Spoken Language Recognition at LSV

Programming Challenge

Report on Phoneme Vector Similarity, Clustering & Dimensionality Reduction

Date of Submission: 05-09-2020

Akshay Joshi
s8akjosh@stud.uni-saarland.de
M. Sc. Data Science & Artificial Intelligence

I. Project Code Repository & Task Visualizations

1. Code Repository: GitHub
2. Dedicated Task Website: Link

II. Introduction

Accurate Grapheme-to-Phoneme conversion is crucial for the effectiveness and success of Automatic Speech Recognition and Text-to-Speech systems. This task may perhaps be particularly challenging when building multilingual ASR agents which requires un-ambiguous (without being affected by different accents/acoustic domains/similar sounding words in analogous languages ex: Kannada & Sanskrit) conversion of textual words to phonemes.

A typical G2P process has 3 steps:

- Aligning Grapheme token to Phoneme token
- Learning the G to P conversion (neural/statistical methods)
- Triaging the best possible pronunciation provided the model

In this programming assignment, the task is to import and efficiently parse the tsv file which contains 50 phonemes (perhaps retrieved from a CNN/Transformer based neural model for grapheme-to-phoneme conversion) & their corresponding representations in the embedding space as a 236-dimensional vector. Further, to assess if these phonemes are similar or correlated to each other in terms of usage/semantics/audio signature, we perform an array of tasks ranging from calculating the Pairwise Cosine Similarities, Dimensionality Reduction using Linear & Manifold Learning methods and Clustering to uncover hidden patterns in the phoneme vector space.

III. Tasks Implemented

1. Pairwise Cosine Similarity Heat map/Confusion Matrix
2. Hierarchical Clustering & Dendrogram Visualization
3. Principle Component Analysis (PCA)
4. Independent Component Analysis (ICA)
5. t-Distributed Stochastic Neighbor Embedding (t-SNE)
6. Multidimensional Scaling (MDS - Metric)
7. PCA - DBSCAN Clustering

A. Task 1: Pairwise Cosine Similarity Confusion Matrix

Before we proceed with the computation of cosine similarities of phoneme vectors, it is ideal to learn about what are vector embeddings of words/phonemes.

- **Phoneme Embeddings:** It is a learned representation of a phoneme (group of sounds in a language) in the form of a real-valued vector in the vector space. The phoneme could be represented in a vector of multiple hundred dimensions (when working with large corpus).
- **Phoneme Embedding Similarity:** Just like word vectors, phonemes with similar audio signature/semantics/meaning/usage have similar representation in the Vector Space Model (VSM). Provided there is no domain mismatch or contrasting acoustic conditions.
- **Dimensionality of Embeddings:** Processing low dimensional vectors is computationally less intensive and well optimized in popular machine learning frameworks such as Scikit-learn.

In this task, I have built a phoneme dictionary whose keys are populated with the phoneme symbols and values as their vectors. Pandas Data Frames were not used as the task was simple and did not require any data imputations (no missing values in the vectors). Then, utilized Cosine Similarity function from the 'sklearn metrics' package to measure the cosine of the angle between the two phoneme vectors in vector space. It is to be noted that, **smaller** is the angle, **higher** is the vector similarity.

The mathematical equation of Cosine Similarity is:

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\cdot\|\vec{b}\|}$$

The confusion matrix generated from my code is as follows:

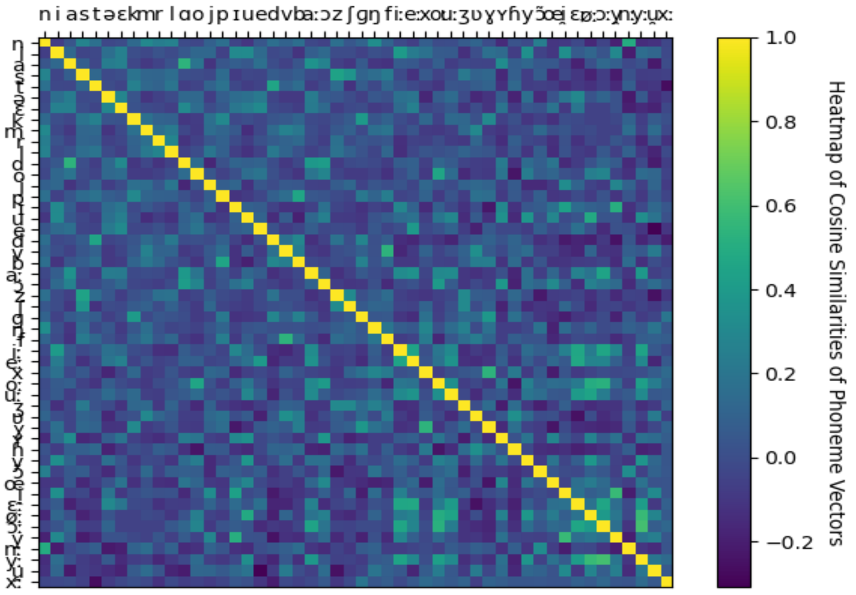


Figure: Pairwise Cosine Similarity between Phoneme Vectors

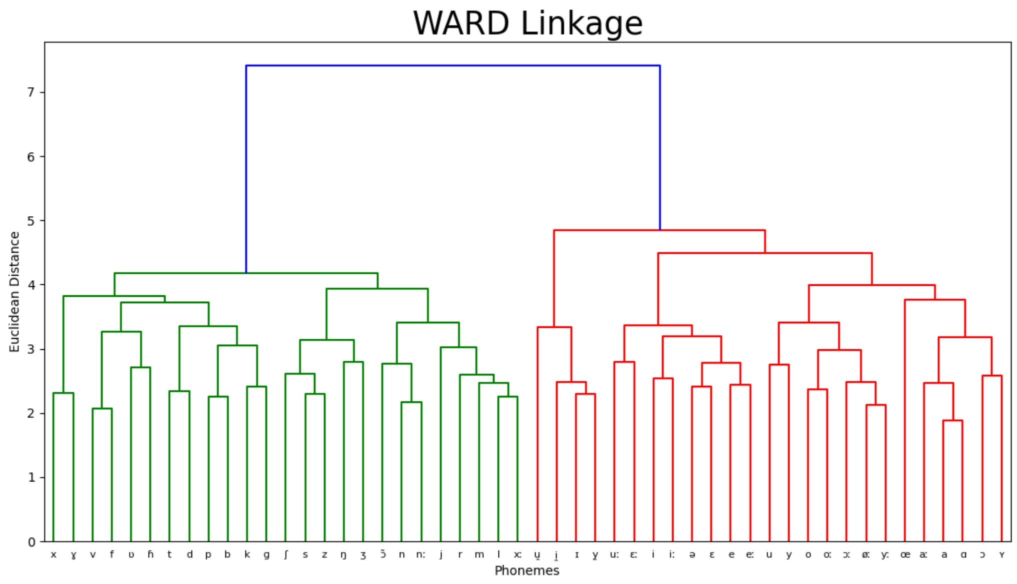
[Please find the higher resolution images of the plots in the 'Images' folder in the submission]

- In the above plot, the bright yellow spot between 2 phonemes signifies higher similarity. Whereas, the dark purple region signifies that the corresponding phonemes are least similar to each other.
- Example: The phonemes 'n' & 'n' are highly similar to each other. Whereas, 'n:' & 'n' are moderately similar and the symbols 'x:' &'t' are completely dissimilar.

B. Task 2: Hierarchical Clustering & Dendrogram Visualization

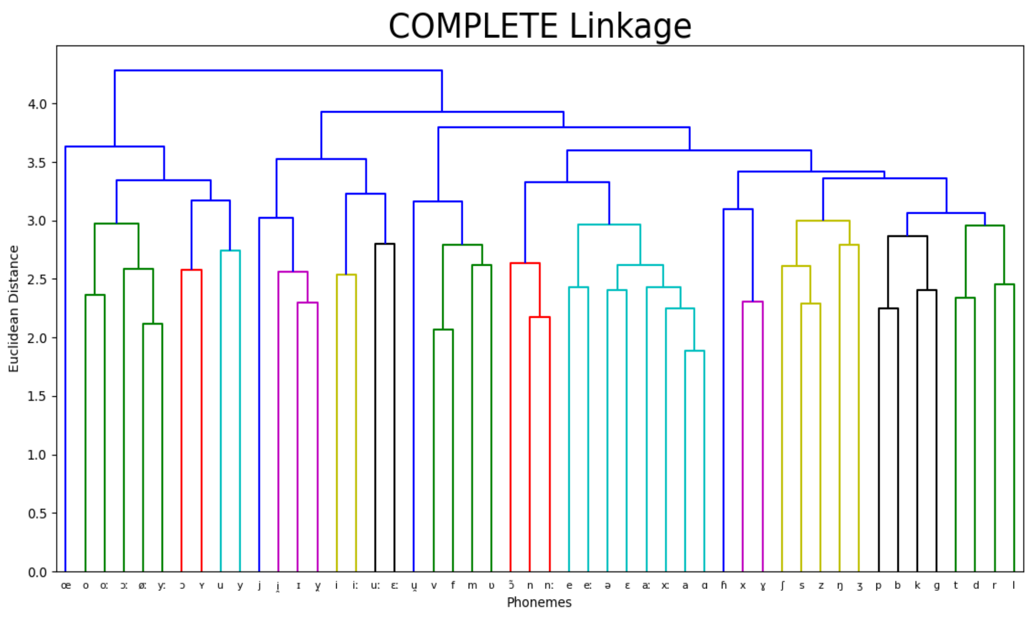
In this task, I have implemented Agglomerative clustering methods to cluster the phonemes in a bottom-up approach.
I have experimented with 4 different linkage criteria, please find the details below:

1. **Ward Linkage:** This method minimizes the sum of squared differences within all clusters.

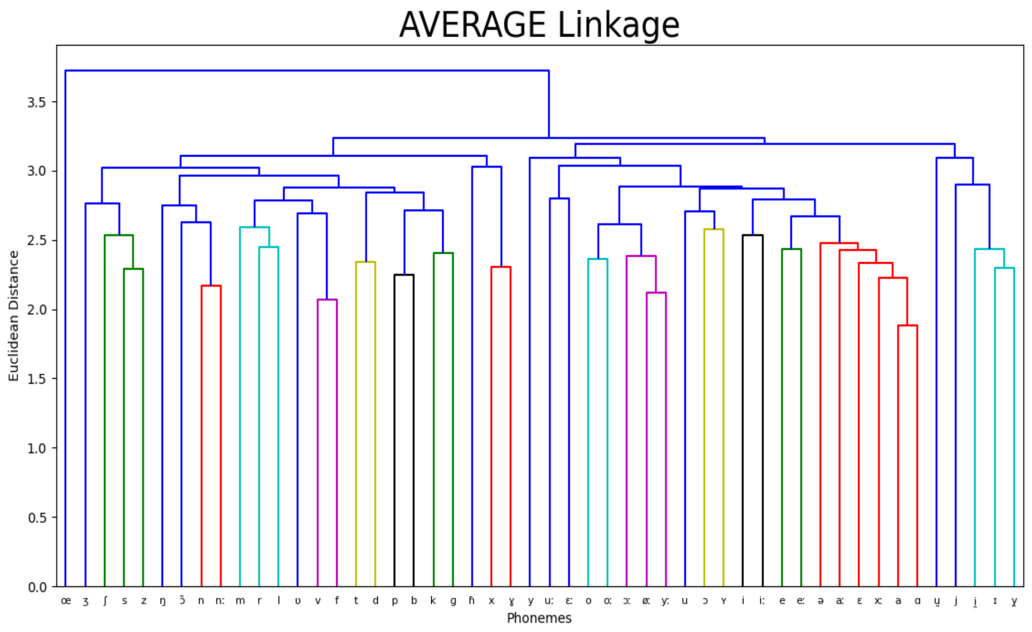


As you can see, 'ɑ' & 'a' are initially clustered together as they are similar to each other (ref: confusion matrix) and later clustered with 'a:' which is similar to both the phonemes. The process continues until 2 clean clusters of phonemes are obtained (which is combined at last).

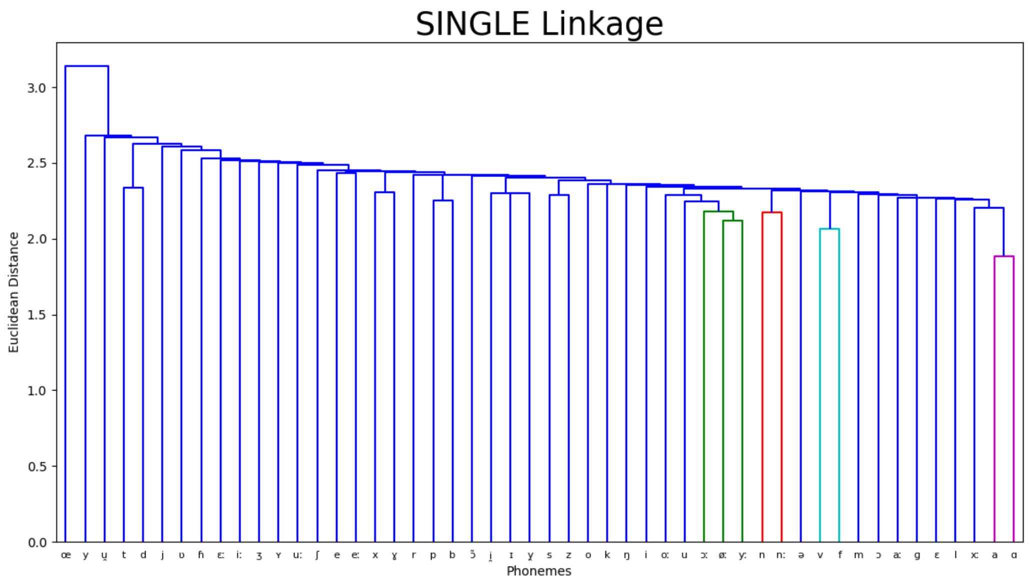
2. **Complete Linkage:** Minimize the maximum distance between observations in pairs of clusters. This method performs slightly worse than Ward as it clusters dissimilar phonemes together over slightly similar ones. Example: 'x:' is clustered with 'ɑ' & 'a' rather than 'a:'. This leads to less crisp clusters and possibility of adding unrelated phonemes.



3. **Average Linkage:** Minimize the average of the distances between all observations among pairs of clusters. Performs on-par with Complete Linkage but initially creates more local and granular clusters as compared to the later criteria. Whereas, in Ward's, all the phonemes are sorted within the context of 2 global clusters. Also, generates clusters with slightly uneven sizes.



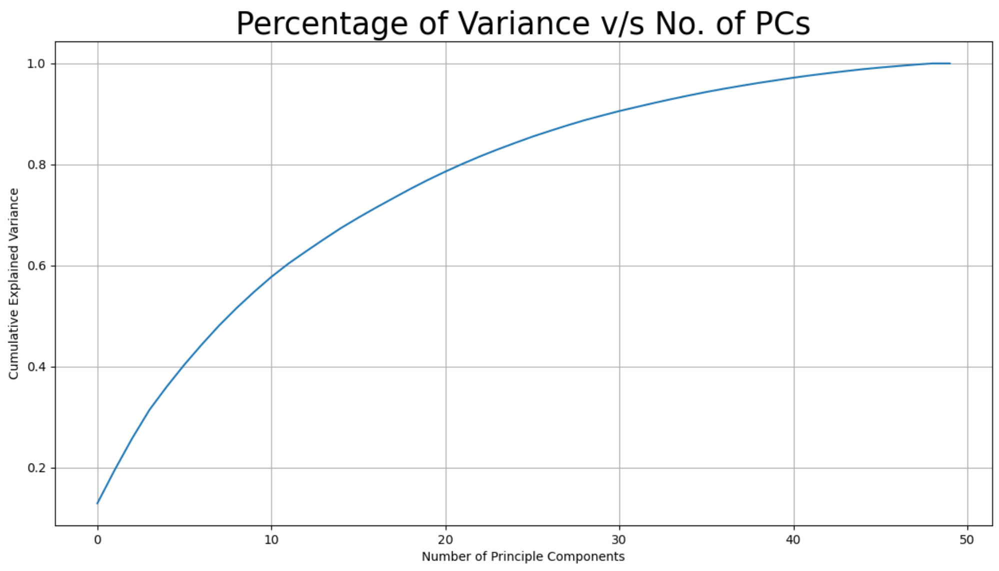
4. **Single Linkage:** Minimize the distance between the closest observations in pair of clusters. Doesn't provide any valuable information when compared to the later methods. Local cluster w.r.t certain patterns are not obtained.



C. Task 3: Dimensionality Reduction with Principal Component Analysis

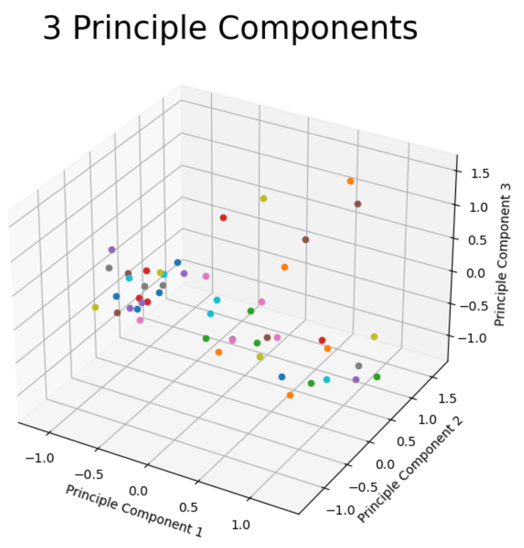
In this task, the effort is to reduce the number of input features (phonemes) to a lower dimension to decrease the input data size and avoid the curse of dimensionality using PCA and SVD. PCA works well only when the input features are linearly correlated to each other to perform the Eigen value decomposition.

1. Explained Cumulative Variance:



As the phoneme vectors (features) are not highly correlated to each other (linear), we approximately need 35 principal components to explain 90% of the variance in the data.

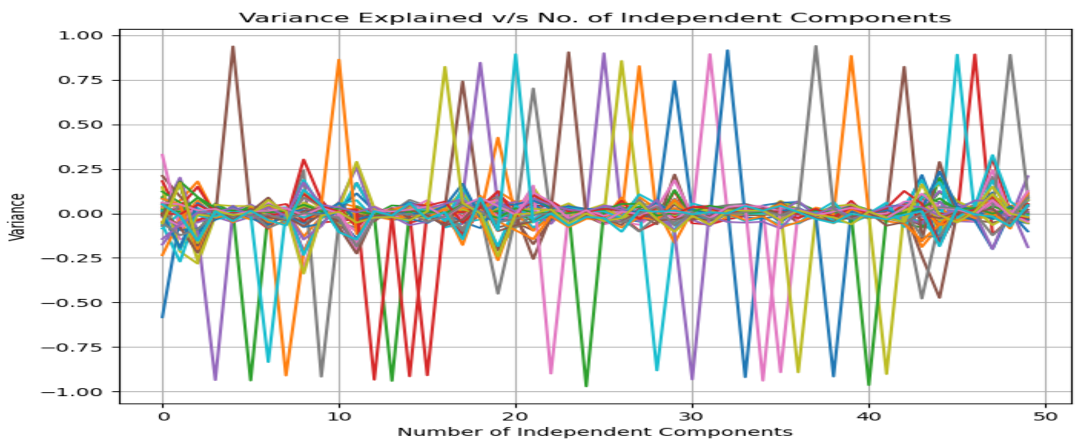
2. **Variance Explained with 3 Principal Components:**



- The Explained Variance Ratio by 3 Principal Components is: **25.77161032886719 %**
- This low variance coverage is evident from the plot in 3 dimensional space.

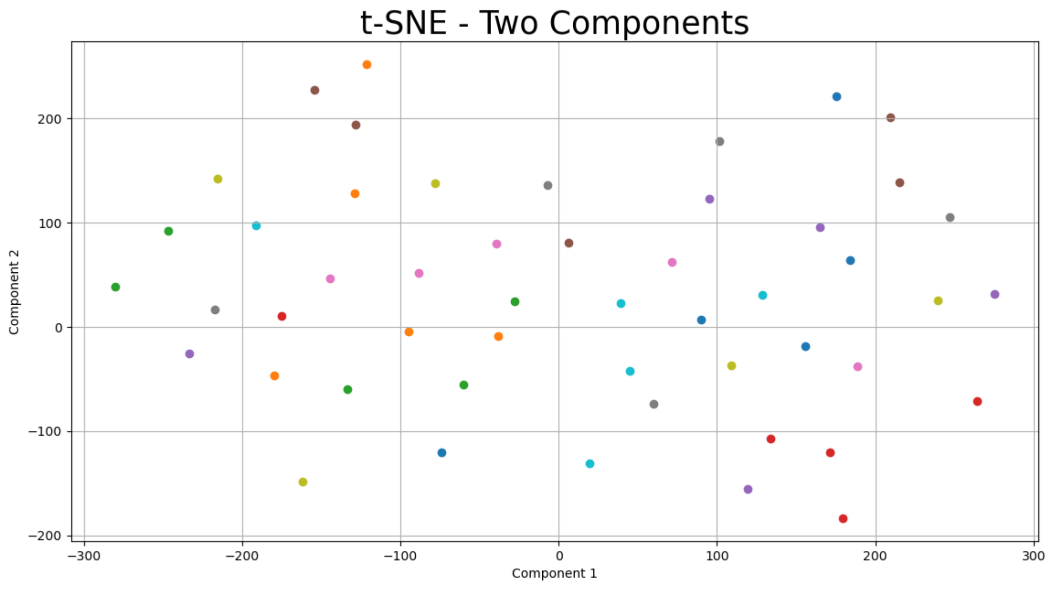
D. Task 4: Independent Component Analysis (ICA)

As there appears to be less correlation among the features in the provided data, PCA would not be a very effective technique for dimensionality reduction. Thus, I intend to explore the effectiveness of ICA as a method to separate/distinguish the features as a separate vector for each independent signal considering the data as mix of signals. With the number of independent components set to 40, it appears to achieve the best possible separation among the signals.



E. Task 5: t-Distributed Stochastic Neighbor Embedding (t-SNE)

To handle the possible non-linearity in the data, I have experimented with t-SNE for dimensionality reduction. This method also preserves local cluster information as compared to global structure in PCA.



From the above plot, it is quite evident that the local structure information is preserved.

1. The following hyperparameter values were chosen after an exhaustive grid search:

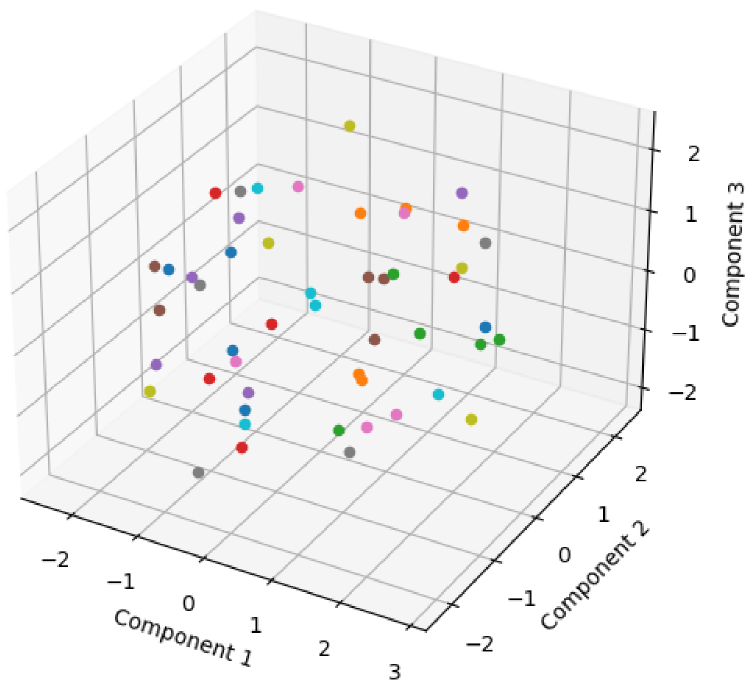
No. of Components	Perplexity	Learning Rate	Iterations
2	17	300	1000
3	37	250 - 300	1000

2. Results & Challenges faced:
   - KL divergence after 250 iterations with early exaggeration: **67.320938**
   - KL divergence after 1000 iterations: **0.470928**
   - Since the algorithm is randomized/non-deterministic (no guaranteed global maxima), the results keep fluctuating a lot!
   - Cannot preserve variance but instead preserve distance in the data

F. Task 6: Multidimensional Scaling (MDS – Metric)

Apart from t-SNE, I have also tried dimensionality reduction using a non-stochastic Manifold Learning method called MDS (Metric). It is already known that MDS works well only if the data is from a small distribution and not a random sample.

MDS - Two Components



The result is almost similar to PCA as Metric MDS with Euclidean distance (used in this task) is equivalent to vanilla Principal Component Analysis and the features may perhaps not be from a tight sample.

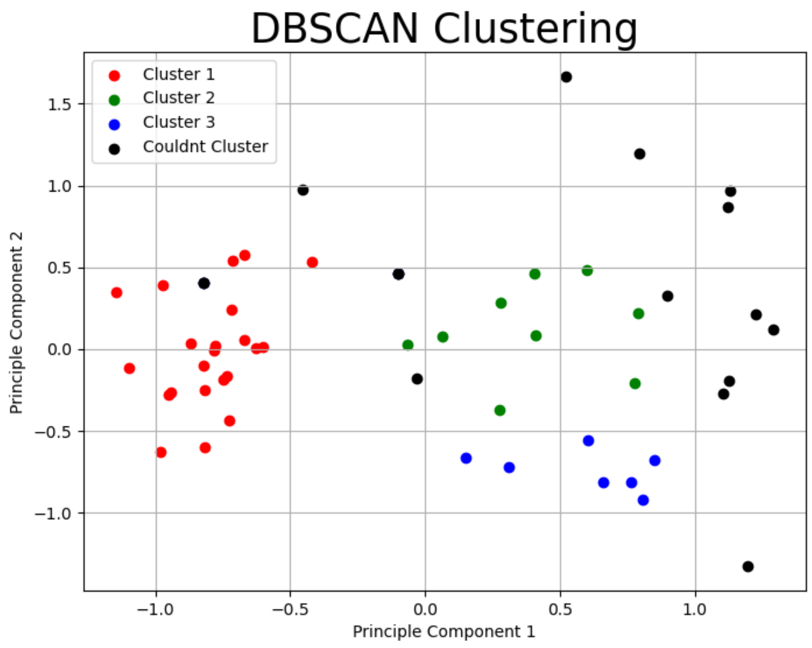
G. Task 7: PCA – DBSCAN Clustering

Finally, experimented with Density Based Spatial Clustering over dimensionality reduced data using PCA (2 PCs) to achieve non-linearly separable clusters as compared to traditional methods such as Gaussian Mixture EM and K-means clustering.

- The following hyperparameter values were chosen after an exhaustive grid search:

eps	Min Samples	Metric
0.5	9	Euclidean

- The method is performing better than the previous agglomerative methods.
- Is able to cleanly separate similar phonemes into their corresponding clusters
- Robust to outliers



H. Conclusion

A state-of-the-art deep neural network for ASR/Grapheme-to-Phoneme conversion must efficiently grasp the semantics, syntax & usage of words in relatively analogous languages and multiple speech accents of the same language in all acoustic conditions to discriminate and classify similar sounding phonemes into corresponding clusters/reduce the pairwise distance in the vector space. Also, there appears to be moderate amounts of similarity among phoneme vector pairs in the right bottom corner of the heat map. Finally, it is onerous to comment further on the effectiveness of similarity/dissimilarity of phonemes in the provided task without the additional context of the problem and the type of neural model used or the languages processed for G2P conversion.

I. References

1. Yolchuyeva, Sevinj et al. “Grapheme-to-Phoneme Conversion with Convolutional Neural Networks.” Applied Sciences 9 (2019): 1143.
2. Abdullah, Badr et al. “Cross-Domain Adaptation of Spoken Language Identification for Related Languages: The Curious Case of Slavic Languages.”
3. Rao, Kanishka et al. “Grapheme-to-phoneme conversion using Long Short-Term Memory recurrent neural networks.”
4. Manifold Learning Scikit-learn Documentation: Link
5. Clustering Scikit-learn Documentation: Link