# Comparative Analysis of Speaker Diarization Techniques using Different Clustering Methods on CNN-Based Speaker Segmentation for Enhanced Precision and Recognition

**Abstract.** Speaker diarization is an important part of audio processing, which is critical for identifying and distinguishing individual speakers within a continuous audio stream. To address this need, our research focuses on speaker segmentation, which involves partitioning an audio signal into segments based on speaker identity. *Convolutional Neural Networks (CNNs)* are used to extract meaningful acoustic features, particularly *Mel Frequency Cepstral Coefficients (MFCCs)*, from audio data. We explore various clustering techniques, including agglomerative clustering with PLDA scoring, segmentation by weighted aggregation, and spectral clustering, to group similar speaker representations and assess the efficacy of different clustering methods for speaker recognition. Our findings have significant implications for enhancing the precision and efficiency of speaker identification in diverse audio-based applications.

**1. Introduction.** In the realm of audio processing, the accurate identification and segmentation of speakers is crucial for applications such as enhancing accessibility in audiobooks for the visually impaired, improving the accuracy of video transcriptions, and enabling efficient analysis of meeting recordings. To address this need, our research focuses on speaker diarization, which involves partitioning an audio signal into segments based on speaker identity. This process is essential for efficiently organizing and analyzing spoken content.

Traditional speaker diarization systems often rely on *Hidden Markov Models (HMMs)* or *Gaussian Mixture Models (GMMs)*. However, recent advances in deep learning have shown promising results in this area. Our methodology explores the use of CNNs [6, 14, 11] to extract discriminative acoustic features, or Mel Frequency Cepstral Coefficients, from audio data, building upon the success of deep learning in related audio processing tasks. These features serve as the foundation for speaker representation, facilitating the effective clustering of speakers. We explore various clustering techniques, including agglomerative with PLDA and CSS scoring, $K$-means, and spectral clustering, to group similar speaker representations and assess the effectiveness of each method based on various feature classification techniques.

Through this research, our goal was not only to improve the accuracy of speaker diarization but also to gain a deeper understanding of the strengths and weaknesses of different clustering methods in this context. Our findings could lead to more robust and efficient speaker diarization systems for companies like Netflix and YouTube with high demand for audio transcription tasks.

The paper is organized as follows. We start with our research question in section 2. We then discuss the background and related works of speaker diarization in section 3, followed by introducing our methods in section 4, our main results in section 5, a discussion of our work in section 6, and concluded with our future work and conclusion in section 7 and section 8. Our code base can be found in this repository[1].

**2. Research Question.** How do clustering methods, specifically agglomerative clustering, spectral clustering, and $K$-means, compare in terms of computational efficiency and clustering accuracy across segmented audio datasets for speaker diarization, and what are the implications of these differences for optimizing real-time audio processing systems?

**3. Background and Related Works.** Speaker diarization is pivotal for distinguishing individual speakers within a continuous audio stream, and recent advancements in clustering and feature extraction have pushed the boundaries of what is possible in this field [1, 11]. Companies that leverage transcription technology for their applications require speaker diarization to ensure an inclusive environment is provided for their users. For example, the work done at Netflix emphasizes the importance of accurate transcription in media production, where speaker identification is crucial for their multilingual environment and customer base [7]. For media companies like Netflix, properly transcribing their content is essential for an enhanced user experience, not to mention its implications for deafness or hearing loss. Netflix's *Timed Text Authoring Lineage (TTAL)* [7] highlights the complexity of transcription tasks, particularly when working with multiple speakers or noisy audio environments. Their approach migrates crucial information, dialogue, timecodes, metadata, and language details,

[1] Our code base and data can be found on our GitHub

to their third-party applications for accurate subtitles. Issues arise because a key component of TTAL is pre-supplied scripts that provide most of the data needed for transcription, reducing algorithmic overhead. Our approach considers a broader application with no reliance on third-party software, using our method of integrating CNN-based feature extraction leveraging Mel Frequency Cepstral Coefficients with various customized clustering algorithms to extract the speaker identities. While TTAL proves to be a non-intrusive process, we aim to provide a one-size-fits-all approach. The application of MFCCs enables our clustering models to operate with enhanced precision, particularly in separating overlapping speech, which traditional methods struggle with [10]. In particular, SVMs and LDA, as shown in the research by [9], demonstrated adequate results in low-dimensional datasets compared to higher-dimensional data with noisier environments. Our approach aims to reanalyze such techniques and build upon the limitations of past models. By embedding speaker data into a lower-dimensional space, PCA-$K$-means makes it easier to differentiate speakers even when their voice patterns are similar, something we have adopted for our more complex datasets [17].

In comparison to previous works, such as [2]'s exploration of diarization techniques, we adopt a multi-clustering approach that includes agglomerative clustering with PLDA scoring, enhancing scalability while preserving high accuracy across diverse data sets. By evaluating the strengths of each clustering method, our system becomes more adaptive and capable of handling real-world audio scenarios. This flexibility in handling both high-quality and challenging audio data, along with innovative feature extraction and clustering, positions our research at the forefront of speaker diarization advancements.

With these in mind, our research has two main goals, transcription accuracy and content understanding. We hope to address these concerns and provide a more comprehensive model for audio transcriptions.

**3.1. Transcription Accuracy.** Speaker diarization helps improve the accuracy of speech-to-text transcription by clustering each segment of audio to the correct speaker [1]. This is crucial in scenarios where multiple speakers are present, such as meetings, interviews, conference calls, or even audiobooks. With modern technologies lacking speaker recognition capabilities, this is increasingly important for users who may have disabilities that require technical support.

**3.2. Content Understanding.** Diarization helps in understanding the conversation flow and context by distinguishing between speakers [17]. It allows for better analysis of who said what, which can be valuable for content summarizing, which can be very important in high-stakes situations like trial proceedings or executive meetings.

## 4. Methods.

In this section, we detail the datasets utilized in this study and summarize the algorithm used to perform the analysis and the variety of clustering methods used.

**4.1. Datasets.** In testing, we utilized two datasets. The preliminary tests used a pretrained MATLAB dataset, discussed in subsection 4.1.1, and our latest testing utilized a more dynamic dataset provided under the Creative Commons Attribution 4.0 International Licence by the International Computer Science Institute (ICSI), discussed in subsection 4.1.2.

**4.1.1. MATLAB Dataset.** Our initial training data consists of a pretrained speaker diarization system and a corresponding dataset from the Statistics and Machine Learning Tool-

box [6]. The audio recordings, visualized in Figure 1, contain a maximum of five speakers along with corresponding ground truth annotations stored in a table format. The ground truth annotations include timestamps and speaker labels indicating exactly when each speaker is talking throughout the recording. The entire dataset is approximately 22 MB in size, with enough audio data for basic testing and evaluation purposes.

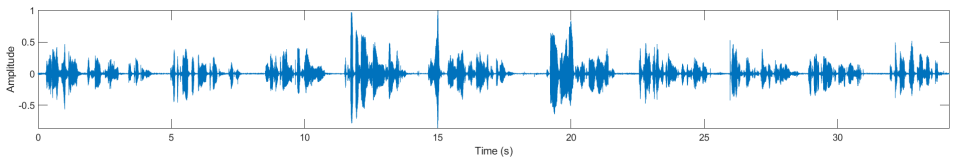


**Figure 1.** *Original MATLAB Dataset Audio Map*

This specific dataset was chosen for a couple of reasons. Firstly, having the annotated data is crucial for evaluating the performance of our speaker diarization system since it provides the correct answers to compare against the system's predictions. Additionally, the multiple speakers in the audio allow for reliable testing of the system's diarization ability in differentiating a many-speaker setting. Lastly, the audios were chosen for their length, as testing multiple clustering algorithms can be taxing with larger datasets.

**4.1.2. International Computer Science Institute Meeting Corpus.** The International Computer Science Institute corpus [3] is a collection of recorded meetings with corresponding transcripts and annotations. The dataset is structured with two main components: annotations, available in two packages - a 19MB core package with basic transcripts and dialogue acts, and a 53MB extended package including additional annotations for topics, hotspots, summaries, and audio signals from various meetings. Each meeting is identified by a unique code - like Bed002, Bmr001, etc. - and can be downloaded individually. The audio files are available in different formats, including headset mix, approximately 120MB per meeting, and individual channel headset recordings, around 350MB per meeting. For our purposes, the initial testing occurred with the Bed002 meeting data, consisting of a 119MB audio file and a 239KB transcript, providing a complete recording and detailed transcription of a single meeting session. The data is formatted in NXT (NITE XML Toolkit) format, requiring version 1.4.4 for proper usage. Further testing utilizied the entirety of the ICSI dataset.

This corpus was utilized in the second stage of our testing for its comprehensive meeting recordings and high-quality annotations. The ICSI dataset provided ideal testing conditions with its diverse speaker interactions, detailed transcripts, and substantial audio duration, as opposed to our first dataset. The meeting recordings allowed us to thoroughly evaluate our speaker diarization system under realistic meeting conditions where multiple participants engage in natural conversation. The availability of precise transcriptions and dialogue act annotations enabled comprehensive validation of our system's speaker segmentation accuracy.

**4.2. Extracting x-Vectors and MFCC features.** Our speaker diarization system utilizes a pretrained x-vector system [14] to characterize audio segments. The system first extracts MFCCs using an `audioFeatureExtractor` object. These MFCCs are then standardized using pre-computed mean and standard deviation factors derived from a representative dataset. The

standardized MFCCs are grouped into 2-second segments with 0.1-second hops. A pretrained deep neural network [14] processes each segment to extract an x-vector, a compact representation of the speaker's voice characteristics. Finally, a linear discriminant analysis (LDA) projection matrix is applied to reduce the dimensionality of the x-vectors. This process results in a set of low-dimensional x-vectors that effectively capture speaker-specific information.

**4.3. Clustering Methods.** After extracting x-vectors, we employ various clustering methods to group similar audio segments and identify different speakers. This study explores five distinct clustering approaches: agglomerative clustering with *Probabilistic Linear Discriminant Analysis (PLDA)* scoring, agglomerative clustering with *Cosine Similarity Scoring (CSS)*, $K$-means with CSS scoring, spectral clustering, and Principal Component Analysis (PCA) with $K$-means. These methods are implemented in MATLAB and are available online [6]. We provide a detailed explanation of each method below.

**4.3.1. Agglomerative with PLDA Scoring.** Agglomerative clustering, a hierarchical clustering method with roots in biological taxonomy [15], is combined with PLDA scoring in this approach. PLDA, as described in [5], is a technique commonly employed in speaker verification and diarization to model the relationship between speaker embeddings (x-vectors) and speaker identities [14, 11]. In this approach, the similarity between x-vectors, denoted as $x_i$ for the $i$-th speaker, is evaluated using PLDA scores, facilitating the merging of clusters, represented by $C_k$, based on the likelihood that they originate from the same speaker. The distance between clusters is denoted as $d(A, B)$. The algorithm iteratively merges pairs of clusters with the highest similarity, represented by the similarity score $S(i, j)$ between x-vectors $x_i$ and $x_j$, until a predefined stopping criterion is met.

While agglomerative clustering does not have a direct objective function, its goal is to maximize the similarity within clusters and minimize the similarity between clusters. This can be represented informally as:

$$\max \sum_{k=1}^{K} \sum_{i,j \in C_k} S(i, j) \tag{4.1}$$

where $S(i, j)$ is the similarity between x-vectors $x_i$ and $x_j$, and $C_k$ represents the $k$-th cluster.

**Algorithm 4.1** Agglomerative Clustering with PLDA Scoring

**Require:** A set of x-vectors $\{x_i\}$, PLDA model
**Ensure:** Cluster assignments $T$

```
Compute pairwise similarity scores S(i,j) = PLDA(x_i, x_j)
Initialize each x-vector as its own cluster C_k = {x_k}
while number of clusters > maxclusters do
    Find clusters A and B with highest similarity S(A,B)
    Merge clusters A and B
    Update similarity scores
end while
return Cluster assignments T
```

**4.3.2. Agglomerative with CSS Scoring.** Agglomerative clustering, originating from work like [15], is enhanced with Consistency Scoring of Segments (CSS) in this method. CSS evaluates the stability of clusters by comparing cluster assignments across multiple iterations of the algorithm. This helps assess the consistency with which x-vectors, denoted as $x_i$, are assigned to the same cluster, providing a measure of cluster reliability. Higher CSS scores, represented by $CSS(C_k)$ for cluster $C_k$, indicate more stable and well-defined clusters. This method aids in determining the optimal number of clusters, as the stability measure ensures that speaker segments are robustly assigned to their respective clusters. $N_k$ represents the number of data points in cluster $C_k$.

Given that

$$CSS(C_k) = \frac{1}{N_k} \sum_{i \in C_k} \sum_{j \in C_k} S(i,j), \tag{4.2}$$

the objective function of agglomerative clustering with CSS scoring can be expressed as:

$$\max \sum_{k=1}^{K} CSS(C_k), \tag{4.3}$$

where $K$ is the number of clusters. This objective aims to maximize the overall stability of the clustering solution.

**Algorithm 4.2** Agglomerative Clustering with CSS Scoring

**Require:** A set of x-vectors $\{x_i\}$, number of clusters $K$
**Ensure:** Cluster assignments $T$
1: Perform agglomerative clustering to obtain initial clusters $C_k$
2: **for** each cluster $C_k$ **do**
3: Calculate $CSS(C_k)$
4: **end for**
5: Evaluate cluster stability based on $CSS(C_k)$ scores
6: Merge or split clusters to optimize stability
7: **return** Cluster assignments $T$

**4.3.3. $K$-means.** The $K$-means algorithm, a classic clustering technique with origins in [4], partitions a set of x-vectors, denoted as $x_i$, into $K$ distinct clusters, represented by $C_k$. This iterative algorithm begins by assigning each x-vector to the cluster with the nearest centroid, denoted as $\mu_k$ for cluster $C_k$. Subsequently, the centroids are updated to reflect the mean of the x-vectors assigned to each cluster. This process is repeated until the cluster assignments stabilize, indicating convergence. In the context of speaker diarization, $K$-means groups x-vectors corresponding to different speakers, where the number of clusters ($K$) typically aligns with the anticipated number of speakers in the audio. The objective function of $K$-means, denoted as $J$, is to minimize the total variance within clusters.

The objective function of $K$-means can be expressed as:

$$J = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|^2, \tag{4.4}$$

This objective aims to find cluster assignments and centroids that minimize the sum of squared distances between x-vectors and their corresponding cluster centroids.

**Algorithm 4.3** $K$-means Clustering

**Require:** A set of x-vectors $\{x_i\}$, number of clusters $K$
**Ensure:** Cluster assignments $T$

```
Randomly initialize K cluster centroids μ_k
repeat
    for each x-vector x_i do
        Assign x_i to the cluster C_k with the nearest centroid μ_k
    end for
    for each cluster C_k do
        Update centroid μ_k as the mean of x-vectors in C_k
    end for
until cluster assignments stabilize
return Cluster assignments T
```

The objective function ($J$) of $K$-means aims to minimize the total variance within clusters:

$$J = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|^2.$$

**4.3.4. Spectral Clustering.** Spectral clustering, with roots in graph theory and spectral analysis [16], offers a powerful approach to partitioning data points into clusters based on the eigenvalues of a similarity matrix. Early influential work in this area includes the normalized cuts algorithm proposed by Shi and Malik [13], which aims to find balanced partitions of a graph. In the context of speaker diarization, the x-vectors, denoted as $x_i$, are represented as nodes in a graph, with edges connecting nodes based on the similarity between the corresponding speakers, represented by the similarity score $S_{ij}$ between x-vectors $x_i$ and $x_j$. The algorithm constructs a graph Laplacian matrix, denoted as $L$, and utilizes its eigenvectors to embed the x-vectors into a lower-dimensional space. The degree matrix is denoted as $D$. Clusters are then formed in this transformed space, often using a traditional clustering method like $K$-means. Spectral clustering excels in scenarios where clusters exhibit non-convex shapes or when data points deviate from forming spherical clusters, as frequently observed in speaker data.

A common objective in spectral clustering, particularly when using the normalized graph Laplacian ($L_{norm} = D^{-1/2} L D^{-1/2}$), is to minimize the following:

$$\min_{U} \operatorname{Tr}(U^T L_{norm} U) \quad \text{subject to} \quad U^T U = I, \tag{4.5}$$

where $U$ is a matrix whose columns are the eigenvectors of $L_{norm}$ corresponding to the $K$ smallest eigenvalues, and $I$ is the identity matrix. This objective effectively aims to find

a low-dimensional embedding that minimizes the weighted cut between clusters, as captured by the normalized graph Laplacian.

**Algorithm 4.4** Spectral Clustering

**Require:** A set of x-vectors $\{x_i\}$, number of clusters $K$
**Ensure:** Cluster assignments $T$
1: Construct similarity matrix $S$ using a similarity function (e.g., Gaussian kernel)
2: Compute graph Laplacian matrix $L = D - S$, where $D$ is the degree matrix of $S$
3: Perform eigenvalue decomposition of $L$ and select the top $K$ eigenvectors
4: Apply $K$-means clustering to the selected eigenvectors to obtain cluster assignments
5: **return** Cluster assignments $T$

**4.3.5. PCA with $K$-means.** *Principal Component Analysis (PCA)* with $K$-means clustering is a two-step dimensionality reduction and clustering technique. First, PCA, a technique with roots in [8], reduces the dimensionality of the x-vectors, denoted as $x_i$, by projecting them onto a lower-dimensional subspace while preserving the most significant variance in the data. This simplifies the subsequent clustering process. Then, $K$-means, as described in [4], is applied to the transformed data, resulting in a reduced feature matrix $Z$, to partition the x-vectors into clusters, represented by $C_k$. This combined approach can enhance the performance of $K$-means by mitigating the impact of noise and irrelevant features in the original x-vector data. The covariance matrix is denoted as $\mathbf{C}$, the matrix of eigenvectors from PCA is denoted as $V$, and the dimensionality of the reduced space is denoted as $d$.

While PCA itself aims to maximize variance, the $K$-means step has the objective of minimizing within-cluster variance. Combining these, the overall objective can be informally represented as:

$$\max_{V_d} \operatorname{Var}(XV_d) \quad \min_{\{C_k\},\{\mu_k\}} \sum_{k=1}^{K} \sum_{i \in C_k} \|z_i - \mu_k\|^2, \tag{4.6}$$

where $X$ is the matrix of x-vectors, $V_d$ is the matrix of the top $d$ eigenvectors, $z_i$ is the $i$-th row of the reduced feature matrix $Z$, and $\mu_k$ is the centroid of cluster $C_k$.

**Algorithm 4.5** PCA with $K$-means Clustering

**Require:** A set of x-vectors $\{x_i\}$, number of clusters $K$
**Ensure:** Cluster assignments $T$

1: Standardize the x-vectors by subtracting the mean and dividing by the standard deviation
2: Compute the covariance matrix $\mathbf{C}$ of the standardized x-vectors
3: Perform eigenvalue decomposition of $\mathbf{C}$ to obtain eigenvectors $V$
4: Project the standardized x-vectors onto the subspace spanned by the top $d$ eigenvectors to obtain reduced feature matrix $Z$
5: Apply $K$-means clustering to $Z$ to obtain cluster assignments
6: **return** Cluster assignments $T$

**5. Results.** In this section, we break down the results obtained from testing our two main datasets. First, we discuss the speaker diarization error calculation in the methodology we employed to calculate the most accurate error. We then address our preliminary tests on our MATLAB dataset and follow it up with our results on the more intricate ICSI corpus. Note that the results of both look significantly different due to differences in the datasets.

**5.1. Speaker Diarization Error Rate Calculation.** Speaker diarization involves partitioning audio recordings into regions corresponding to individual speakers. The *Diarization Error Rate (DER)* [12] evaluates performance by combining three components: miss error (speech incorrectly classified as non-speech), false alarm error (non-speech incorrectly classified as speech), and speaker error (speech regions attributed to the wrong speaker). Our algorithm mainly focuses on the *Speaker Error Rate (SER)*, as miss and false alarm errors are negligible in our experiments.

The speaker error rate quantifies the proportion of speech regions incorrectly assigned to clusters. To compute this, the unique true speaker labels, $l_{\text{true}}$, and detected cluster labels, $l_{\text{guess}}$, are identified. For each $l_{\text{true}}$, the $l_{\text{guess}}$ that minimizes mismatched regions is determined, defined as:

$$l^{\star}_{\text{guess}} = \arg\min_{l_{\text{guess}}} |S_{l_{\text{true}}} \setminus S_{l_{\text{guess}}}| \tag{5.1}$$

where $S_{l_{\text{true}}}$ and $S_{l_{\text{guess}}}$ represent the sets of speech regions for the true and guessed labels, respectively. $|\cdot|$ denotes the cardinality of a set. The total number of errors is computed by summing over all mismatched regions:

$$\text{TotalErrors} = \sum_{l_{\text{true}}} \min_{l_{\text{guess}}} |S_{l_{\text{true}}} \setminus S_{l_{\text{guess}}}| \tag{5.2}$$

The speaker error rate is then calculated as:

$$\text{SpeakerErrorRate} = \frac{\text{TotalErrors}}{\text{Total True Speech Regions}} \tag{5.3}$$

This approach evaluates speaker assignment accuracy independently of speech activity detection, aligning with established methodologies in [12].

**5.2. Preliminary Results.** We utilized various clustering methods to parse through audio recordings to distinguish individuals. Each clustering method exhibited varying levels of accuracy in speaker diarization, which will be shown below. $K$-Means clustering and PCA $K$-Means clustering emerged as the most effective approaches.

Specifically, it was found that the combination of PCA with $K$-Means clustering yielded the most compelling results in our experiments, both in speaker recognition and CPU usage. By reducing dimensionality and extracting essential speaker features, PCA enhanced the clustering process, leading to a quick and precise speaker diarization outcome. The synergy between PCA and $K$-Means not only improved accuracy but also enhanced the scalability of our speaker diarization system.

Below, we showcase representations of the results of each audio clustering algorithm. This analysis compares ground truth speaker labels, denoted by SP1 through SP5, against each algorithm's predictions, represented by the colored clusters in the legend and imposed upon the audio waves. This evaluation highlights how well the algorithms performed in segmenting speakers and identifies their strengths and weaknesses across various scenarios. Note that the color-coded legends exhibit corresponding numerics that at times do not match ground truth labels. Such errors were excluded from analysis as the focus was on ensuring each SP label occurred under their own unique colored cluster.

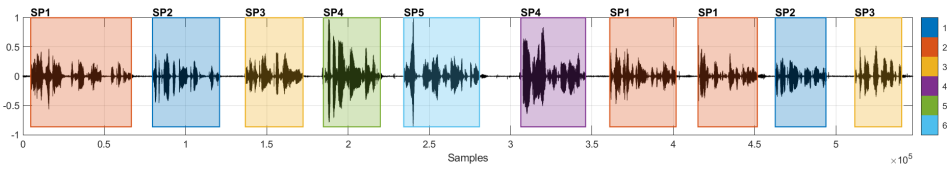


**Figure 2.** *Agglomerative - PLDA Scoring Speaker Recognition Results*

Agglomerative clustering results using PLDA scoring, as shown in Figure 2, demonstrated several shortcomings in accurately segmenting the five speakers (SP1 to SP5). One of the primary issues is the overestimation of speakers, as the model identifies six distinct clusters instead of five, indicating an over-segmentation problem. This additional cluster, represented in cyan, disrupts the intended alignment between the predictions and the ground truth.

While SP1, SP2, SP3, and SP5 are labeled incorrectly per the legend, all instances are clustered within the same label, indicating accurate clustering performance and recognition of the individual speakers. SP4's representation suffered from misclassifications, with it being considered two separate speakers in the two speaking instances. SP4 was classified as SP4 and with the cyan sixth cluster, or the non-existant SP6.

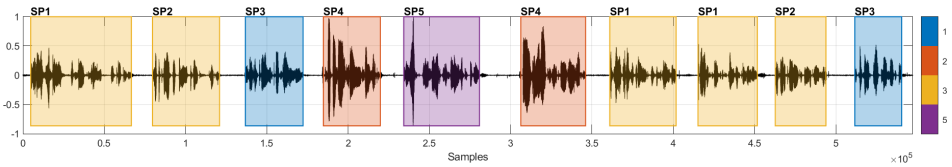


**Figure 3.** *Agglomerative - CSS Scoring Speaker Recognition Results*

Agglomerative clustering with cosine similarity scoring demonstrated reasonable success in capturing speaker boundaries and grouping acoustically similar segments. As seen in Figure 3, SP3, SP4, and SP5 regions align well with their respective predicted clusters, indicating that the algorithm effectively separates distinct speakers when their acoustic profiles differ significantly. However, challenges arise in scenarios with within-speaker variability or brief, less distinguishable audio segments. In clusters representing SP1's region, the parts are fragmented into multiple clusters, suggesting the algorithm struggles with consistent classification when acoustic features vary, such as shifts in tone or volume. Additionally, boundary confusion occurs where transitions between speakers are misclassified; SP2 is incorrectly assigned to SP1, highlighting difficulty in modeling seamless speaker transitions.

Overall, the agglomerative clustering with CSS scoring method shows promise in capturing broad speaker patterns but exhibits weaknesses in handling edge cases, such as overlapping or dynamically changing speaker characteristics.

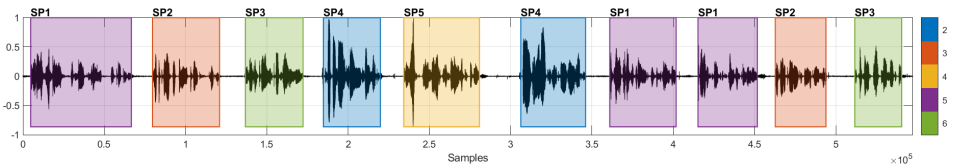


**Figure 4.** *K-Means Speaker Recognition Results*

The $K$-means clustering result in Figure 4 highlights several key observations and challenges in speaker diarization. The visualization illustrates six predicted clusters with only five clusters depicted, indicating a numbering error within the model cluster characterization step. Of the five existent speakers, all were found and while $K$-means identifies the speaker clusters perfectly, discrepancies emerge in its segmentation quality and alignment with ground truth.

The clustering process introduces an extra sixth cluster that does not correspond to any real speaker, and rather removes the first one, indicating only five were found, further complicating the interpretation. This situation likely results from noise or the inability of $K$-means to determine the correct number of speakers dynamically, causing algorithmic overcorrection. These issues reflect inherent limitations of $K$-means for speaker diarization tasks but also concerns with the ability of algorithms to consistently and dynamically label speakers. While its results depend heavily on feature separability, making it insensitive to speaker similarity and noise, labeling is a concern that is just as important in situations that drive change.

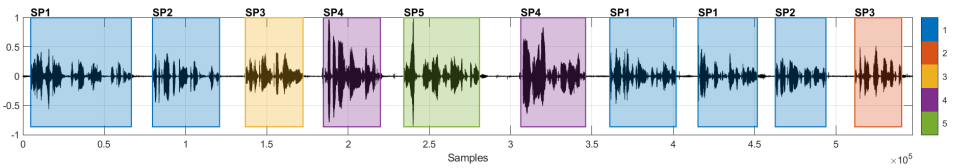


**Figure 5.** *Spectral Clustering Speaker Recognition Results*

Based on the spectral clustering results shown in Figure 5, the algorithm identifies five clusters corresponding to the speakers SP1–SP5. Spectral clustering turned out to be the

worst of our clustering algorithms, where there were very noticeable inaccuracies and points of confusion in the segmentation.

For SP1, the algorithm generally performed well at capturing speaker turns, especially at the start of the timeline. However, SP2 segments were misclassified as SP1 segments. SP4 and SP5 exhibited good alignment with their respective ground truths, but SP3 experienced the worst segmentation, with the speaker being classified as speakers two and three according to the legend. SP3 is the most problematic; segments expected to belong to SP3 are occasionally split across SP3 and SP4, suggesting difficulties in delineating cluster boundaries.

Overall, while spectral clustering achieves the accurate number of unique speaker regions, it struggles with boundary precision, particularly at speaker transitions, example shown with SP1 and SP2, and in cases of overlapping spectral features. This results in misclassifications and an inconsistent separation of clusters. These issues highlight the overall impracticality of specific clustering methods for speaker diarization.

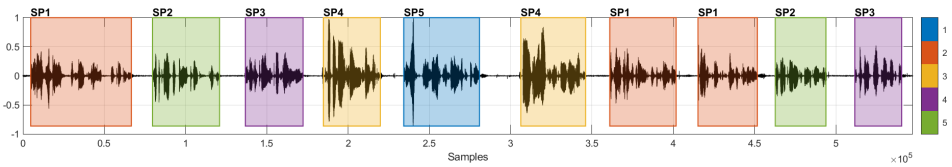


**Figure 6.** *PCA K-Means Speaker Recognition Results*

In analyzing Figure 6, it is demonstrated that $K$-means clustering algorithm with PCA dimensionality reduction identified five distinct speakers, SP1 through SP5, matching the given ground truths. Similar to $K$-means, the results indicated the usefulness of $K$-means clustering for speaker diarization, but the use of PCA removed the indexing and incorrect speaker number issues. The results showed some notable patterns in its clustering approach. The algorithm demonstrated a tendency to consistently identify SP1, which appears three times in the recording, suggesting the PCA features for this speaker are well-separated in the reduced dimensional space. Similar results are noted for speakers: SP2, SP3, and SP4. SP5, with only one contribution to the audio, is also properly characterized. While the algorithm handled amplitude variations well, the rigid cluster boundaries inherent to $K$-means appear to cause no issues following a dimensionality reduction.

**Table 1**
*Algorithm Performance Summary*

| **Algorithm** | **Speakers** | **Errors** | **Error Rate** | **CPU Time** |
|---|---|---|---|---|
| PLDA | 6 | 1 | 0.1 | 0.0776 |
| CSS | 4 | 5 | 0.5 | 0.0105 |
| $K$-Means | 5 | 0 | 0.0 | 0.0557 |
| Spectral Clustering | 5 | 6 | 0.6 | 0.1464 |
| PCA K-Means | 5 | 0 | 0.0 | 0.0343 |

As shown in Table 1, each clustering method demonstrates different performance characteristics. Below are all the numbers associated with each model and its respective run-

time. Error rate equations are found in subsection 5.1. Agglomerative clustering with PLDA achieved moderate performance with 6 speakers and 1 error, resulting in an error rate of 0.1. Its CPU time of 0.0776 seconds suggests that it is reasonably effective, though not the fastest in terms of computational efficiency. Agglomerative clustering with CSS showed a higher error rate of 0.5, with 4 speakers and 5 errors. However, its CPU time of 0.0105 seconds indicates high computational efficiency, making it a fast method, although its accuracy could be improved. $K$-means performed excellently with 5 speakers and 0 errors, resulting in an error rate of 0.0. It also exhibited a moderate CPU time of 0.0557 seconds, maintaining perfect accuracy while being computationally efficient. Spectral clustering had a higher error rate of 0.6, with 5 speakers and 6 errors. It took the longest CPU time of 0.1464 seconds, indicating that it is computationally more intensive, but it may offer more complex clustering capabilities. PCA with $K$-means also achieved an error rate of 0.0, with 5 speakers and no errors. It was the fastest method with a CPU time of 0.0343 seconds, demonstrating both high accuracy and efficiency.

**5.3. ICSI Corpus Results.** For the evaluation of the ICSI corpus, we utilized all of the same clustering methods, including agglomerative clustering with CSS scoring, $K$-means, spectral clustering, and PCA $K$-means. Our PLDA method used on the MATLAB dataset was excluded from consideration due to its prohibitive CPU usage, which made it unsuitable for efficient processing across multiple large datasets. We therefore concluded that PLDA is only useful for small test runs and can be excluded from usability for the broader applications of speaker diarization.

Figure 7 through Figure 14 present the results of the most unique audio file results, particularly with Bdb001, Bed005, Bed008, Bed009, Bed014, Bed015, Bmr010, and Bmr013. For each figure, the left-hand graphic shows the CPU time for each method, while the right-hand side illustrates the corresponding error rates for our methods. These results provide a comprehensive view of the performance of each clustering approach, highlighting their computational efficiency and accuracy across different audio files. The other tests were excluded for redundancy, but a complete set of results is available in our GitHub repository, as referenced in the introduction, within the /results/ICSI/figs/ folder.

The error and CPU usage analysis across all test cases reveals consistent patterns and key distinctions among agglomerative clustering with CSS scoring, spectral clustering, $K$-means, and PCA $K$-means methods.

In Bed001, Figure 7, CSS scoring demonstrated a gradual and steady increase in CPU usage while maintaining relatively low utilization overall. In contrast, spectral clustering exhibited a sharp and continuous rise in CPU consumption, indicating significant computational demands. Both $K$-means and PCA $K$-means operated with nearly constant, low CPU usage, with PCA $K$-means consuming slightly less CPU overall. Error rates revealed that agglomerative clustering with CSS scoring fluctuated erratically over time, particularly for longer audio files, while spectral clustering presented frequent large spikes and periods of stability. $K$-means and PCA $K$-means displayed much more consistent and lower error rates, often aligning closely.

For Bed005, Figure 8, CSS scoring continued to show low CPU usage with a slow, linear increase. Spectral clustering followed a similar trend to Bed001, with progressively higher

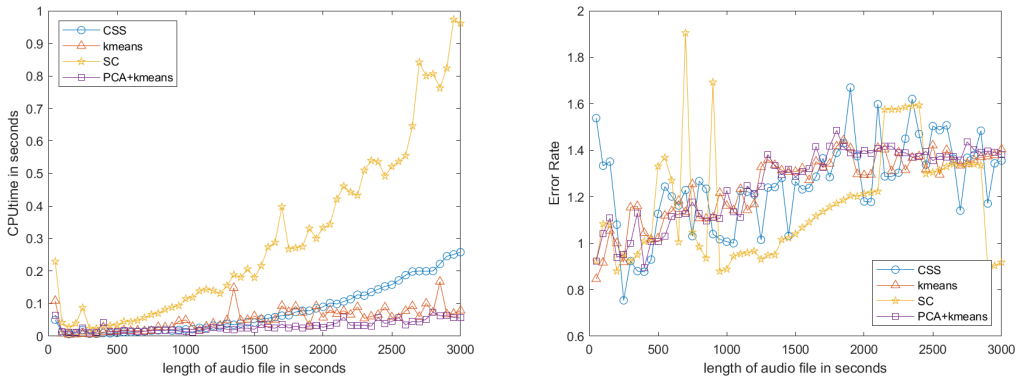


**Figure 7.** *CPU time and Error Rate for the audio file Bdb001.wav*

usage, reflecting its computationally intensive nature. $K$-means and PCA $K$-means demonstrated near-zero CPU consumption, punctuated by minor spikes. Error rates were more erratic with CSS scoring and spectral clustering fluctuated within a range of 0.8 to 1.6, with spectral clustering at times achieving greater stability. However, $K$-means demonstrated the most variable error rates, reaching the highest peaks, often accompanied by PCA $K$-means following a similar trend but slightly lower.

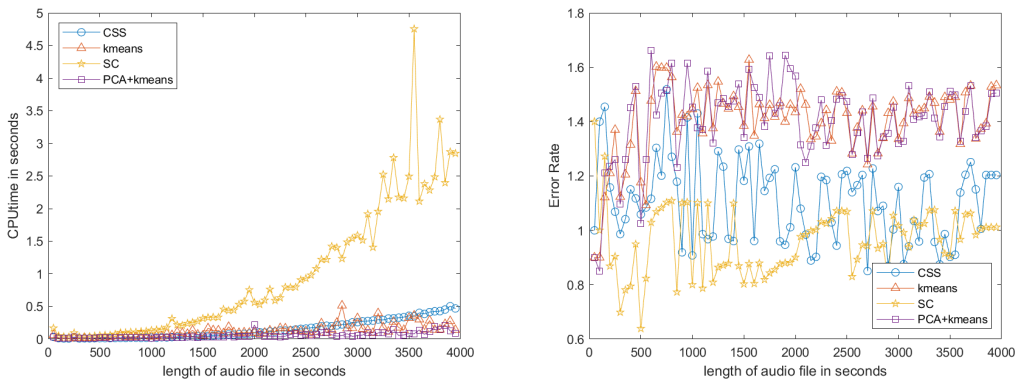


**Figure 8.** *CPU time and Error Rate for the audio file Bed005.wav*

In Bed008, Figure 9, CPU usage for CSS scoring showed a slight linear increase, while spectral clustering again demonstrated drastic upward trends. Both $K$-means and PCA $K$-means remained relatively constant, with PCA consuming the least CPU resources. Agglomerative clustering with CSS scoring error rates were highly variable and increased over time, whereas spectral clustering displayed less drastic fluctuations but mirrored the trend. $K$-means and PCA $K$-means aligned closely, presenting stable, lower error rates compared to CSS scoring and spectral clustering.

For Bed009, Figure 10, CSS scoring exhibited a linear increase in CPU usage with generally low overall utilization. Spectral clustering, however, demonstrated its characteristic drastic

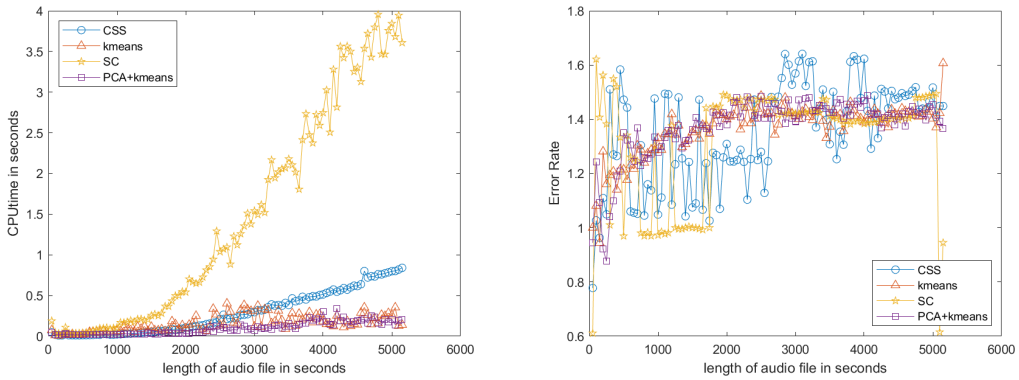


**Figure 9.** *CPU time and Error Rate for the audio file Bed008.wav*

spikes in CPU consumption. $K$-means displayed periodic increases, with occasional spikes, while PCA $K$-means remained low and consistent. Error rates for CSS scoring and spectral clustering were erratic, with frequent large spikes and increases over time. $K$-means and PCA $K$-means continued to reflect each other's error rates, maintaining medium, consistent errors that were lower than the other methods.

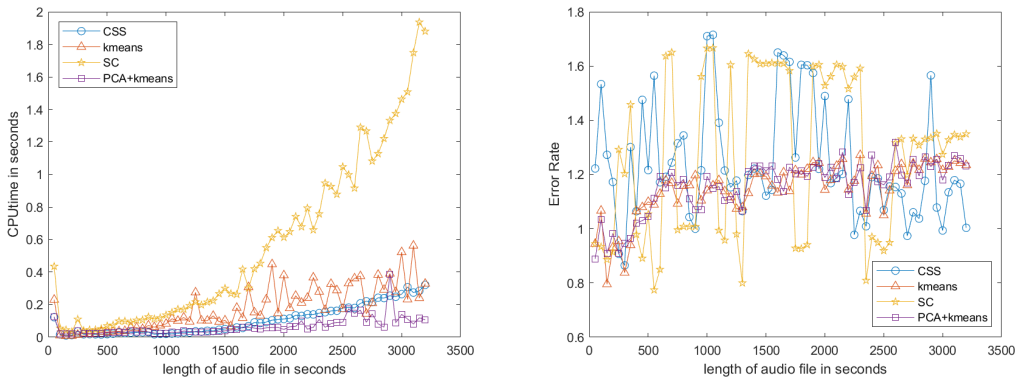


**Figure 10.** *CPU time and Error Rate for the audio file Bed009.wav*

In Bed014, Figure 11, CSS scoring and spectral clustering demonstrated distinct CPU usage trends. CSS scoring exhibited a gradual decline in CPU usage over time, while spectral clustering's resource demands were less drastic compared to other cases and showed periods of drops in usage. $K$-means CPU usage increased over time, with occasional spikes, whereas PCA $K$-means remained constant with minimal fluctuations. Error rates across all methods were very similar, with spectral clustering occasionally producing high spikes. Interestingly, PCA $K$-means displayed a unique dip near the end, differentiating it from the other methods, despite no distinct difference in audio data.

For Bed015, Figure 12, CSS scoring CPU usage increased gradually but remained relatively low. Spectral clustering exhibited drastic and sharp increases in CPU consumption. $K$-means

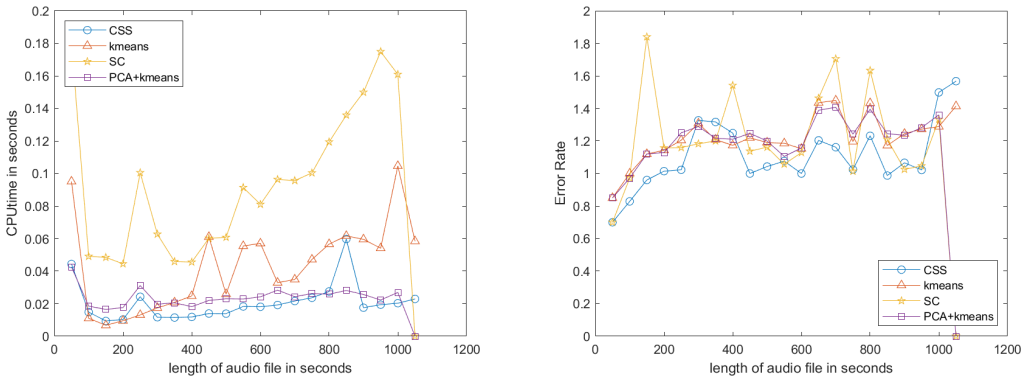


**Figure 11.** *CPU time and Error Rate for the audio file Bed014.wav*

displayed slightly more erratic behavior compared to previous cases, with periodic spikes, while PCA $K$-means maintained its characteristic low, near-constant CPU usage. In terms of error, CSS scoring and spectral clustering were highly erratic, with large spikes that reached the highest recorded error values. $K$-means and PCA $K$-means maintained steady error trends, with PCA occasionally achieving the lowest error rates overall. Generally, all methods had very similar error trends.

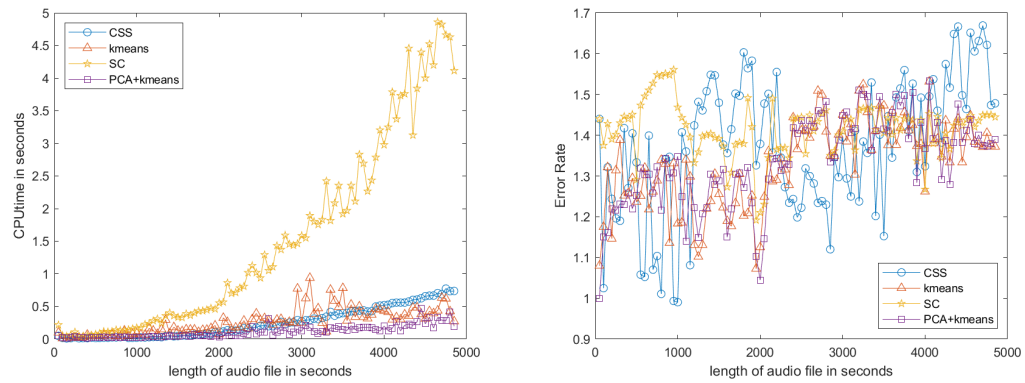


**Figure 12.** *CPU time and Error Rate for the audio file Bed015.wav*

In Bmr010, Figure 13, CSS scoring's CPU usage showed a higher overall trend compared to previous cases, with consistent linear growth. Spectral clustering continued to display sharp increases, while $K$-means and PCA $K$-means remained constant and efficient, consuming minimal CPU resources. Error rates for CSS scoring were persistently high, with frequent dips and peaks. Spectral clustering error rates remained more stable, with occasional dips and smaller spikes. $K$-means and PCA $K$-means maintained low, steady error rates around the 1.3 error mark, demonstrating their reliability.

Finally, for Bmr013, Figure 14, agglomerative clustering with CSS scoring's CPU usage exhibited a progressive increase over time, while spectral clustering retained its usual sharp

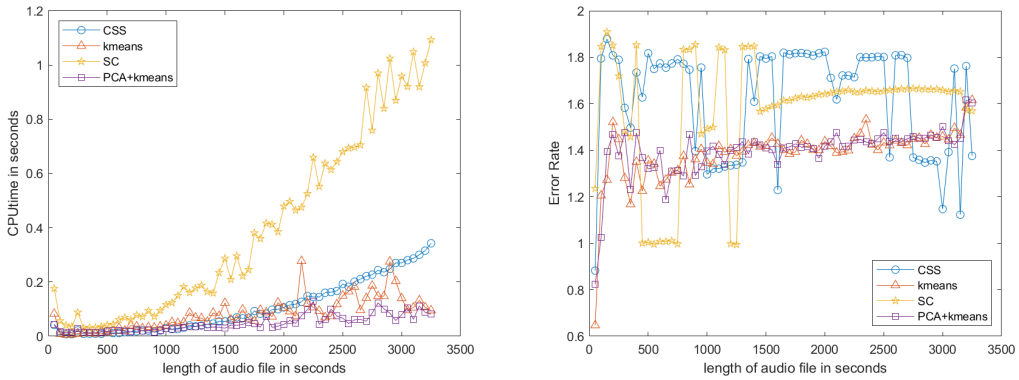


**Figure 13.** *CPU time and Error Rate for the audio file Bmr010.wav*

upward trend. Notably, $K$-means CPU usage featured large spikes, deviating from its previous constant behavior. PCA $K$-means remained stable with minimal resource demands. CSS scoring error rates were the highest, displaying frequent spikes and increasing trends. Uniquely, spectral clustering demonstrated the lowest error rates overall, with stability punctuated by minor dips and small spikes. $K$-means and PCA $K$-means followed their characteristic patterns, maintaining consistent error rates that closely aligned.

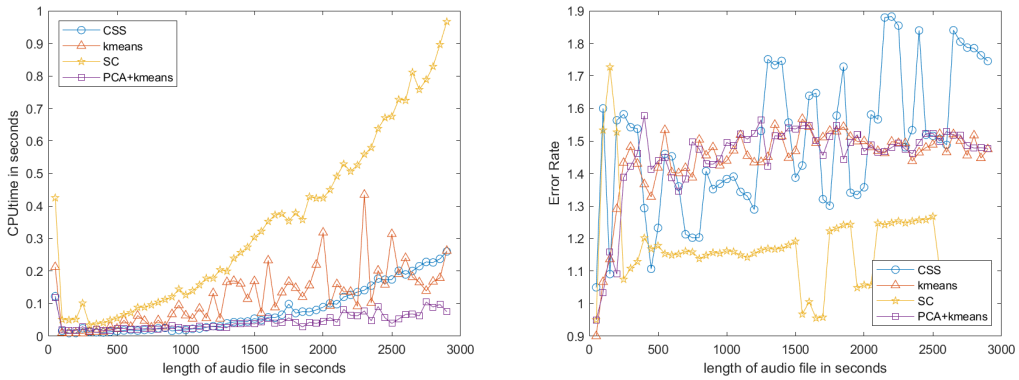


**Figure 14.** *CPU time and Error Rate for the audio file Bmr013.wav*

To summarize these chosen tests, agglomerative clustering with CSS scoring exhibited the most variable and often highest error rates across all cases, particularly for longer audio files, while spectral clustering displayed erratic yet occasionally stable behavior. $K$-means and PCA $K$-means consistently aligned in both error rates and CPU usage, demonstrating lower variability and computational cost. Spectral clustering showed the most drastic CPU increases, while PCA $K$-means maintained consistently minimal usage across all test cases. These findings highlight the trade-offs between computational efficiency and accuracy, particularly for resource-constrained environments.

**6. Discussion.** In our analysis with the MATLAB dataset, PCA $K$-means clustering proved the most effective for speaker diarization, combining dimensionality reduction with clustering to improve accuracy and computational efficiency. Spectral clustering, on the other hand, struggled with boundary precision, leading to misclassifications. While $K$-means clustering showed potential, it faced challenges with over-segmentation and noise. Agglomerative clustering had moderate success but struggled with complex speaker transitions. Overall, PCA $K$-means demonstrated the best performance, suggesting that combining dimensionality reduction with clustering enhances speaker diarization accuracy and scalability.

When evaluating the performance of our clustering methods with the ICSI corpus, spectral clustering proved to be the most time-consuming algorithm. It consistently required higher CPU time as it processed the data, particularly as the model continued to iterate over larger datasets. This trend suggests that spectral clustering's complexity and reliance on pairwise similarity matrices contribute to its slower performance. Conversely, $K$-means showed variability in its performance depending on the problem, but it was generally not as efficient as PCA $K$-means. The latter demonstrated a very consistent performance profile with minimal fluctuations in CPU usage, which likely contributed to its more efficient execution. Agglomerative clustering with CSS scoring, while providing a smooth CPU time curve, never surpassed PCA $K$-means and exhibited a steady, linear increase in processing time across all tests. This suggests that CSS, though stable, lacks the optimization seen in PCA $K$-means, where dimensionality reduction leads to more efficient processing.

Regarding accuracy, agglomerative clustering with CSS scoring and spectral clustering exhibited the most volatile performance. Spectral clustering, in particular, showed significant accuracy fluctuations, with error rates spiking and dropping unpredictably. These inconsistencies may stem from the algorithm's sensitivity to noise and the underlying data distribution, leading to erratic speaker segmentation. CSS scoring also struggled with drastic inflection points in accuracy, reflecting similar instability in its segmentation results. On the other hand, both $K$-means and PCA $K$-means demonstrated much more stable accuracy levels, rarely deviating significantly in their error rates. This consistency is a strength, as it indicates these methods are less affected by small variations in the data and provide more reliable performance. PCA $K$-means, in particular, showed a slight improvement over $K$-means, which can be attributed to the PCA step, where dimensionality reduction helps filter out irrelevant features, thus improving both computational efficiency and accuracy.

While spectral clustering and CSS scoring occasionally produced lower error rates, their performance inconsistencies and high CPU time costs make them less reliable choices for practical applications. Interestingly, although spectral clustering sometimes achieved the lowest error rates, it also displayed the highest error rates in almost all the tests, occasionally being outperformed by CSS scoring. Over time, however, spectral clustering did show improvements, with its error rate becoming more linear as processing progressed, suggesting that it may be more adaptable in long-running tasks. Nonetheless, its overall performance remains less predictable compared to $K$-means and PCA $K$-means, which combines accuracy with stable computational efficiency.

**7. Future Work.** Future research could expand on our findings by testing our models with live data streams from diverse sources such as Zoom meetings, Netflix movies, and podcasts,

all of which would be primary utilizations for speaker diarization algorithms. This would help assess the scalability and adaptability of the algorithms to real-world conditions, including variations in data quality and noise. A promising direction would be to train the Convolutional Neural Networks on live data, experimenting with tuning techniques such as adding more layers or increasing the number of epochs to improve training accuracy. This could enhance the robustness of the models and provide more precise, real-time results for dynamic environments. Additionally, further exploration into hybrid methods combining dimensionality reduction techniques like PCA with deep learning could potentially optimize both speed and accuracy across larger datasets, making the models even more suitable for large-scale and real-world applications.

**8. Conclusions.** We present this research as a cutting-edge solution to current speaker diarization applications. Our research explored the performance of several cutting edge clustering algorithms, with a primary focus on CPU usage and accuracy calculated based on Diarization Error Rate when handling diverse datasets. Our findings suggest that while spectral clustering and CSS scoring showed considerable fluctuation in performance and computational time, PCA $K$-means stood out for its consistency and efficiency, delivering reliable results across varying conditions with minimal CPU usage. Overall, the study underscores the importance of balancing accuracy with efficiency, particularly when handling large datasets, and sets the stage for further refinements in clustering techniques for more dynamic, real-world applications. We expect more research to delve into fine-tuning neural networks and testing with machine learning models such as NLPs and transformers.

We present this research as a cutting-edge solution to current speaker diarization applications. Our research explored the performance of several cutting edge clustering algorithms, with a primary focus on CPU usage and accuracy calculated based on Diarization Error Rate when handling diverse datasets. Our findings suggest that while spectral clustering and CSS scoring showed considerable fluctuation in performance and computational time, PCA $K$-means stood out for its consistency and efficiency, delivering reliable results across varying conditions with minimal CPU usage. Overall, the study underscores the importance of balancing accuracy with efficiency, particularly when handling large datasets, and sets the stage for further refinements in clustering techniques for more dynamic, real-world applications. We expect more research to delve into fine-tuning neural networks and testing with machine learning models including *Natural Language Processing (NLP)*, Transformers, and *Large Language Models (LLM)*.

## REFERENCES


[1] X. Anguera, *Speaker diarization: A review of recent research*, IEEE Transactions on Audio, Speech, and Language Processing, 20 (2012), pp. 356–370.

[2] A. Gomizelj, *A Netflix original closed captioning study: How Netflix closed captions make audiovisual content accessible to deaf audiences*, PhD thesis, Universite d'Ottawa / University of Ottawa, 2022.

[3] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, et al., *The icsi meeting corpus*, in 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03)., vol. 1, IEEE, 2003, pp. I–364.

[4] J. B. MacQueen, *Some methods for classification and analysis of multivariate observations*, Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1 (1967), pp. 281–297.

[5] P. Matejka, O. Glembek, F. Castaldo, M. Alam, O. Plchot, P. Kenny, L. Burget, and J. ernocky, *Full-covariance ubm and heavy-tailed plda in i-vector speaker verification*, in 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2011, pp. 4628–4631.

[6] Mathworks, *Speaker diarization using x-vectors*, https://www.mathworks.com/help/audio/ug/speaker-diarization-using-x-vectors.html#mw_rtc_SpeakerDiarizationUsingXVectorsExample_DB133825.

[7] Netflix, *Introducing netflix timed text authoring lineage.* https://netflixtechblog.com/introducing-netflix-timed-text-authoring-lineage-6fb57b72ad41, 2023. Accessed: 2024-11-26.

[8] K. Pearson, *Liii. on lines and planes of closest fit to systems of points in space*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2 (1901), pp. 559–572.

[9] O. G. Ploscaru, P. S. Popescu, M. C. Mihaescu, S. Heras, and V. Julian, *Detection of topics from video transcripts by ml/dl techniques*, in 2024 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), 2024, pp. 1–6, https://doi.org/10.1109/INISTA62901.2024.10683753.

[10] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, *Speaker identification and verification using gaussian mixture speaker models*, in Speech communication, vol. 31, Elsevier, 2000, pp. 173–190.

[11] G. Sell, D. Snyder, A. McCree, D. Garcia-Romero, J. Villalba, M. Maciejewski, V. Manohar, N. Dehak, D. Povey, S. Watanabe, and S. Khudanpur, *Diarization is hard: Some experiences and lessons learned for the jhu team in the inaugural dihard challenge*, in Proc. Interspeech 2018, 2018, pp. 2808–2812, https://doi.org/10.21437/Interspeech.2018-1893.

[12] G. Sell, D. Snyder, A. McCree, D. Garcia-Romero, J. Villalba, M. Maciejewski, V. Manohar, N. Dehak, D. Povey, S. Watanabe, and S. Khudanpur, *Diarization is hard: Some experiences and lessons learned for the jhu team in the inaugural dihard challenge*, in Proc. Interspeech 2018, 2018, pp. 2808–2812, https://doi.org/10.21437/Interspeech.2018-1893.

[13] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Transactions on pattern analysis and machine intelligence, 22 (2000), pp. 888–905.

[14] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, *X-vectors: Robust dnn embeddings for speaker recognition*, in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 5329–5333.

[15] R. R. Sokal and C. D. Michener, *A statistical method for evaluating systematic relationships*, University of Kansas science bulletin, 38 (1958), pp. 1409–1438.

[16] U. Von Luxburg, *A tutorial on spectral clustering*, Statistics and computing, 17 (2007), pp. 395–416.

[17] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, *Speaker diarization with lstm*, arXiv preprint arXiv:1806.07630, (2018).