L-Università ta' Malta

ICS2203

Statistical Natural Language Processing

Speech Phoneme Analysis and Classification

Programmed using Python

By Mark Dingli
20703H

Feature Extraction

The product of the first task can be found in the CSV file *"feature_extraction_data.csv"*, which contains data for different speakers pronouncing three selected words: "hid", "had", and "hood". These words were chosen because they represent distinct vowel phonemes (in ARPABET notation): IH, AE, and UH, respectively. Analyzing different vowel phonemes can help the classifier differentiate between them and improve its overall performance.

The average formant values for each vowel phoneme are as follows:

IH (as in "hid")
F1: 350-500 Hz
F2: 1800-2200 Hz
F3: 2400-2600 Hz

AE (as in "had")
F1: 600-850 Hz
F2: 1000-1200 Hz
F3: 2300-2600 Hz

UH (as in "hood")
F1: 400-500 Hz
F2: 700-900 Hz
F3: 2700-2900 Hz

The five different accents I chose to use are *brm_001, crn_001, ean_001, eyk_001,* and *gla_001*. From each of the five chosen accents, ten speakers were selected, five female and five male, resulting in a total of 150 rows.

The three formants for each speaker were extracted using *"Pratt"*. This allowed me to choose a specific section of the WAV file and find the formants of each vowel. This helps to understand the central tendency of each phoneme's formant distribution and provides a reference for comparison. The CSV file *"feature_extraction_data.csv"* was then used to create the dataset for the second part of the task, which involves building the k-Nearest Neighbors classifier.

Classifier

The "main.py" script implements a k-Nearest Neighbors (kNN) classifier for phoneme recognition using extracted formant features. After importing necessary libraries and reading the data from the CSV file, Principal Component Analysis (PCA) is employed to reduce dimensionality and select the most informative features. The kNN classifier is trained and evaluated using multiple experiments with different k values and training-testing splits to find the best performing k value. During these experiments, the *"random_state"* parameter is set to an integer that determines the seed for the random number generator used by the *train_test_split* function. By setting it to the current iteration index, it ensures that each iteration uses a different random seed, resulting in a different training and test set each time. The classifier's performance is assessed using confusion matrices and F1 scores.

The output displays the results of running the kNN classifier for different k values (3, 5, 7, 9, and 11) across five experiments. For each combination of k value and experiment, a confusion matrix and an F1 score are shown. After completing all experiments for a particular k value, the average F1 score is calculated and displayed. The best k value is determined by comparing the average F1 scores across all experiments, and its corresponding average F1 score is presented as the final output.

Questions and Investigation

1. *How does performance change with different values of K?*

In the k-Nearest Neighbors algorithm implementation, different k values were tested using the *"Chebyshev"* distance metric. For each k value, five experiments were conducted with seed values ranging from 0 to 4 (inclusive) to control the randomization of the training and testing splits. The F1 scores were recorded for each individual experiment, and the average F1 score was calculated for each k value. The test began with a k value of 3 and continued with k values of 5, 7, 9, and 11. The performance of the classifier fluctuated as the k value increased, and the best performing k value was determined to be 3, with the highest average F1 score (0.8705).

K value	*Experiment 1*	*Experiment 2*	*Experiment 3*	*Experiment 4*	*Experiment 5*	*Average F1*
K=3	0.8421	0.8962	0.8725	0.8962	0.8454	**0.8705**
K=5	0.8412	0.8702	0.8762	0.8692	0.8723	**0.8658**
K=7	0.8435	0.8729	0.8717	0.8434	0.8438	**0.8551**
K=9	0.8666	0.8531	0.8255	0.8697	0.8166	**0.8463**
K=11	0.8666	0.8250	0.8567	0.8458	0.8166	**0.8421**

Figure 1 – Graph illustrating the k-Nearest Neighbors performance for each k value

2. *What distance metric did you use? Tried any others?*

First, I tested the algorithm using the widely used default Euclidean distance metric. However, to explore the performance of other distance metrics, I tested the algorithm using different distance metrics. From the results obtained, the Chebyshev distance metric outperformed the other metrics, including Euclidean and Manhattan, and provided the most accurate results. Therefore, the Chebyshev metric with a k-value of 3 was found to be the best option for my algorithm. Resulting in an F1-score of 0.8705.

Metric	*k=3 Average*	*k=5 Average*	*k=7 Average*	*k=9 Average*	*k=11 Average*	*Best*
Euclidean	0.8547	0.8502	0.8652	0.8450	0.8376	**k=7**
Manhattan	0.8488	0.8556	0.8651	0.8391	0.8525	**k=7**
Chebyshev	0.8705	0.8658	0.8551	0.8463	0.8421	**k=3**

Figure 2 - Graph showing a comparison of average F1 scores for k values 3, 5, 7, 9, and 11 across Euclidean, Manhattan, and Chebyshev distance metrics, identifying the best k value for each metric

3. *How does performance change when classification is done one data for a single gender alone, or when data from both genders are put together?*

I modified the code so that I could separately evaluate the performance of the phoneme classification model on data from the male and female datasets, as well as on combined data from both genders. It is interesting to note that the best performing k value for the male dataset was different from that of the female dataset and the combined dataset. This shows that the optimal k value for phoneme classification depends on the specific characteristics of the data being used.

Overall, it seems that combining data from both genders results in a slightly lower performance compared to using data from a single gender alone. However, the differences in performance are relatively small.

Gender	*k=3 Average*	*k=5 Average*	*k=7 Average*	*k=9 Average*	*k=11 Average*	*Best*
Female	0.875	0.864	0.874	0.854	0.855	**K=3**
Male	0.834	0.834	0.866	0.877	0.855	**K=9**
Both	0.870	0.866	0.855	0.846	0.842	**K=3**

Figure 3 – Graph showing a comparison of average F1 scores for k values 3, 5, 7, 9, and 11 across Female, Male, and Both gender , identifying the best k value for each gender

4. *What are the vowel-based phonemes that produce the most confusion (base this off your confusion matrix)*

The analysis of the confusion matrix reveals that the phoneme "AE" from the word "had" had the highest frequency of false positives and false negatives, indicating the most confusion. On the other hand, the phoneme "UH" from the word "hood" had the lowest frequency of false positives and false negatives, indicating the least confusion. These findings suggest that certain phonemes may be more difficult to distinguish than others, and that this may be influenced by differences in pronunciation across accents or dialects that were used for the feature extraction.

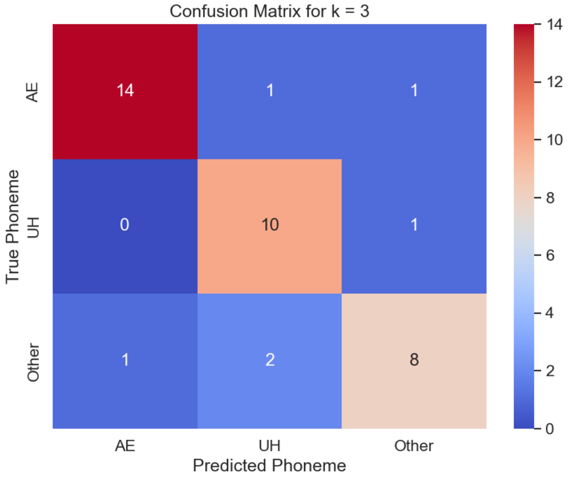

Figure 4 – Heatmap of the confusion matrix for vowel-based phoneme classification (k=3)

This heatmap was created using a Python script and the appropriate visualization libraries, representing the confusion matrix for the vowel-based phoneme classification (k=3).The rows correspond to the true phoneme classes, while the columns represent the predicted classes. The first row shows the results for the 'AE' phoneme (from the word "had", 14 instances were correctly classified, while 1 was misclassified as 'UH' and another as 'Other'. The second row displays the 'UH' phoneme (from the word "hood"), out of 11 instances, 10 were accurately predicted, and 1 was misclassified as 'Other'. Lastly, the third row represents the 'Other' phoneme, 8 instances were classified correctly, whereas 1 was misclassified as 'AE' and 2 as 'UH'. The color intensity in each cell indicates the frequency of each classification outcome, highlighting the most confused phonemes.