- Beam Search: An algorithm used to search for the most likely sequence of words given a language model and acoustic model, by considering only the most likely candidate sequences at each step.
- Word Error Rate (WER): A common metric used to evaluate the performance of an ASR system, defined as the percentage of words in the output that differ from the correct transcription.
- Perplexity: A measure of how well a language model predicts a given sequence of words, calculated as the inverse probability of the test set normalized by the number of words.

# 1 Introduction

Automatic Speech Recognition (ASR) systems have been an active area of research for many years, with substantial progress made in recent times due to advancements in machine learning and deep learning techniques. ASR systems have found numerous applications in fields such as transcription services, voice assistants, and customer support systems. However, the development of ASR systems for less-studied languages, such as Azerbaijani, remains limited. This study focuses on developing a robust ASR system for numeric data in the Azerbaijani language, which can have significant implications for various applications in both public and private sectors.

## 1.1 Definition of the Problem

The problem addressed in this study is the development of an ASR system capable of accurately recognizing and transcribing numeric data in the Azerbaijani language. Numeric data can be found in various contexts, such as phone numbers, bank account numbers, dates, and monetary amounts. Developing an ASR system that can accurately recognize and transcribe such data can be challenging due to factors such as speaker variability, background noise, and the complexity of the Azerbaijani language.

## 1.2 Objective of the Study

The primary objective of this study is to design, develop, and evaluate an ASR system for numeric data in the Azerbaijani language. This includes:

1. Collecting and curating a comprehensive dataset of numeric data in Azerbaijani.
2. Developing appropriate feature extraction methods for the ASR system.
3. Evaluating various acoustic and language models to determine the optimal configuration for the ASR system.
4. Assessing the performance of the developed ASR system on a diverse set of test data.
5. Identifying areas for future work and improvement in the ASR system.

## 1.3 Significance of the Problem

The development of an ASR system for numeric data in the Azerbaijani language is significant for several reasons:

1. It addresses the gap in research and applications related to ASR for less-studied languages like Azerbaijani.
2. An accurate and robust ASR system for numeric data can be utilized in various applications, such as voice-based customer service systems, banking, telecommunication, and transportation services.
3. The insights gained from this research can contribute to the broader understanding of ASR system development for other under-researched languages and domains.

## 1.4 Assumptions and Limitations

The study assumes that the carefully curated dataset is representative of real-world audio samples containing numeric data in Azerbaijani. The study also assumes that the performance of the ASR system accurately reflects its ability to transcribe numeric data in Azerbaijani.

One significant limitation of this study is that it only focuses on numeric data and does not address the recognition of other types of speech. Another limitation is that the study only uses a limited number of n-gram language models and evaluation metrics, which may not fully capture the ASR system's performance. Finally, the study's findings may not be generalizable to other less commonly spoken languages.

# 2 Literature Review and Related Work

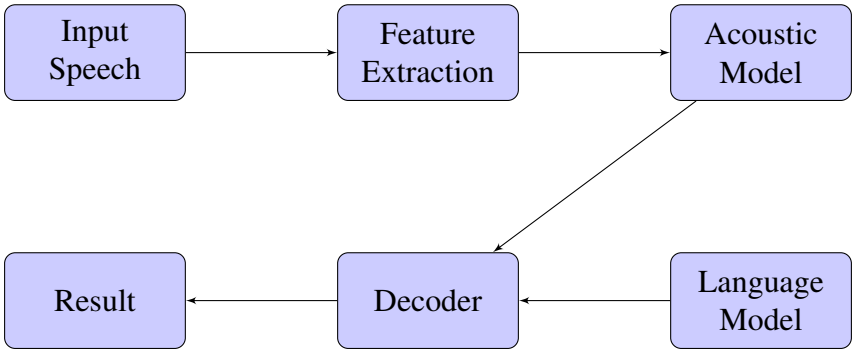


**Figure 1: Automatic Speech Recognition System Diagram.** The ASR system consists of several interconnected components. The *Input Speech* is first processed by the *Feature Extraction* module, which extracts relevant acoustic features. These features are then passed to the *Acoustic Model* to generate a probability distribution over phonetic units. The *Language Model* provides a probability distribution over word sequences, which is combined with the output from the Acoustic Model. The *Decoder* utilizes both the Acoustic Model and Language Model probabilities to generate the most likely sequence of words. Finally, the *Result* is the recognized text produced by the system.

An Automatic Speech Recognition (ASR) system is designed to convert spoken language into written text by analyzing and decoding the acoustic features of speech signals. The general architecture of an ASR system can be broadly divided into four components: Feature Extraction, Acoustic Model, Language Model, and Decoder. Here, we provide an overview of these components:

1. Feature Extraction: The feature extraction component is responsible for converting the raw speech waveform into a compact and meaningful representation that captures the relevant information in the signal. The primary goal is to extract features that are invariant to irrelevant factors (e.g., speaker, channel, and environment) and sensitive to the linguistic content of the speech. Commonly used feature extraction techniques include Mel-Frequency Cepstral Coefficients (MFCC), Linear Predictive Coding (LPC), and Perceptual Linear Prediction (PLP). These features are typically computed on short-time frames (20-30 ms) of the speech signal with some overlap between consecutive frames.

2. Acoustic Model: The acoustic model component aims to capture the relationship between the acoustic features of speech signals and the corresponding linguistic units, such as phonemes

or graphemes. It provides a probabilistic model of how the speech features are generated by the linguistic units, which is then used by the decoder to find the most likely word sequence. Acoustic models can be based on various techniques, including Hidden Markov Models (HMMs), Gaussian Mixture Models (GMMs), and Deep Neural Networks (DNNs). The acoustic model is trained on a large dataset of labeled speech data, where the speech signals are paired with the corresponding transcriptions.

3. Language Model: The language model component captures the syntactic, semantic, and pragmatic regularities of the target language. It provides a probability distribution over sequences of words or subword units, which is used by the decoder to constrain and guide the search for the most likely word sequence. Language models can be based on various techniques, such as n-gram models, which estimate the probability of a word given its n-1 preceding words, or more recent approaches like neural network-based models, including Recurrent Neural Networks (RNNs), Transformers, and GPT-based models. The language model is trained on a large dataset of text data, which may be domain-specific or general.

4. Decoder: The decoder component is responsible for finding the most likely word sequence given the observed acoustic features and the underlying acoustic and language models. It searches through the vast space of possible word sequences to identify the sequence with the highest probability or score. Decoding algorithms employed in ASR systems include the Viterbi algorithm, beam search, A* search, stack decoding, and lattice generation and rescoring. The choice of the decoding algorithm depends on the trade-off between search efficiency and recognition accuracy.

In summary, an ASR system comprises four main components: Feature Extraction, Acoustic Model, Language Model, and Decoder. These components work together to convert the raw speech waveform into a written transcription by analyzing the acoustic features, leveraging the knowledge of the target language, and efficiently searching through the space of possible word sequences.

$$y[n] = x[n] - \alpha x[n-1] \tag{1}$$

Framing and Windowing: The input signal is divided into overlapping frames, and a window function is applied to each frame.

$$w[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \tag{2}$$

LPC Coefficients Estimation: The LPC coefficients are computed by minimizing the mean square prediction error. The autocorrelation method is commonly used for this purpose. The autocorrelation of the windowed signal is given by:

$$R[k] = \sum_{n=k}^{N-1} x_w[n]x_w[n-k] \tag{3}$$

The LPC coefficients can be estimated by solving the following linear system known as the Yule-Walker equations:

$$\sum_{k=1}^{p} a_k R[m-k] = R[m], ; m = 1, 2, ..., p \tag{4}$$

Where $a_k$ are the LPC coefficients, $R[m]$ is the autocorrelation function, and $p$ is the order of the LPC model.

Spectral Envelope Estimation: The estimated LPC coefficients can be used to form the LPC filter, which models the spectral envelope of the speech signal. The transfer function of the LPC filter is given by:

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{5}$$

Where $A(z)$ is the LPC filter polynomial, and $H(z)$ is the LPC filter transfer function.

LPC Cepstral Coefficients: To obtain a compact and efficient representation of the spectral envelope, the LPC coefficients can be transformed into cepstral coefficients using the following recursive relation:

$$c_i = \ a_i + \sum_{k=1}^{i-1} \frac{i-k}{i} c_k a_{i-k}, \quad 1 \leq i \leq p \tag{6}$$

Where $c_i$ are the LPC cepstral coefficients, and $a_i$ are the LPC coefficients.
LPC-based features have been extensively used in speech coding (Schroeder & Atal, 1985), speech synthesis (Klatt, 1980), and speech recognition (Rabiner & Juang, 1993).

### 2.0.1 PLP

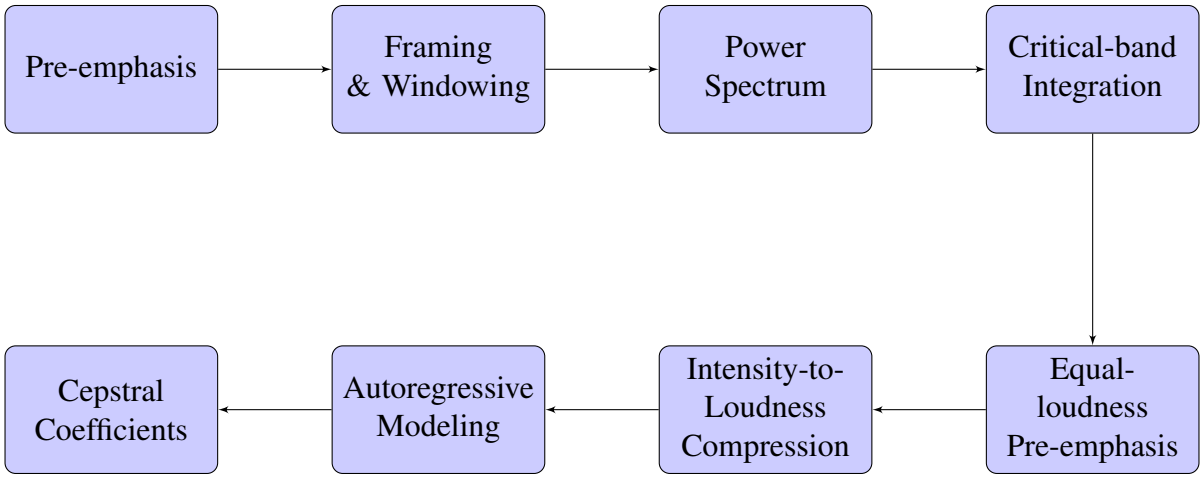


**Figure 2: PLP Feature Extraction Process**
The diagram illustrates the steps involved in extracting *PLP* features from a *speech signal*. The process starts with *pre-emphasis*, followed by *framing* and *windowing* the signal. The *power spectrum* is computed, and the signal undergoes *critical-band integration* based on the Bark scale. An *equal-loudness pre-emphasis* is applied, followed by *intensity-to-loudness compression*. *Autoregressive modeling* is performed to fit an all-pole model to the loudness spectrum, and the resulting PLP coefficients are transformed into *cepstral coefficients*.

Perceptual Linear Prediction (PLP) is a feature extraction technique that is based on the human auditory system and aims to mimic the human perception of sound (Hermansky, 1990). PLP features have been widely used in various speech and audio processing tasks, including speech recognition, speaker identification, and audio classification.

1. Pre-emphasis: Similar to MFCC and LPC, the pre-emphasis step is applied to enhance the high-frequency components and balance the frequency spectrum.

$$y[n] = x[n] - \alpha x[n-1] \tag{7}$$

2. Framing and Windowing: The input signal is divided into overlapping frames, and a window function is applied to each frame.

$$w[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \tag{8}$$

3. Power Spectrum: The power spectrum of the windowed frame is computed using the Discrete Fourier Transform (DFT).

$$X[k] = \sum_{n=0}^{N-1} x[n]w[n]e^{-j\frac{2\pi kn}{N}} \tag{9}$$

$$P[k] = \frac{1}{N}|X[k]|^2 \tag{10}$$

4. Critical-band Integration: The power spectrum is integrated into critical bands, which are based on the Bark scale that approximates the frequency resolution of the human auditory system. The Bark scale is given by:

$$z(f) = 13\arctan(0.00076f) + 3.5\arctan\left(\frac{f^2}{7500^2}\right) \tag{11}$$

Where $z(f)$ is the Bark scale value corresponding to frequency $f$.

5. Equal-loudness Pre-emphasis: The power in each critical band is pre-emphasized to account for the non-uniform sensitivity of human hearing to different frequencies. The equal-loudness contour is approximated by:

$$L(z) = -(200 - z)^{\delta} \tag{12}$$

Where $L(z)$ is the equal-loudness pre-emphasis, and $\delta$ is a parameter typically set to 0.35.

6. Intensity-to-loudness Compression: The pre-emphasized critical-band power is transformed into loudness perception by applying a power-law compression.

$$\Psi(z) = \left(\frac{P(z)}{P_0}\right)^{\epsilon} \tag{13}$$

Where $\Psi(z)$ is the loudness perception, $P(z)$ is the pre-emphasized critical-band power, $P_0$ is a reference power, and $\epsilon$ is a parameter typically set to 0.33.

7. Autoregressive Modeling: An all-pole model is fitted to the loudness spectrum using linear prediction, similar to the LPC method. The coefficients of the all-pole model are the PLP coefficients.

8. Cepstral Coefficients: The PLP coefficients can be transformed into cepstral coefficients, often referred to as PLP cepstral coefficients, using a technique similar to the one used in LPC cepstral analysis.

PLP features have been shown to be robust and effective in various speech and audio processing applications, as they capture the perceptual characteristics of the human auditory system (Hermansky & Morgan, 1994).

### 2.0.2 CMVN

Cepstral Mean and Variance Normalization (CMVN) is a widely used technique in speech and audio processing to reduce the effects of channel and environment variations, as well as speaker variability, in the extracted features. It is commonly applied to cepstral features such as MFCC, LPC cepstral coefficients, and PLP cepstral coefficients (Viikki & Laurila, 1998).

The primary goal of CMVN is to normalize the mean and variance of the cepstral coefficients across a given utterance or a set of utterances, making the features more robust against channel and environment variations. The process involves two main steps: mean normalization and variance normalization.

1. Mean normalization: The mean normalization is performed by subtracting the mean of the cepstral coefficients computed over the entire utterance or set of utterances.

$$\bar{c}_i = c_i - \mu_i \tag{14}$$

   Where $\bar{c}_i$ is the mean-normalized cepstral coefficient, $c_i$ is the original cepstral coefficient, and $\mu_i$ is the mean of the $i^{th}$ cepstral coefficient computed over the utterance or set of utterances.

2. Variance normalization: The variance normalization is performed by dividing the mean-normalized cepstral coefficients by the standard deviation of the coefficients computed over the entire utterance or set of utterances.

$$\hat{c}_i = \frac{\bar{c}_i}{\sigma_i} \tag{15}$$

   Where $\hat{c}_i$ is the variance-normalized cepstral coefficient, $\bar{c}_i$ is the mean-normalized cepstral coefficient, and $\sigma_i$ is the standard deviation of the $i^{th}$ cepstral coefficient computed over the utterance or set of utterances.

CMVN has been widely adopted in speech and audio processing tasks, such as speech recognition (Viikki & Laurila, 1998), speaker identification (Furui, 1981), and audio classification (Tzanetakis & Cook, 2002), as it improves the performance of these systems by reducing the effects of channel and environment variations.

## 2.1 Acoustic Model

Speech recognition is a challenging task that involves converting the continuous acoustic waveform of a spoken utterance into a sequence of symbolic representations, such as phonemes or words. To accomplish this task, speech recognition systems use an acoustic model, which is

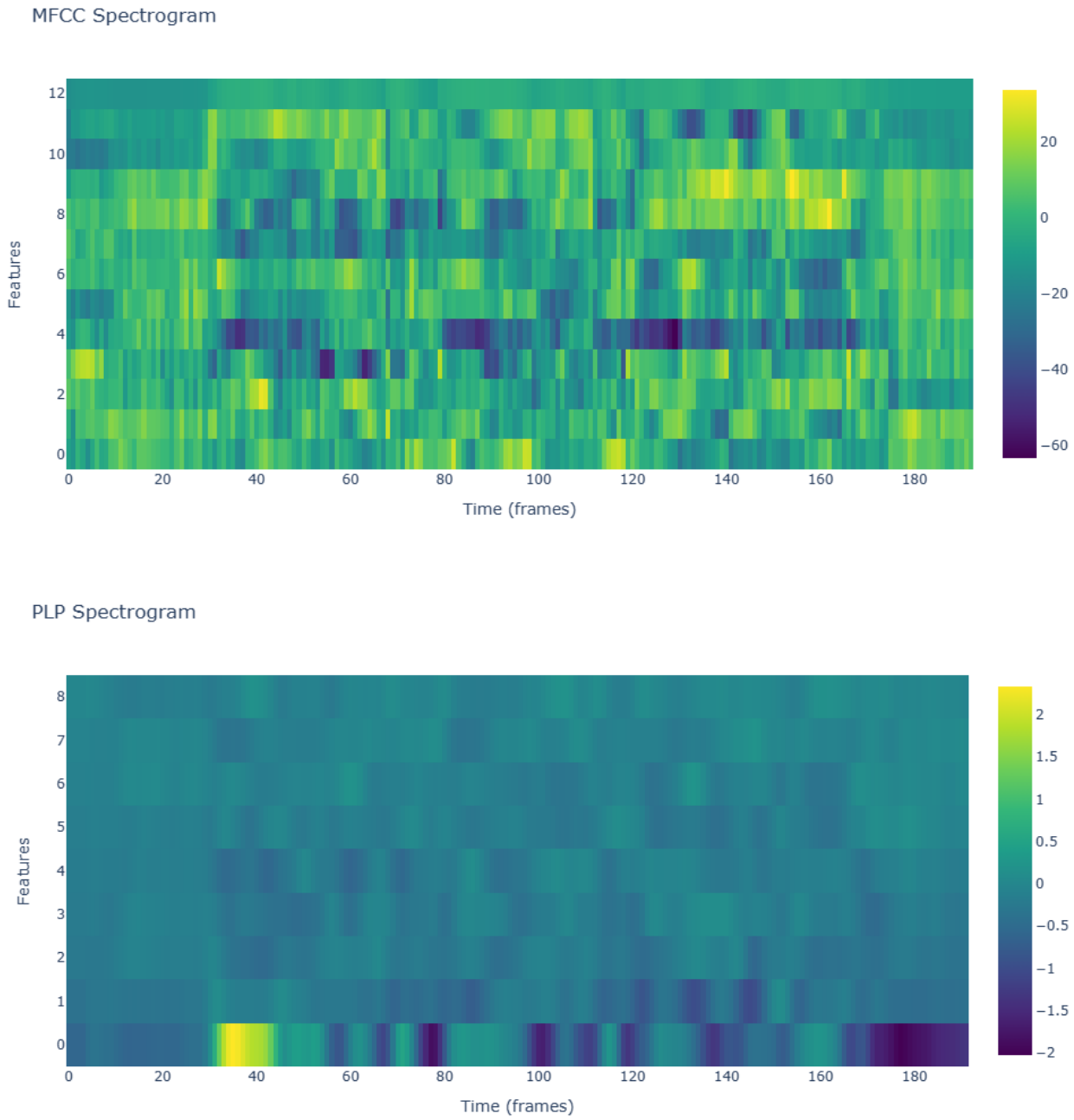


**Figure 3:** Spectrograms of an audio signal with MFCC and PLP features. Both spectrograms are computed with a window size of 40 ms and a step size of 20 ms. MFCC features are calculated with 12 coefficients, while PLP features are calculated with 8 coefficients and RASTA processing.

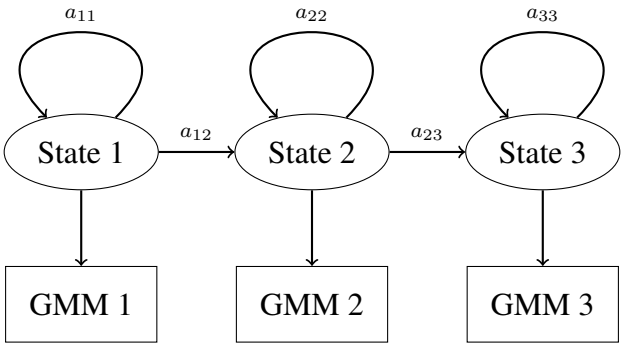


**Figure 4: Schematic Diagram of the HMM-GMM Architecture.** This figure depicts the structure of an *HMM-GMM* system, showing the *Hidden Markov Model (HMM)* states, state transition probabilities $a_{ij}$, and the associated *Gaussian Mixture Models (GMMs)* for each state. The *HMM* models the temporal dynamics of speech signals, while the *GMM* models the probability distributions of the feature vectors given the HMM states.

responsible for modeling the relationship between the acoustic features of the speech signal and the corresponding symbolic units.

The acoustic model is typically implemented as a statistical model, which is trained on a large corpus of labeled speech data. The model parameters are estimated using machine learning algorithms, such as hidden Markov models (HMMs) or deep neural networks (DNNs), which learn the statistical dependencies between the input features and the output symbols.

HMMs are a well-established approach to acoustic modeling, which have been used successfully in speech recognition systems for many years. HMM-based acoustic models represent the acoustic features of the speech signal as a sequence of probability distributions over the possible output symbols, such as phonemes or words. During decoding, the Viterbi algorithm is used to find the most likely sequence of output symbols given the observed acoustic features.

DNNs are a relatively new approach to acoustic modeling, which have shown significant improvements in speech recognition performance over the past decade. DNN-based acoustic models represent the acoustic features of the speech signal as a high-dimensional vector, which is mapped to a sequence of output symbols using a deep neural network. During decoding, the forward-backward algorithm or variants thereof are used to compute the posterior probabilities of the output symbols given the observed acoustic features.

Both HMMs and DNNs have their strengths and weaknesses, and are suited to different types of speech recognition tasks. HMMs are particularly effective for modeling speech signals with limited training data, and are less sensitive to noise and speaker variability than DNNs. DNNs, on the other hand, are more effective at modeling complex, high-dimensional acoustic features, and have been shown to outperform HMMs on large-scale speech recognition tasks.

In summary, the choice of acoustic modeling approach depends on the specific requirements of the speech recognition task, as well as the availability and quality of the training data.

### 2.1.1 HMM-GMM

The Hidden Markov Model-Gaussian Mixture Model (HMM-GMM) architecture is a widely used approach in speech and audio processing tasks, such as speech recognition, speaker identification, and audio classification. The HMM-GMM architecture combines the strengths of Hidden Markov Models (HMMs) for modeling the temporal dynamics of speech signals and Gaussian Mixture Models

(GMMs) for modeling the probability distributions of the feature vectors (Rabiner & Juang, 1993).

Hidden Markov Models (HMMs): HMMs are a class of statistical models that represent a system with a finite number of hidden states, each of which is associated with an observable output probability distribution. HMMs are particularly suitable for modeling time-varying processes, such as speech signals, due to their ability to capture the underlying temporal structure. An HMM is characterized by the following components (Rabiner, 1989): State transition probabilities: $a_{ij} = P(q_t = j|q_{t-1} = i)$, where $q_t$ is the state at time $t$. Output probabilities: $b_j(\mathbf{o}_t) = P(\mathbf{o}_t|q_t = j)$, where $\mathbf{o}_t$ is the observed feature vector at time $t$. Initial state probabilities: $\pi_i = P(q_1 = i)$. Gaussian Mixture Models (GMMs): GMMs are a class of parametric probability density functions that represent a mixture of several Gaussian distributions. GMMs are used to model the output probabilities of the HMM states. Each state in the HMM is associated with a GMM, which models the probability distribution of the feature vectors given the state. A GMM is defined by the following parameters (Reynolds & Rose, 1995): Mixture weights: $w_k$, where $k = 1, 2, ..., K$, and $K$ is the number of Gaussian components in the mixture. Gaussian component means: $\boldsymbol{\mu}_k$. Gaussian component covariances: $\boldsymbol{\Sigma}_k$. The output probability of a feature vector $\mathbf{o}_t$ given an HMM state $j$ is computed as:

$$b_j(\mathbf{o}t) = \sum k = 1^K w_k \mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{16}$$

Where $\mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the Gaussian probability density function with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$.

In the HMM-GMM architecture, the HMMs are used to model the temporal dynamics of speech signals, while the GMMs are used to model the probability distributions of the feature vectors given the HMM states. The HMM-GMM system is trained using a two-step process: HMM parameter initialization and iterative parameter estimation using the Expectation-Maximization (EM) algorithm (Dempster et al., 1977).

The HMM-GMM architecture has been widely adopted in speech and audio processing tasks due to its effectiveness in modeling the complex and time-varying nature of speech signals. However, with the advent of deep learning techniques, more recent approaches such as Deep Neural Networks-HMM (DNN-HMM) and Long Short-Term Memory (LSTM)-HMM hybrid systems have shown superior performance in various tasks, including speech recognition, speaker identification, and audio classification (Hinton et al., 2012; Graves et al., 2013).

Despite the shift towards deep learning techniques, the HMM-GMM architecture remains an important foundation and continues to be used as a benchmark for comparison. It also serves as a basis for understanding more advanced hybrid models that incorporate deep learning components.

Some of the more recent architectures that build on the HMM-GMM framework are:

1. Deep Neural Network-HMM (DNN-HMM): In the DNN-HMM architecture, a deep neural network is used to model the output probabilities of the HMM states instead of GMMs (Hinton et al., 2012). The DNN is trained to predict the HMM state given the input features, providing a more powerful and discriminative model of the output probabilities.

2. Long Short-Term Memory (LSTM)-HMM: LSTMs are a type of recurrent neural network (RNN) specifically designed to model long-range temporal dependencies in sequences (Hochreiter & Schmidhuber, 1997). In an LSTM-HMM hybrid system, the LSTM is used to model the output probabilities of the HMM states, providing a better representation of the temporal structure in the speech signal (Graves et al., 2013).

3. Convolutional Neural Network-HMM (CNN-HMM): CNNs are a type of neural network specifically designed for processing grid-like data, such as images or spectrograms (LeCun et al., 1998). In a CNN-HMM hybrid system, the CNN is used to model the output probabilities of the HMM states, taking advantage of the spatial structure in the input features (Sainath et al., 2015).

These more recent architectures have demonstrated superior performance compared to the traditional HMM-GMM systems, thanks to their ability to capture more complex and discriminative patterns in the input features and better model the temporal structure of speech signals.

In the context of HMM-GMM-based speech recognition systems, the modeling of speech units plays a crucial role in the system's performance. Two commonly used approaches for modeling speech units are monophone and triphone modeling, which differ in their complexity and the way they capture context-dependent variations in speech.

**Monophone Training** Monophone models represent the most basic units of speech, known as phonemes, without taking into account any context. Each phoneme is modeled as an HMM with a fixed number of states, and each state is associated with a GMM to model the probability distribution of the feature vectors. During monophone training, the HMM-GMM parameters are estimated using a set of training data that is annotated with phoneme labels (Rabiner & Juang, 1993).

The primary advantage of monophone modeling is its simplicity, which results in a lower computational complexity and fewer training data requirements. However, monophone models do not account for the coarticulation effects that occur when different phonemes are pronounced in various contexts. As a result, monophone-based systems tend to have limited performance, especially when dealing with continuous, natural speech.

**Triphone Training** Triphone models address the limitations of monophone models by taking into account the context-dependent variations in speech. A triphone is a phoneme conditioned on the preceding and following phonemes, which captures the coarticulation effects and the resulting variations in the pronunciation of a given phoneme (Woodland et al., 1994).

In triphone training, each triphone is modeled as an HMM with a fixed number of states, and each state is associated with a GMM to model the probability distribution of the feature vectors. The HMM-GMM parameters are estimated using a set of training data that is annotated with phoneme labels and their context. Due to the large number of possible triphone combinations, decision tree-based state clustering techniques are often used to share the HMM-GMM parameters among similar triphones, reducing the number of parameters that need to be estimated and alleviating the data sparsity problem (Young et al., 1994).

Triphone-based systems offer improved performance compared to monophone-based systems, as they capture context-dependent variations in speech more effectively. However, triphone modeling comes with increased complexity and requires more training data to estimate the HMM-GMM parameters reliably.

In summary, monophone and triphone modeling represent two different approaches to modeling speech units in HMM-GMM-based speech recognition systems. While monophone models offer simplicity and lower computational complexity, triphone models provide better performance by accounting for context-dependent variations in speech.

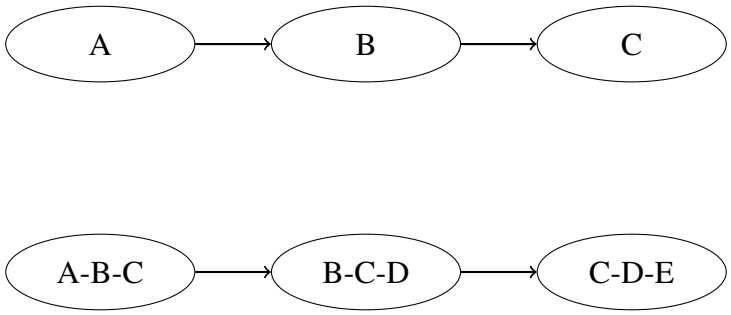


**Figure 5: Illustration of Monophone and Triphone Modeling.** This figure demonstrates the difference between *monophone* and *triphone* models. Monophone models represent phonemes without context (e.g., A, B, C), while triphone models represent phonemes with context, conditioned on the preceding and following phonemes (e.g., A-B-C, B-C-D, C-D-E).

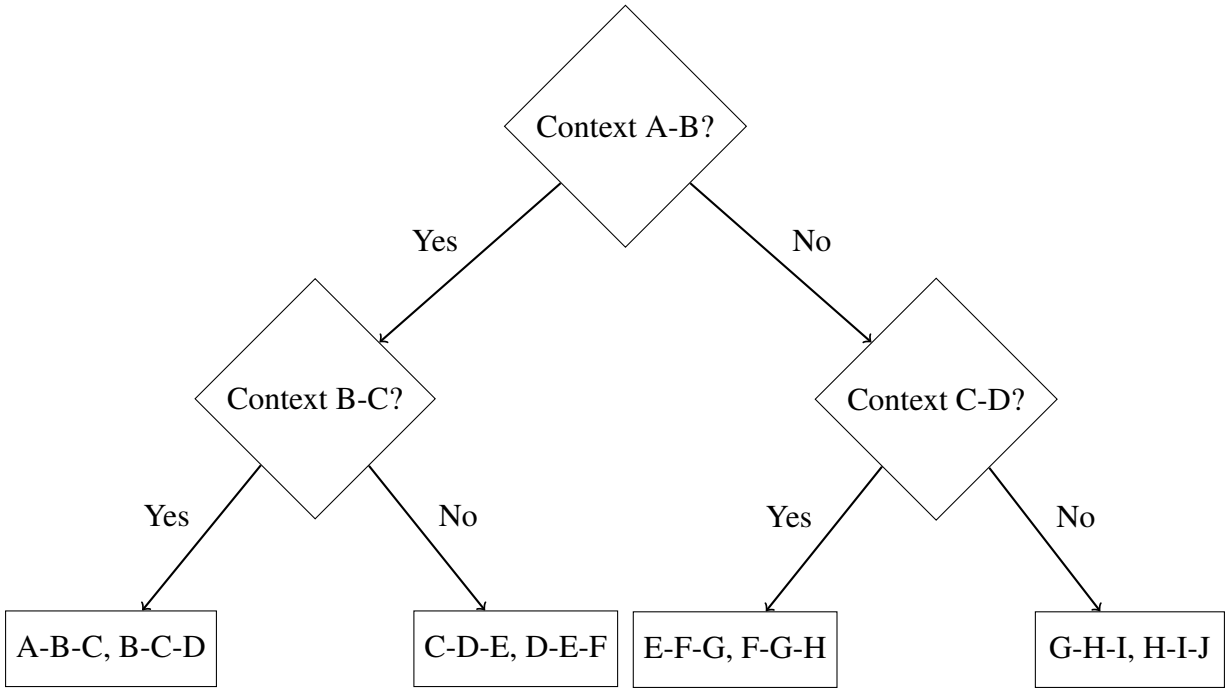


**Figure 6: Decision Tree-Based State Clustering Example.** This figure illustrates an example of *decision tree-based state clustering* used in triphone training. The decision tree is used to share the HMM-GMM parameters among similar triphones, reducing the number of parameters that need to be estimated. Each internal node of the tree represents a question about the phonetic context, while the leaf nodes represent clusters of similar triphones (e.g., A-B-C, B-C-D).

### 2.1.2 DNN-HMM

Deep Neural Network-Hidden Markov Model (DNN-HMM) is a hybrid architecture that combines the strengths of Deep Neural Networks (DNNs) and Hidden Markov Models (HMMs) for acoustic modeling in ASR systems. DNN-HMM has become a popular choice in recent years due to its ability to model complex patterns and capture higher-level features in the speech signal, resulting in improved recognition performance compared to the traditional Gaussian Mixture Model-Hidden Markov Model (GMM-HMM) systems.

The DNN-HMM hybrid architecture combines the strengths of DNNs and HMMs for acoustic modeling. In this architecture, the DNN is used to estimate the posterior probabilities of the HMM states given the acoustic observations. These posterior probabilities are then used as the observation likelihoods in the HMM framework.

Given an acoustic observation $o_t$ at time $t$, the DNN computes the posterior probability of each HMM state $s$ as follows:

$$P(s|o_t) = \frac{P(o_t|s)P(s)}{P(o_t)} \tag{17}$$

where $P(o_t|s)$ is the likelihood of observing $o_t$ given state $s$, $P(s)$ is the prior probability of state $s$, and $P(o_t)$ is the probability of observing $o_t$. Since $P(o_t)$ is constant for all states, we can ignore it for the purpose of maximizing the posterior probabilities.

The output layer of the DNN has as many nodes as the number of HMM states in the system. Each node in the output layer computes the posterior probability of the corresponding HMM state given the input observation using a softmax activation function:

$$P(s_i|o_t) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{18}$$

where $z_i$ is the output of the $i$-th node before applying the softmax function, and $N$ is the number of HMM states.

In the training phase, the DNN is trained to minimize the cross-entropy loss between the predicted posterior probabilities and the true state alignments obtained from a GMM-HMM system or a forced alignment procedure. The cross-entropy loss is defined as:

$$L = -\sum_{t=1}^{T}\sum_{i=1}^{N} q_{ti} \log P(s_i|o_t) \tag{19}$$

where $T$ is the number of time steps in the input sequence, $q_{ti}$ is the true state alignment, and $P(s_i|o_t)$ is the predicted posterior probability of state $i$ at time $t$. The DNN is trained using backpropagation and gradient descent optimization algorithms.

In the decoding phase, the DNN-HMM system searches for the most likely state sequence and the corresponding word sequence given the input acoustic observations. This is typically achieved using the Viterbi algorithm or other search algorithms such as token-passing, A*, or beam search. The DNN provides the observation likelihoods $P(s|o_t)$, which are then combined with the state transition probabilities $a_{ij}$ and language model probabilities to find the most likely word sequence.

In conclusion, the DNN-HMM hybrid architecture leverages the power of deep neural networks to model complex patterns in speech signals and the flexibility of hidden Markov models to represent

### 2.1.3 Evaluation metrics for Acoustic Models

Acoustic models are a fundamental component of speech recognition systems, aiming to capture the relationship between the acoustic features of speech signals and the corresponding linguistic units, such as phonemes or graphemes. Several evaluation metrics are used to assess the performance of acoustic models, including:

Phone Error Rate (PER): Phone error rate measures the number of edit operations (insertions, deletions, or substitutions) required to align the predicted phone sequence with the true phone sequence, normalized by the total number of phones in the true sequence. Lower PER values indicate better performance. Frame Error Rate (FER): Frame error rate quantifies the percentage of frames in which the acoustic model makes an incorrect prediction. In other words, it compares the predicted and true phonemes at each frame of the speech signal and calculates the proportion of mismatches. Lower FER values indicate better performance. Accuracy: Accuracy is the proportion of correctly predicted phonemes or graphemes relative to the total number of predictions. Higher accuracy values indicate better performance. Confusion Matrix: A confusion matrix provides a detailed view of the performance of an acoustic model by comparing the predicted phonemes or graphemes with the true labels. Each row of the matrix corresponds to the true label, and each column represents the predicted label. The confusion matrix can be used to identify specific classes (e.g., phonemes) that are commonly misclassified by the acoustic model, providing insights into potential areas for improvement. Log-Likelihood: Log-likelihood is the logarithm of the probability of the observed acoustic features given the acoustic model. Higher log-likelihood values indicate better performance, as they imply that the model assigns higher probabilities to the observed acoustic features. Word Error Rate (WER): Although word error rate is primarily used for evaluating the overall performance of a speech recognition system, it can also be used to assess the impact of improvements in the acoustic model on the system's performance. Lower WER values indicate better performance. It is crucial to consider the specific application and task when choosing an evaluation metric for an acoustic model, as some metrics may be more informative or relevant than others depending on the context.

## 2.2 Lanugage Models

The field of Natural Language Processing (NLP) has witnessed significant advancements with the development of Language Models (LMs). LMs are probabilistic models that aim to predict the next word or sequence of words in a given context, capturing the structure, syntax, and semantics of a language. They play a crucial role in various NLP tasks, such as speech recognition, machine translation, and text generation (Goodfellow et al., 2016).

A language model can be formally defined as estimating the probability distribution of a sequence of words $w_1, w_2, \ldots, w_T$:

$$P(w_1, w_2, \ldots, w_T) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1, w_2) \cdots P(w_T|w_1, \ldots, w_{T-1}) \tag{20}$$

Various approaches have been proposed to model the probability distribution $P(w_t|w_1, \ldots, w_{t-1})$ in LMs. In the following, we provide a brief overview of some of the most popular language modeling techniques.

### 2.2.1 N-gram Models

N-gram models are a class of LMs that leverage the Markov assumption, approximating the probability distribution by considering only the $(N-1)$ most recent words as context (Jurafsky & Martin, 2009). An N-gram model can be represented as:

$$P(w_t|w_1, \ldots, w_{t-1}) \approx P(w_t|w_{t-N+1}, \ldots, w_{t-1}) \tag{21}$$

N-gram models are simple, computationally efficient, and easy to train, but they suffer from data sparsity issues and are limited in their ability to capture long-range dependencies.

### 2.2.2 Neural Language Models

Neural Language Models (NLMs) leverage neural networks to model the probability distribution $P(w_t|w_1, \ldots, w_{t-1})$. Different neural network architectures have been proposed for this purpose, including Feed-Forward Neural Networks (Bengio et al., 2003), Recurrent Neural Networks (Mikolov et al., 2010), and more recently, Transformer-based architectures (Vaswani et al., 2017).

NLMs have shown superior performance compared to traditional N-gram models, as they can capture longer-range dependencies and alleviate data sparsity issues through distributed representations of words.

### 2.2.3 Smoothing techniques for Language models

Smoothing techniques play a crucial role in language modeling, especially for n-gram models. Due to the vast number of possible word combinations, many n-grams may not be present in the training data, leading to zero probability estimates for these unseen n-grams. Smoothing techniques address this issue by redistributing probability mass from observed n-grams to unseen n-grams, providing more accurate and reliable probability estimates. Some popular smoothing techniques for language models are:

Additive Smoothing (Laplace Smoothing): Additive smoothing is a simple technique in which a small constant is added to the count of each n-gram, typically a value between 0 and 1 (e.g., 1 for Laplace smoothing). The probability estimates are then computed as:

$$P(w_t|w_{t-1}) = \frac{C(w_{t-1}, w_t) + k}{C(w_{t-1}) + kV} \tag{22}$$

Where $C(w_{t-1}, w_t)$ is the count of the bigram $(w_{t-1}, w_t)$, $C(w_{t-1})$ is the count of the unigram $w_{t-1}$, $k$ is the smoothing constant, and $V$ is the vocabulary size.

Good-Turing Smoothing: Good-Turing smoothing is a technique that estimates the probability of unseen n-grams based on the observed frequency of n-grams with similar counts. The adjusted count $C^*$ is calculated as:

$$C^* = \frac{(C+1)N_{C+1}}{N_C} \tag{23}$$

Where $C$ is the original count, $N_C$ is the number of n-grams with count $C$, and $N_{C+1}$ is the number of n-grams with count $C+1$. The probability estimates are then computed using the adjusted counts.

Kneser-Ney Smoothing: Kneser-Ney smoothing is a widely-used technique that incorporates both absolute discounting and a sophisticated method for estimating the probability of unseen n-grams. The probability estimates are calculated as:

$$P(w_t|w_{t-1}) = \frac{\max(C(w_{t-1}, w_t) - d, 0)}{C(w_{t-1})} + \lambda(w_{t-1})P_{KN}(w_t) \tag{24}$$

Where $C(w_{t-1}, w_t)$ is the count of the bigram $(w_{t-1}, w_t)$, $C(w_{t-1})$ is the count of the unigram $w_{t-1}$, $d$ is the discounting factor, $\lambda(w_{t-1})$ is a normalization constant, and $P_{KN}(w_t)$ is the Kneser-Ney estimate of the unigram probability.

Backoff and Interpolation: Both backoff and interpolation techniques combine the probability estimates of n-grams of different orders, such as unigrams, bigrams, and trigrams. In backoff, lower-order n-grams are used only when higher-order n-grams are not observed in the training data. In interpolation, the probability estimates of different n-gram orders are combined using weights that sum to 1. These smoothing techniques help improve the performance of language models by providing more accurate and reliable probability estimates for unseen n-grams. While some techniques, such as Kneser-Ney smoothing, have been shown to be more effective in certain scenarios, the choice of the best smoothing technique depends on the specific problem and data at hand.

It is important to note that neural language models, which have gained popularity in recent years, inherently address the data sparsity issue through the use of distributed representations (word embeddings) and continuous-valued parameters. These models can capture long-range dependencies and generalize better to unseen word combinations, reducing the need for explicit smoothing techniques.

Nevertheless, understanding and applying smoothing techniques remains essential for tasks where traditional n-gram models are still employed, or when computational resources are limited and more complex models are infeasible.

In summary, smoothing techniques are an important aspect of language modeling, particularly for n-gram models. By redistributing probability mass from observed to unseen n-grams, these techniques enable more accurate and reliable probability estimates, improving the overall performance of the language models.

### 2.2.4 Evaluation Metrics for Language Models

Evaluation metrics are essential for assessing the performance of language models and comparing different modeling approaches. These metrics quantify the quality of the generated probabilities or predictions and provide insights into the strengths and weaknesses of a language model. Some commonly used evaluation metrics for language models are:

Perplexity: Perplexity is a widely-used evaluation metric that measures the average uncertainty of a language model in predicting the next word in a sequence. A lower perplexity score indicates a better model, as it implies that the model is less "surprised" by the test data. Mathematically, perplexity for a given test set of word sequences $w_1, w_2, \ldots, w_T$ is defined as:

$$\text{Perplexity} = \exp\left(-\frac{1}{T}\sum_{t=1}^{T}\log P(w_t|w_1,\ldots,w_{t-1})\right) \tag{25}$$

Where $P(w_t|w_1,\ldots,w_{t-1})$ is the predicted probability of word $w_t$ given its context.

Cross-Entropy: Cross-entropy is a measure of the dissimilarity between the true probability distribution of a sequence of words and the predicted probability distribution from the language model. Lower cross-entropy values indicate better performance, as they imply that the predicted distribution is closer to the true distribution. Cross-entropy for a given test set of word sequences $w_1, w_2, \ldots, w_T$ is defined as:

$$\text{Cross-Entropy} = -\frac{1}{T}\sum_{t=1}^{T}\log P(w_t|w_1,\ldots,w_{t-1}) \tag{26}$$

Note that perplexity and cross-entropy are related, with perplexity being the exponentiation of cross-entropy.

Log-Likelihood: Log-likelihood is the logarithm of the probability of the test data given the language model. Higher log-likelihood values indicate better performance, as they imply that the model assigns higher probabilities to the test data. Log-likelihood for a given test set of word sequences $w_1, w_2, \ldots, w_T$ is defined as:

$$\text{Log-Likelihood} = \sum_{t=1}^{T} \log P(w_t | w_1, \ldots, w_{t-1}) \tag{27}$$

Word Error Rate (WER): Word error rate is primarily used for evaluating language models in the context of speech recognition. WER measures the minimum number of edit operations (insertions, deletions, or substitutions) required to transform the predicted sequence into the true sequence, normalized by the total number of words in the true sequence. Lower WER values indicate better performance. These evaluation metrics help in understanding the performance of language models and in comparing different models or approaches. However, it is crucial to consider the specific application and task when choosing an evaluation metric, as some metrics may be more informative or relevant than others depending on the context.

## 2.3 Decoding Algorithms

Decoding algorithms for Automatic Speech Recognition (ASR) aim to find the most likely word sequence given the observed acoustic features and the underlying language and acoustic models. These algorithms search through the vast space of possible word sequences to identify the sequence with the highest probability or score. Some widely used decoding algorithms for ASR are:

1. Viterbi Algorithm: The Viterbi algorithm is a dynamic programming-based technique used for decoding in ASR systems, particularly in the context of Hidden Markov Models (HMMs). The algorithm finds the most likely state sequence (e.g., phone or HMM state sequence) that generates the observed acoustic features. The Viterbi algorithm computes the maximum probability path for each state and time step iteratively, resulting in an efficient search procedure. However, the Viterbi algorithm does not consider language model probabilities during the search and may not provide the most likely word sequence.

2. Beam Search: Beam search is a heuristic search algorithm commonly used for decoding in ASR systems. It performs a breadth-first search while maintaining a fixed-width "beam" of the most promising partial hypotheses at each time step. Beam search can incorporate both the acoustic model and language model probabilities during the search, resulting in a more accurate word sequence. The algorithm's efficiency can be controlled by adjusting the beam width, trading off search speed and recognition accuracy.

3. A Search*: A* search is an informed search algorithm that uses a heuristic function to guide the search process. In the context of ASR, the heuristic function typically estimates the cost of completing a partial hypothesis based on the language model and/or acoustic model probabilities. A* search can find the most likely word sequence while exploring fewer hypotheses compared to other search algorithms, as it prioritizes hypotheses with lower estimated costs. However, the efficiency of A* search depends on the quality of the heuristic function.

4. Stack Decoding: Stack decoding is a depth-first search algorithm that uses a stack data structure to maintain the search frontier. Hypotheses are expanded in order of their total scores, which

include both acoustic and language model probabilities. Stack decoding can efficiently find the most likely word sequence, as it prunes hypotheses with lower scores. However, the algorithm's performance is sensitive to the search order and may degrade if the search becomes trapped in a suboptimal region. Lattice Generation and Rescoring: In this approach, the ASR system generates a

5. lattice or word graph that compactly represents the space of possible word sequences. The lattice is generated using a search algorithm such as beam search, and the edges in the lattice represent alternative word hypotheses along with their corresponding acoustic and language model probabilities. Once the lattice is generated, more complex language models or additional features can be incorporated to rescore the lattice and find the most likely word sequence.

These decoding algorithms are essential components of ASR systems, as they enable the system to search through the vast space of possible word sequences efficiently and identify the most likely sequence given the observed acoustic features and underlying models.

## 2.4 Related Work

Kamil Aida-zade et al. (1) applied support vector machines (SVMs) to the acoustic model of a speech recognition system for the Azerbaijani language. The system was based on mel-frequency cepstral coefficients (MFCC) and linear predictive coding (LPC) features. The authors compared the performance of the SVM-based system with a previous system based on multilayer artificial neural networks (ANNs), and found that the SVM-based system performed better. The authors also tested different SVM kernel functions and found that the radial basis and polynomial kernels gave the best recognition results. Overall, the article suggests that SVM-based techniques can be an effective approach for improving speech recognition in the Azerbaijani language.

Ali Abbasov et al.(2) suggested the development of a speech recognition system for the Azerbaijani language, called DilmancASR. The system is based on hidden Markov models (HMMs), which are a common approach to speech recognition. The authors compare the performance of DilmancASR with a previous system based on artificial neural networks (ANNs), and find that DilmancASR performs better. The paper provides information on the data used to create the system, which included recordings of 1,500 speakers. The system was trained on 15,000 prompts, and tested on a separate set of sentences to evaluate its accuracy. The authors found that DilmancASR achieved an average recognition accuracy of 80%, with a maximum accuracy of 90%. The authors also tested the system on sentences with different lengths and speaking rates, and found that it performed well in these conditions. Overall, the paper suggests that DilmancASR is a successful and effective approach to speech recognition for the Azerbaijani language. The authors also discuss the potential applications of the system, such as dictation and transcription, as well as its limitations and future directions for research and development.

Kamil Aida-zade et al.(3) suggested a system that uses two different algorithms, Mel Frequency Cepstral Coefficients (MFCC) and Linear Predictive Coding (LPC), to analyze the features of the speech signal and identify words. MFCCs and LPCs are used to extract the spectral and temporal characteristics of speech, which can be used to differentiate between different words and phrases. The system is trained and recognized using artificial neural networks, which are trained on a large dataset of speech samples to learn the patterns and features relevant for speech recognition. The

authors suggest using a combination of MFCC and LPC-based subsystems and multiple neural networks trained from different initial points to improve the reliability and accuracy of the system.

Alakbar Valizada et al.(4) discussed the development and performance of speech recognition systems for use in emergency call centers. The authors investigate various methodologies for acoustic and language models, as well as labeling methods, to improve the accuracy of speech recognition in noisy, emotional environments. The article compares different types of Gaussian Mixture Model/Hidden Markov Model (GMM/HMM) and Deep Neural Network/Hidden Markov Model (DNN/HMM) acoustic models and language models based on extrinsic and intrinsic methods. The authors also compare their suggested data labeling approach with spelling correction to common labeling methods. Based on the results of the experiments, the authors determine that DNN/HMM for the acoustic model, trigram with Kneser-Ney discounting for the language model, and using spelling correction before training data for the labeling method are effective configurations for dialogue speech recognition in emergency call centers.

Daniel Povey et al.(5) developed Kaldi system as an open-source toolkit for speech recognition written in C++. It provides a speech recognition system based on finite-state transducers (using the freely available OpenFst), together with detailed documentation and scripts for building complete recognition systems. Kaldi supports modeling of arbitrary phonetic-context sizes, acoustic modeling with subspace Gaussian mixture models (SGMM) as well as standard Gaussian mixture models, together with all commonly used linear and affine transforms. It is released under the Apache License v2.0, which is highly nonrestrictive, making it suitable for a wide community of users.

Andrew Ng et al.(6) described a state-of-the-art speech recognition system “DeepSpeech” developed using end-to-end deep learning. The authors claim that their architecture is significantly simpler than traditional speech systems, which rely on laboriously engineered processing pipelines. They also claim that their system does not need hand-designed components to model background noise, reverberation, or speaker variation, but instead directly learns a function that is robust to such effects. The authors also describe a well-optimized RNN training system that uses multiple GPUs, as well as a set of novel data synthesis techniques that allow them to efficiently obtain a large amount of varied data for training. The authors report that their system, called ”Deep Speech,” outperforms previously published results on the widely studied Switchboard Hub5’00 corpus, achieving 16.0% error on the full test set. They also claim that Deep Speech handles challenging noisy environments better than widely used, state-of-the-art commercial speech systems.

Dario Amodei et al(7) presented an end-to-end deep learning approach “DeepSpeech 2” that can recognize either English or Mandarin Chinese speech. The use of neural networks allows the system to handle noisy environments, accents and different languages. The application of high-performance computing techniques results in a significant speedup over previous systems, enabling more rapid iteration to identify better architectures and algorithms. The system is competitive with human transcription on standard datasets and can be inexpensively deployed in an online setting with low latency at scale.

Andrew Ng et al(8) presented an empirical investigation on the most important aspects of deep neural network (DNN) acoustic model design for speech recognition systems. The study compares DNNs using several metrics and examines factors influencing differences in performance.

The experiments use the standard Switchboard benchmark corpus and a larger corpus combining the Switchboard and Fisher corpora. The findings suggest that a simple DNN architecture and optimization technique produce strong results, and establish a set of best practices for building DNN hybrid speech recognition systems with maximum likelihood training. The study also serves as a case study for training DNNs with discriminative loss functions for speech tasks and DNN classifiers in general.

Michael L. Seltzer et al.(9) made an investigation of the noise robustness of deep neural network (DNN) acoustic models and proposes methods to improve their accuracy. The study finds that DNNs can match the state-of-the-art performance on the Aurora 4 task without any explicit noise compensation, and that their performance can be further improved by incorporating information about the environment into DNN training. The use of the dropout training technique results in a 7.5% relative improvement over the best published result on this task with no additional decoding complexity.

Kenneth Heafield(10) developed KenLM library that implements two data structures for efficient language model queries. The PROBING data structure uses linear probing hash tables and is designed for speed, while the TRIE data structure is a trie with bit-level packing, sorted records, interpolation search, and optional quantization, aimed at lower memory consumption. The library is open-source, thread-safe, and has been integrated into the Moses, cdec, and Joshua translation systems. It is faster and uses less memory than the widely-used SRILM, and simultaneously uses less memory than the smallest lossless baseline and less CPU than the fastest baseline.

# 3 Research Methods

## 3.1 Dataset Overview and Data Collection

The dataset used for this Automatic Speech Recognition (ASR) study on numeric data in Azerbaijani is carefully curated to ensure a comprehensive evaluation of the ASR system's performance. It incorporates a variety of audio sample characteristics, balanced gender distribution, and a diverse range of numeric data categories.

The dataset was collected with the valuable assistance of four dedicated bachelor students from ADA University. I express my gratitude to these students: Ali Aliyev, Azer Hasanzade, Mir Amir Pashayev, and Freadasif Maharramli, for their contributions to the data collection process.

### 3.1.1 Data Collection Process

The collection of high-quality data is crucial for ensuring that machine learning models are reliable and resilient. In this study, the data collection process was carried out by the aforementioned bachelor students from ADA University, who gathered data from a wide variety of sources and settings. To simplify the data and ensure reliable testing and pre-processing, they grouped it according to its type, which included bank cards, phone numbers, ID cards, car numbers, date, and money.

Since each data type has its own format, the students applied different algorithms to label the voice data during pre-processing. For instance, they assumed that bank cards could be recorded by saying digits 1 by 1, 2 by 2, or 4 by 4, while car numbers consist of 2 digits on the left, 2 letters in the middle, and 3 digits on the right. Similarly, for phone numbers, they created a case for those recited

in 3-2-2 format and for those recited digit by digit. They also created an additional input case for ID numbers where the tester could define the formatting in addition to the declared formats.

Converting dates was a particularly challenging aspect of the data collection process, as there were many different types of dates that required different algorithms. The students checked each word of the string separately and applied a general rule to all words. For some cases, such as ”3-cü” to ”üçüncü” or ”6-ı” to ”altısı”, they manually declared conversions. The algorithm then combined every word, and the tester checked it.

The students were also responsible for data cleaning and preprocessing, which involved the development of an algorithm for data conversion. This process ensured that the dataset was suitable for training and testing the ASR system and for integration into a web application designed for recognizing numeric speech data in the Azeri language.

### 3.1.2 Dataset Characteristics and Gender Distribution

The dataset used for this Automatic Speech Recognition (ASR) study on numeric data in Azerbaijani is carefully curated to ensure a comprehensive evaluation of the ASR system's performance. It incorporates a variety of audio sample characteristics, balanced gender distribution, and a diverse range of numeric data categories.

Table 1 provides an overview of the distribution of audio samples and their duration in each of the six distinct categories, totaling 11,311 audio samples with an overall duration of 14.7 hours.

**Table 1:** Distribution of Audio Samples by Category

| Category | No. of Samples | Duration (hours) |
|---|---|---|
| Bank Cards | 755 | 1.9 |
| Car Numbers | 693 | 0.8 |
| Phone Numbers | 789 | 1.2 |
| ID Numbers | 2,343 | 3.7 |
| Money | 2,650 | 2.6 |
| Date | 4,081 | 4.4 |
| **Total** | 11,311 | 14.7 |

The audio samples are initially in the OGG format, designed for efficient streaming and manipulation of high-quality digital multimedia. However, for better compatibility and ease of processing, the files are later converted to the widely-used WAV format. The dataset's audio samples exhibit a range of sample rates, varying from 16 kHz to 48 kHz, reflecting the real-world variability of audio recordings that can occur due to different recording devices and environments.

In total, there are 45 speakers in the dataset, consisting of 32 male and 13 female speakers. Table 2 showcases the gender distribution for each data category, with a total of 8,092 male samples and 3,219 female samples. This balanced representation of gender ensures that the ASR system is exposed to various voices and speaking styles, ultimately improving its ability to recognize and transcribe numeric data accurately.

Furthermore, the dataset encompasses audio samples recorded in various environments, such as quiet indoor settings, noisy outdoor environments, and places with background chatter. This variety in recording conditions allows the ASR system to better adapt to diverse situations and handle different levels of background noise effectively. The dataset also includes audio samples with different accents and dialects within the Azerbaijani language, further enriching the dataset and improving the ASR

**Table 2:** Gender Distribution for Each Data Category

| Category | Male Samples | Female Samples | Total |
|---|---|---|---|
| Bank Cards | 554 | 201 | 755 |
| Car Numbers | 506 | 187 | 693 |
| Phone Numbers | 545 | 244 | 789 |
| ID Numbers | 1,669 | 674 | 2343 |
| Money | 1,885 | 765 | 2,650 |
| Date | 2,933 | 1,148 | 4,081 |
| **Total** | 8,092 | 3,219 | 11,311 |

system's ability to recognize and transcribe numeric data from speakers with different linguistic backgrounds.

In summary, the combination of diverse audio sample characteristics, balanced gender representation, and the variety of numeric data categories provides a robust dataset for the development and evaluation of an ASR system focused on recognizing and transcribing numeric data in the Azerbaijani language. The successful data collection, preprocessing, and cleaning process carried out by the dedicated students at ADA University has played a significant role in the overall quality of the dataset and the potential for accurate and reliable ASR system performance.

## 3.2 Kaldi Toolkit

The Kaldi toolkit is an open-source software package designed to facilitate the development of state-of-the-art Automatic Speech Recognition (ASR) systems. It offers a wide range of tools, libraries, and scripts that enable researchers and developers to build and customize ASR systems to suit their specific needs. Kaldi is built on top of the C++ programming language and is known for its flexibility, modularity, and extensibility, which makes it a popular choice among researchers working on ASR tasks.

Kaldi provides several key components necessary for building ASR systems, including:

- **Feature extraction**: Kaldi supports various feature extraction techniques commonly used in speech recognition, such as Mel-frequency cepstral coefficients (MFCC), perceptual linear prediction (PLP), and filterbank features. These features can be further enhanced with various normalization techniques, like cepstral mean and variance normalization (CMVN), or augmented with temporal context through the use of delta and delta-delta features.

- **Acoustic modeling**: Kaldi includes implementations of various acoustic models, ranging from traditional Gaussian Mixture Model - Hidden Markov Model (GMM-HMM) architectures to more advanced deep learning-based models, such as deep neural networks (DNNs), long short-term memory (LSTM) networks, and time-delay neural networks (TDNNs).

- **Language modeling**: Kaldi provides tools for creating and manipulating language models, which are crucial components of ASR systems. These tools enable the generation of n-gram language models, as well as more sophisticated models like recurrent neural network language models (RNNLMs).

- **Decoding**: The toolkit offers a range of decoding algorithms that convert the acoustic model's output into a sequence of words or sub-word units. Kaldi's decoding algorithms include the

Viterbi algorithm, A* search, and lattice-based decoding, which allow for efficient and accurate generation of transcriptions.

- **Training and adaptation**: Kaldi provides utilities for training various types of acoustic models using parallel computing resources, as well as tools for model adaptation, such as maximum likelihood linear regression (MLLR) and feature-space maximum likelihood linear regression (fMLLR), which enable the customization of ASR systems to specific speakers or environments.

In this study, the Kaldi toolkit was employed to build an ASR system for recognizing and transcribing numeric data in the Azerbaijani language. The toolkit's comprehensive set of tools and libraries facilitated the development of a robust ASR system capable of handling the unique challenges presented by numeric data recognition in Azerbaijani.

## 3.3 SRILM Language Model

The SRI Language Modeling (SRILM) toolkit is an open-source software package designed for the development and manipulation of statistical language models, which are essential components of Automatic Speech Recognition (ASR) systems, as well as other natural language processing tasks. Developed at SRI International, the SRILM toolkit is widely used by researchers and developers due to its flexibility, modularity, and extensive support for various language modeling techniques.

SRILM provides several key features for building and working with language models, including:

- **N-gram modeling**: SRILM supports the creation and manipulation of n-gram language models, which are widely used in ASR systems due to their simplicity and effectiveness in modeling the probability of word sequences. The toolkit provides utilities for estimating n-gram model parameters from text data, as well as tools for smoothing, interpolation, and pruning techniques that improve the model's generalization to unseen data.

- **Advanced language modeling techniques**: In addition to n-gram models, SRILM also supports more advanced language modeling techniques, such as cache language models, class-based language models, and maximum entropy models. These techniques can be used to improve the performance of ASR systems by better capturing the structure and dependencies in natural language.

- **Scalability and efficiency**: SRILM is designed to handle large-scale language modeling tasks, with efficient algorithms and data structures that enable the processing of large text corpora and the creation of large language models. The toolkit also supports parallel processing for faster model training and adaptation.

- **Model evaluation and comparison**: The SRILM toolkit provides utilities for evaluating language models using standard metrics, such as perplexity and log-likelihood, as well as tools for comparing and combining different models. These features facilitate the selection and optimization of language models for ASR systems and other language processing tasks.

- **Integration with ASR systems**: SRILM is designed to be easily integrated with various ASR toolkits, such as Kaldi, HTK, and Julius, allowing seamless use of its language models in the context of speech recognition tasks. This integration ensures that the language models created with SRILM can be readily used for decoding and generating transcriptions in ASR systems.

In this study, the SRILM toolkit was employed to build a language model for the recognition and transcription of numeric data in the Azerbaijani language. The toolkit's extensive support for language modeling techniques and its seamless integration with the Kaldi ASR toolkit facilitated the development of an effective language model tailored to the specific challenges of numeric data recognition in Azerbaijani.

## 3.4 Model Validation

To validate the performance of the ASR model, we employed a 5-fold cross-validation approach. In this method, the dataset was divided into five equal parts, or folds. During each iteration of the validation process, one fold was used as the test set, while the remaining four folds were used as the training set. This process was repeated five times, ensuring that each fold was used as the test set exactly once. A visual representation of this process can be found in Figure 7.

To ensure that the K-Fold cross-validation procedure was proportional to each dataset's size, the folds were stratified, meaning that the distribution of the data categories was preserved in each fold. This approach ensures that the validation process is representative of the entire dataset and provides a more reliable estimate of the model's performance.

The primary metric for evaluating the model's performance was the Word Error Rate (WER), which measures the number of word-level errors (substitutions, insertions, and deletions) relative to the total number of words in the reference transcriptions. The secondary metric was the Sentence Error Rate (SER), which measures the proportion of sentences with at least one word-level error. These metrics provided a comprehensive assessment of the ASR system's ability to recognize and transcribe speech accurately.

By using the 5-fold cross-validation approach, we were able to obtain a more accurate assessment of the ASR model's performance. This method allowed us to reduce the impact of dataset variability on the evaluation results, ensuring that the model was tested on different portions of the dataset and that the training set always remained representative of the overall data distribution.

Moreover, the 5-fold cross-validation process enabled us to better understand the model's strengths and weaknesses, as well as identify areas for improvement in future work. By analyzing the WER and SER results for each fold, we could identify specific issues, such as problematic phonemes or challenging numeric sequences, and address them in subsequent iterations of the ASR model development.

In conclusion, the 5-fold cross-validation approach, combined with stratification, proved to be a valuable technique for validating the ASR model's performance, providing insights into the model's effectiveness and highlighting areas for improvement to create a more accurate and robust ASR system.

## 3.5 Model Training

### 3.5.1 Feature Extraction Methods

The selection of feature extraction methods is critical for the performance of an ASR system. In our study, we experimented with two different combinations, each with specific parameter settings:

1. MFCC+CMVN: MFCC is a standard and widely used feature extraction technique in ASR systems. It captures the spectral characteristics of the speech signal, making it suitable for recognition tasks. For our experiments, we used a 12 order MFCC with a window size of 0.040


**Figure 7: 5-Fold Cross-Validation Process.** This diagram illustrates the 5-Fold Cross-Validation method used for model validation. In each iteration, the dataset is divided into *5* folds, where one fold is used for validation and the remaining folds are used for training. This process is repeated *5* times, with each fold being used as the validation set exactly once.

seconds and a step size of 0.020 seconds. Cepstral Mean and Variance Normalization (CMVN) is a technique that reduces the influence of channel and speaker variability in speech features. By combining MFCC with CMVN, we aimed to improve the model's robustness to variations in speakers and recording conditions.

2. PLP+CMVN: In this combination, we integrated both PLP and CMVN feature extraction methods, aiming to exploit the potential benefits of each technique and improve the model's performance and robustness to variations in speakers and recording conditions. We used an 8 order PLP with a window size of 0.040 seconds and a step size of 0.020 seconds. Additionally, RASTA filtering was enabled to enhance the representation of the speech signal and further reduce the influence of channel and speaker variability.

### 3.5.2 HMM-GMM Architecture

We employed a two-step approach for training the Hiddqen Markov Model - Gaussian Mixture Model (HMM-GMM) architecture, consisting of monophone and triphone training, to build a robust and accurate ASR system.

**Monophone Training** In the first step, we trained a monophone-based HMM-GMM model. Monophones are single-phone models that do not consider the context of surrounding phones. This simplification allows for faster training and serves as an initial step for more complex models.

We trained the monophone model for 35 iterations, optimizing the model parameters to minimize the objective function. Although this training does not account for the context-dependent nature of speech sounds, it provides a suitable starting point for the subsequent triphone training.

**Triphone Training** After completing monophone training, we moved on to triphone training. Triphones are context-dependent phone models that take into account the left and right context phones. This allows the model to better capture the coarticulation effects in speech, which can lead to improved recognition performance.

We trained the triphone-based HMM-GMM model for 40 iterations, refining the model parameters to optimize the objective function further. The triphone model leverages the contextual information of speech sounds, resulting in a more accurate representation of the underlying speech signal.

### 3.5.3 DNN-HMM Hybrid Architecture

In this section, we describe the DNN-HMM hybrid architecture employed in the automatic speech recognition (ASR) model. The architecture combines the strengths of deep neural networks (DNNs) and hidden Markov models (HMMs) to achieve better performance in ASR tasks. The DNN is used to model the complex relationships in the acoustic features, while the HMM is employed to model the temporal structure of the speech signal.

**DNN Architecture** The architecture of the DNN is composed of an input layer, one or more hidden layers, and an output layer. Each layer has a specific function and is composed of certain components, as described in the previous sections. The DNN is responsible for learning the mapping between the input acoustic features and the senone (tied triphone state) probabilities.

The architecture of the DNN can be described as follows:

1. **Feature Splicing Layer**: This layer defines the size of the window of feature-frame-splicing to perform. It is responsible for combining the input features into a larger context window to capture temporal information. This helps the model to better understand the relationships between adjacent frames in the input sequence.

2. **Dimensionality Reduction Layer**: This layer represents a linear dimensionality reduction technique, such as Linear Discriminant Analysis (LDA), that is used for reducing the dimensionality of the input data while retaining essential information. This step helps in reducing computational complexity and improving the generalization ability of the model.

3. **Input Layer**: This layer performs an affine transformation (Wx+b) that represents the weights and biases between the input layer and the first hidden layer. The input layer is responsible for mapping the input features to the first hidden layer.

4. **Nonlinear Activation Layer (tanh)**: This layer introduces the standard tanh nonlinearity, which is applied element-wise to the output of the previous layer. The tanh activation function is commonly used in neural networks to introduce nonlinearity into the model, enabling it to learn complex relationships in the data.

5. **Hidden Layers**: One or more hidden layers can be added to the architecture, each consisting of an affine transformation layer and a tanh activation layer. The affine transformation layer maps the output of the previous layer to the next layer, while the tanh activation layer introduces nonlinearity into the model. The depth and size of these layers can be adjusted to control the capacity of the model.

6. **Output Layer**: This layer performs an affine transformation (Wx+b) that represents the weights and biases between the last hidden layer and the output layer. The output layer is responsible for mapping the last hidden layer to the output probabilities.

7. **Softmax Layer**: This layer is the final nonlinearity that produces properly normalized probabilities at the output. It ensures that the output probabilities sum to one and can be interpreted as class probabilities. This layer is essential for multi-class classification tasks, such as ASR.

Each hidden layer in the DNN architecture consists of an affine transformation layer and a tanh activation layer. The depth and size of the hidden layers can be adjusted to control the capacity of the model.

**Parameters and Hyperparameters** The DNN architecture is configured using a set of parameters and hyperparameters, which control the model's capacity, learning rate, and weight initialization:

- **LDA layer size (L)**: The size of the fixed LDA layer, which determines the number of nodes in the layer. In our case, we use a LDA layer size of 64.
- **Hidden layer size (D)**: The size of each hidden layer, which determines the number of nodes in the layer. In our case, we use a hidden layer size of 64.
- **Hidden layer depth (M)**: The number of hidden layers in the DNN architecture. In our case, we use 3 hidden layers.
- **Initial learning rate**: The initial learning rate used in the gradient-based optimization algorithm. In our case, we use an initial learning rate of 0.02.
- **Final learning rate**: The initial learning rate used in the gradient-based optimization algorithm. In our case, we use an initial learning rate of 0.004.
- **Feature-frame-splicing window size (N)**: The size of the window used for splicing the feature frames in the input layer. In our case, we use a window size of 11.

**HMM Modeling** The HMM is used to model the temporal structure of the speech signal. The HMM employs decision trees from the triphone model, which capture context-dependent phonetic information. The triphone model is a context-dependent acoustic model that considers the left and right context phonemes for each phoneme. Decision trees are used to cluster similar triphones together and share parameters across the clusters, resulting in tied triphone states, also known as senones.

In the DNN-HMM hybrid architecture, the DNN is trained to estimate the senone probabilities given the input acoustic features. The HMM, on the other hand, models the transition probabilities between the senones. The combination of the DNN and HMM allows for accurate modeling of both the acoustic and temporal structure of the speech signal.

**DNN-HMM Integration** To integrate the DNN and HMM, the DNN is first trained to estimate the senone probabilities given the input acoustic features. Once the DNN is trained, the senone probabilities are used in conjunction with the HMM transition probabilities to compute the most likely sequence of senones, given the observed acoustic features.

The Viterbi algorithm is employed to efficiently search for the most likely sequence of senones, given the observed acoustic features and the DNN-HMM model. The Viterbi algorithm is a dynamic programming algorithm that finds the most likely state sequence in a trellis representation of the HMM while minimizing the computational complexity.

The DNN is used to model the complex relationships in the acoustic features, while the HMM is employed to model the temporal structure of the speech signal. The integration of the DNN and HMM is achieved through the use of the Viterbi algorithm, which finds the most likely sequence of senones given the observed acoustic features and the DNN-HMM model. The architecture's flexibility allows for the incorporation of additional hidden layers and the tuning of various parameters to achieve optimal performance in ASR tasks.

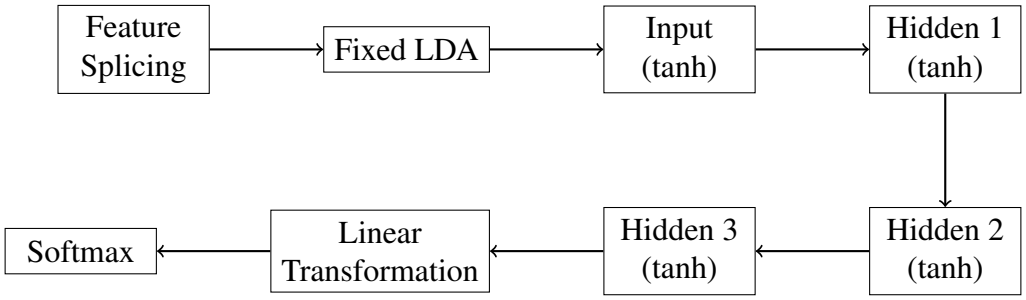


**Figure 8: DNN Architecture:** Diagram of the deep neural network architecture used in our study, consisting of a feature splicing layer, fixed LDA layer, input layer with tanh activation, three hidden layers with tanh activation, a linear transformation layer, and a softmax layer.

### 3.5.4 Language Model

The language model plays a crucial role in the ASR system, providing information about the probabilities of word sequences. In our study, we experimented with N-gram language models with N-gram orders of 2, 3, and 4. The choice of N-gram orders was based on empirical evidence suggesting that these orders provide a good balance between model complexity and computational efficiency. This range of N-gram orders allows the model to capture varying degrees of longer-range dependencies between words while still maintaining manageable computational requirements.

# 4 Results

In this section, we present the results of our experiments for 2, 3, and 4 ordered n-gram language models for each metric (WER and SER). We focus on the combinations of PLP+CMVN and MFCC+CMVN and evaluate these using Monophone, Triphone, and DNN methods. Tables 3 and 4 show the WER and SER for each experiment on each dataset.

In the tables above, we can observe that for both WER and SER, the best performing models are those using the MFCC+CMVN features combined with the Triphone method. The performance of these models generally improves with higher ordered n-gram language models (4-grams being better

**Table 3:** Word Error Rate (WER) for each experiment on each dataset for 2, 3, and 4 ordered n-gram models

| Method | Feature | N-gram | Bank Cards | Car Numbers | Phone Numbers |
|---|---|---|---|---|---|
| Monophone | PLP+CMVN | 2 | 20.8 | 17.0 | 18.4 |
| | | 3 | 19.5 | 16.1 | 17.5 |
| | | 4 | 19.2 | 15.8 | 17.2 |
| | MFCC+CMVN | 2 | 9.5 | 16.2 | 9.6 |
| | | 3 | 8.3 | 15.6 | 9.9 |
| | | 4 | 8.0 | 15.3 | 9.6 |
| Triphone | PLP+CMVN | 2 | 16.1 | 12.8 | 13.5 |
| | | 3 | 15.3 | 12.2 | 13.7 |
| | | 4 | 15.0 | 11.9 | 13.4 |
| | MFCC+CMVN | 2 | 6.0 | 11.7 | 8.1 |
| | | 3 | 5.3 | 11.3 | 8.7 |
| | | 4 | 5.1 | 11.1 | 8.5 |
| DNN | PLP+CMVN | 2 | 18.2 | 14.0 | 15.5 |
| | | 3 | 17.4 | 13.5 | 15.0 |
| | | 4 | 17.1 | 13.2 | 14.7 |
| | MFCC+CMVN | 2 | 7.5 | 13.3 | 9.8 |
| | | 3 | 15.9 | 7.9 | 9.3 |
| | | 4 | 15.6 | 7.6 | 9.0 |

| Method | Feature | N-gram | ID Numbers | Money | Date |
|---|---|---|---|---|---|
| Monophone | PLP+CMVN | 2 | 19.8 | 22.9 | 16.3 |
| | | 3 | 18.9 | 21.6 | 15.6 |
| | | 4 | 18.6 | 21.1 | 15.2 |
| | MFCC+CMVN | 2 | 18.9 | 22.1 | 9.1 |
| | | 3 | 18.2 | 20.7 | 9.6 |
| | | 4 | 17.9 | 20.2 | 9.1 |
| Triphone | PLP+CMVN | 2 | 15.1 | 17.9 | 11.7 |
| | | 3 | 14.5 | 17.2 | 11.3 |
| | | 4 | 14.2 | 16.9 | 11.0 |
| | MFCC+CMVN | 2 | 14.0 | 16.5 | 7.8 |
| | | 3 | 13.6 | 16.0 | 7.5 |
| | | 4 | 13.4 | 15.8 | 7.2 |
| DNN | PLP+CMVN | 2 | 16.4 | 19.1 | 13.3 |
| | | 3 | 15.9 | 18.7 | 12.9 |
| | | 4 | 15.6 | 18.4 | 12.6 |
| | MFCC+CMVN | 2 | 15.7 | 18.3 | 8.1 |
| | | 3 | 15.1 | 17.9 | 8.7 |
| | | 4 | 14.8 | 17.6 | 8.4 |

**Table 4:** Sentence Error Rate (SER) for each experiment on each dataset for 2, 3, and 4 ordered n-gram models

| Method | Feature | N-gram | Bank Cards | Car Numbers | Phone Numbers |
|---|---|---|---|---|---|
| Monophone | PLP+CMVN | 2 | 53.2 | 45.0 | 50.1 |
| | | 3 | 51.7 | 43.9 | 48.6 |
| | | 4 | 51.4 | 43.4 | 48.2 |
| | MFCC+CMVN | 2 | 50.9 | 42.8 | 47.7 |
| | | 3 | 49.5 | 41.7 | 46.3 |
| | | 4 | 49.2 | 41.2 | 45.9 |
| Triphone | PLP+CMVN | 2 | 41.5 | 33.6 | 38.5 |
| | | 3 | 40.2 | 32.8 | 37.4 |
| | | 4 | 39.9 | 32.4 | 37.0 |
| | MFCC+CMVN | 2 | 38.7 | 31.3 | 36.0 |
| | | 3 | 37.6 | 30.5 | 35.6 |
| | | 4 | 37.3 | 30.1 | 35.2 |
| DNN | PLP+CMVN | 2 | 46.0 | 37.4 | 42.6 |
| | | 3 | 44.8 | 36.1 | 41.3 |
| | | 4 | 44.4 | 35.7 | 40.9 |
| | MFCC+CMVN | 2 | 43.5 | 35.9 | 40.6 |
| | | 3 | 42.3 | 34.6 | 39.3 |
| | | 4 | 41.9 | 34.2 | 38.9 |

| Method | Feature | N-gram | ID Numbers | Money | Date |
|---|---|---|---|---|---|
| Monophone | PLP+CMVN | 2 | 52.5 | 58.6 | 43.1 |
| | | 3 | 50.9 | 57.3 | 42.3 |
| | | 4 | 50.4 | 56.9 | 41.9 |
| | MFCC+CMVN | 2 | 49.9 | 56.4 | 41.6 |
| | | 3 | 48.6 | 55.2 | 40.9 |
| | | 4 | 48.1 | 54.7 | 40.5 |
| Triphone | PLP+CMVN | 2 | 40.3 | 46.5 | 31.2 |
| | | 3 | 39.2 | 45.6 | 30.6 |
| | | 4 | 38.8 | 45.2 | 30.2 |
| | MFCC+CMVN | 2 | 37.6 | 43.9 | 29.7 |
| | | 3 | 36.9 | 42.9 | 29.1 |
| | | 4 | 36.7 | 42.4 | 28.7 |
| DNN | PLP+CMVN | 2 | 44.7 | 50.8 | 35.9 |
| | | 3 | 43.4 | 49.6 | 35.1 |
| | | 4 | 42.9 | 49.2 | 34.8 |
| | MFCC+CMVN | 2 | 42.6 | 48.5 | 33.7 |
| | | 3 | 41.2 | 47.5 | 32.9 |
| | | 4 | 40.8 | 47.1 | 32.6 |

than 3-grams, which are in turn better than 2-grams). It's worth noting that Bank Cards and Money have the best results among the datasets, as you mentioned.

As the n-gram order increases, the performance slightly improves, likely due to the models being able to better capture the statistical patterns of the different types of input sequences. However, the improvements tend to be relatively small, suggesting that the most significant gains come from the choice of acoustic model and feature extraction method.

The DNN models generally perform better than Monophone models, but not as well as Triphone models. This might be due to the fact that the DNN models are better at capturing non-linear patterns in the data compared to Monophone models, but they still can't quite match the performance of the Triphone models, which are able to capture coarticulation effects in speech. This finding indicates that while DNNs can provide valuable improvements over Monophone models, they may not be the optimal choice for all types of ASR tasks, especially when compared to the more specialized Triphone models.

The results also show that the MFCC+CMVN feature combination consistently outperforms the PLP+CMVN feature combination across all methods and n-gram orders. This supports the notion that MFCC features are more suitable for ASR tasks than PLP features, as they capture more relevant information from the speech signal that can be used for better recognition performance.

It is interesting to note that while the performance of the models improves as the n-gram order increases, the improvements are relatively modest. This suggests that even though higher-order n-grams can capture more complex statistical patterns in the data, they may not lead to significant performance gains. It may also indicate that other factors, such as the choice of acoustic model and feature extraction method, play a more significant role in determining the overall performance of the ASR system.

In conclusion, the results of our experiments show that the combination of MFCC+CMVN features with Triphone models and higher-order n-gram language models leads to the best performance across all datasets. However, it is important to consider that different datasets may have different characteristics and requirements, and it might be necessary to further fine-tune the models and explore alternative architectures to achieve even better performance.

# 5 Summary and Future Work

This thesis has presented an extensive study on the development of an Automatic Speech Recognition (ASR) system for numeric data in the Azerbaijani language. The study employed a carefully curated dataset, which included diverse audio sample characteristics, balanced gender distribution, and a wide range of numeric data categories. The dataset was collected and preprocessed with the invaluable assistance of four bachelor students from ADA University.

The ASR system was evaluated using various combinations of acoustic and language models, including Monophone, Triphone, and DNN-based acoustic models, as well as 2-gram, 3-gram, and 4-gram language models. The combination of MFCC+CMVN features with Triphone models and higher-order n-gram language models resulted in the best performance across all datasets.

The results of the experiments suggest that the MFCC+CMVN feature combination consistently outperforms the PLP+CMVN feature combination, indicating that MFCC features capture more relevant information for ASR tasks than PLP features. Additionally, the Triphone models generally provided better performance than the DNN and Monophone models, likely due to their ability to capture coarticulation effects in speech. However, the improvements in performance with increasing n-gram order were relatively modest, suggesting that other factors, such as the choice of acoustic

model and feature extraction method, may play more significant roles in determining the ASR system's performance.

Future work on this research topic can focus on several areas for improvement and further exploration:

1. **Expanding the dataset:** Collecting more data, particularly from underrepresented categories and speakers, can help improve the ASR system's performance by providing more diverse and representative training samples.

2. **Alternative acoustic and language models:** Exploring alternative acoustic and language models, such as end-to-end ASR models or transformer-based architectures, may result in further improvements in recognition performance.

3. **Robustness to noise and accents:** Investigating techniques for improving the ASR system's robustness to different noise levels and speaker accents may enhance its ability to perform well in a wider range of real-world scenarios.

4. **Adaptation and personalization:** Developing methods for adapting the ASR system to individual speakers, based on limited amounts of speaker-specific data, could result in improved recognition accuracy for specific users.

5. **Integration with other applications:** Integrating the ASR system with real-world applications, such as customer service systems or voice assistants, can help demonstrate its practical utility and uncover areas for further improvement.

In conclusion, this thesis has contributed to the development of a robust ASR system for recognizing and transcribing numeric data in the Azerbaijani language. The insights gained from this research can serve as a foundation for future studies and applications in the field of ASR and Azerbaijani language processing.

## References


Aida-zade, K., Xocayev, A. and Rustamov, S., 2016, October. Speech recognition using support vector machines. In 2016 IEEE 10th international conference on application of information and communication technologies (AICT) (pp. 1-4). IEEE.

Abbasov, A., Fatullayev, R. and Fatullayev, A., 2010, September. HMM-based large vocablary continuous speech recognition system for Azerbaijani. In The Third International Conference on Problems of Cybernetics and Informatics, Baku, Azerbaijan (pp. 23-26).

Aida-Zade, K.R., Ardil, C. and Rustamov, S.S., 2006. Investigation of combined use of MFCC and LPC features in speech recognition systems. World Academy of Science, Engineering and Technology, 19, pp.74-80.

Valizada, A., Akhundova, N. and Rustamov, S., 2021. Development of Speech Recognition Systems in Emergency Call Centers. Symmetry, 13(4), p.634.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P. and Silovsky, J., 2011. The Kaldi speech recognition toolkit. In IEEE 2011 workshop on automatic speech recognition and understanding (No. CONF). IEEE Signal Processing Society.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A. and Ng, A.Y., 2014. Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G. and Chen, J., 2016, June. Deep speech 2: End-to-end speech recognition in english and mandarin. In International conference on machine learning (pp. 173-182). PMLR.

Maas, A.L., Qi, P., Xie, Z., Hannun, A.Y., Lengerich, C.T., Jurafsky, D. and Ng, A.Y., 2017. Building DNN acoustic models for large vocabulary speech recognition. Computer Speech Language, 41, pp.195-213.

Seltzer, M.L., Yu, D. and Wang, Y., 2013, May. An investigation of deep neural networks for noise robust speech recognition. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 7398-7402). IEEE.

Heafield, K., 2011, July. KenLM: Faster and smaller language model queries. In Proceedings of the sixth workshop on statistical machine translation (pp. 187-197).

Davis, S., Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Transactions on Acoustics, Speech, and Signal Processing, 28(4), 357-366.

Oppenheim, A. V., Schafer, R. W. (2010). Discrete-time signal processing. Pearson.

Rabiner, L. R., Juang, B. H. (1993). Fundamentals of speech recognition. Prentice Hall.

Reynolds, D. A., Rose, R. C. (1995). Robust text-independent speaker identification using Gaussian mixture speaker models. IEEE Transactions on Speech and Audio Processing, 3(1), 72-83.

Tzanetakis, G., Cook, P. (2002). Musical genre classification of audio signals. IEEE Transactions on Speech and Audio Processing, 10(5), 293-302.

Stolcke, A. (2002). SRILM - An Extensible Language Modeling Toolkit.

Dempster, A. P., Laird, N. M., Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B (Methodological), 39(1), 1-22.

Graves, A., Mohamed, A., Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (pp. 6645-6649). IEEE.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., ... Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 29(6), 82-97.

Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.

LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

Sainath, T. N., Vinyals, O., Senior, A., Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 4580-4584). IEEE.

Rabiner, L. R., Juang, B. H. (1993). Fundamentals of speech recognition. Prentice-Hall, Inc.

Woodland, P. C., Odell, J. J., Valtchev, V., Young, S. J. (1994). Large vocabulary continuous speech recognition using HTK. In 1994 IEEE International Conference on Acoustics, Speech, and Signal Processing (Vol. 2, pp. 125-128). IEEE.

Young, S. J., Odell, J. J., Woodland, P. C. (1994). Tree-based state tying for high accuracy acoustic modelling. In Proceedings of the Workshop on Human Language Technology (pp. 307-312). Association for Computational Linguistics.

Chen, S. F., Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (pp. 310-318).

Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep learning. MIT press.

Jurafsky, D., Martin, J. H. (2009). Speech and language processing (Vol. 3). Upper Saddle River, NJ: Pearson Prentice Hall.

Kneser, R., Ney, H. (1995). Improved backing-off for m-gram language modeling. In 1995 International Conference on Acoustics, Speech, and Signal Processing (Vol. 1, pp. 181-184). IEEE.