Ameen Qureshi

Khalil Iskarous

LING 487

May 10th, 2024

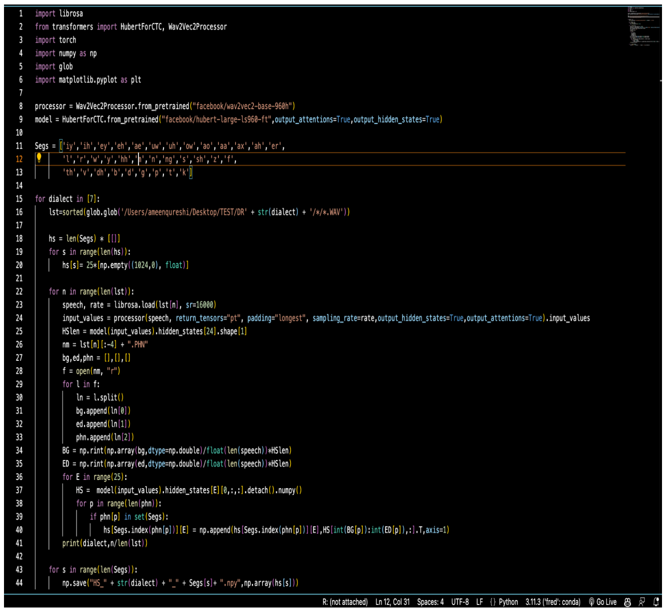
Sonarity in the Timit Dataset

Sonority refers to the relative loudness or prominence of a sound within a language. Through spending time looking at amplitude and the effect it has on attention, I shifted focus to looking at how sonority affects the hidden states of a verb.
To start things off, let's look into the background of Hubert. Hubert, developed by Facebook AI, is the most important LLM when it comes to speech processing. Due to processing sound, we run into issues such as the fact that there are multiple sound units in each input utterance, there is no lexicon of input sound units during the pre-training phase, and sound units have variable lengths with no explicit segmentation. To deal with these three problems, Hidden-Unit BERT was created for speech representation learning which utilizes an offline clustering step to provide aligned target labels to enable the user to get a prediction loss similar to BERT. Since it is built on the BERT architecture it can achieve high accuracy and is also highly effective for tasks like speech recognition, speaker recognition, and classification.

In this assignment, I will use the following phonemic segments: /k/, /t/, /b/, /g/, /d/ and /aa/, /ah/, /ow/, /ih/, /ey/ using HuBERT. These segments represent both low and high sonority stops and vowels and I will be looking at how the representations differ when compared to vowels and also in different dialects. This approach is important as by examining the magnitude of Cohen's d for different phonetic categories and dialects, we can identify which features or aspects of the hidden representations contribute most to the observed differences. This information can guide further analysis and research. Through the utilization of HuBERT, my aim is to realize this objective by capturing phonetic distinctions and its capacity to generalize across diverse linguistic contexts.

Setup Process

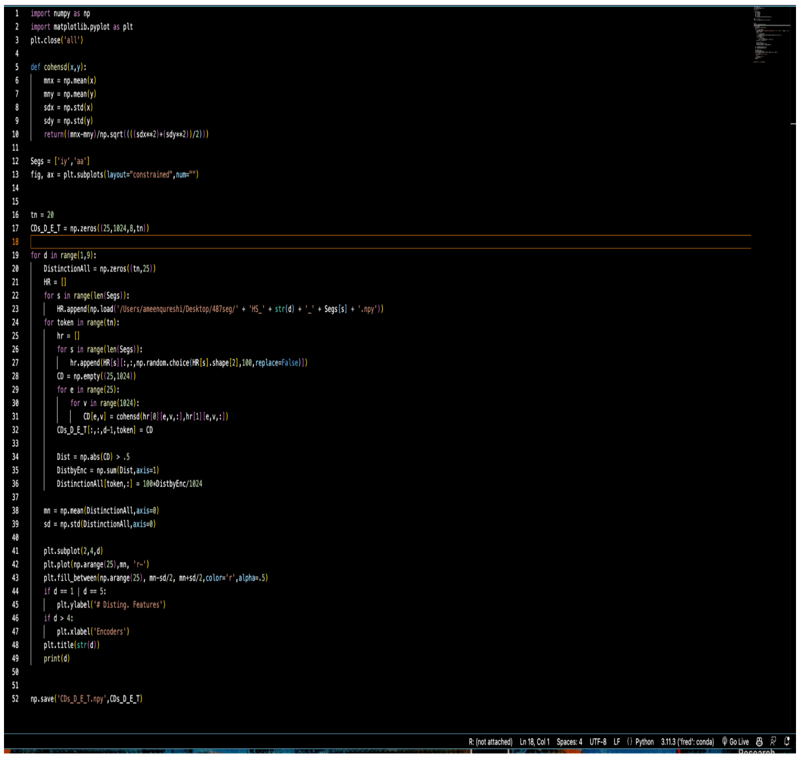
To set myself up for this paper, I started by downloading the TIMIT dataset, and trying to run the HUBERTGetData.py. I tried running it on Colab but during the runtime of specific dialects, the internet went out so I switched to VScode. I was fortunate enough to have enough space as I already had a version of the TIMIT Dataset from prior work and was able to use that. I spent a while running the code and after trying each dialect I realized that there was an issue when saving the hidden states to .py files. The code also took ages to run so I shifted trying it based on Dialects. I thought this would have less issues but I still ran into those issues and was annoyed. Though this was very frustrating to deal with, I decided to use the Segment data within the FinalProject folder. I still am unsure on why this error was present as when I had my peers double check my work, they could not find any issues with it and may have to deal with python/python library being out of date. There may also be other issues regarding my



```
import librosa
from transformers import HubertForCTC, Wav2Vec2Processor
import torch
import numpy as np
import glob
import matplotlib.pyplot as plt

processor = Wav2Vec2Processor.from_pretrained("facebook/wav2vec2-base-960h")
model = HubertForCTC.from_pretrained("facebook/hubert-large-ls960-ft",output_attentions=True,output_hidden_states=True)

Segs = ['iy','ih','ey','eh','ae','uw','uh','ow','ao','aa','ax','ah','er',
        'l','r','w','y','hh','m','n','ng','s','sh','z','f',
        'th','v','dh','b','d','g','p','t','k']

for dialect in [7]:
    lst=sorted(glob.glob('/Users/ameenqureshi/Desktop/TEST/DR' + str(dialect) + '/*/*.WAV'))

    hs = len(Segs) * [[]]
    for s in range(len(hs)):
        hs[s]= 25*[np.empty((1024,0), float)]

    for n in range(len(lst)):
        speech, rate = librosa.load(lst[n], sr=16000)
        input_values = processor(speech, return_tensors="pt", padding="longest", sampling_rate=rate,output_hidden_states=True,output_attentions=True).input_values
        HSlen = model(input_values).hidden_states[24].shape[1]
        nm = lst[n][:-4] + ".PHN"
        bg,ed,phn = [],[],[]
        f = open(nm, "r")
        for l in f:
            ln = l.split()
            bg.append(ln[0])
            ed.append(ln[1])
            phn.append(ln[2])
        BG = np.rint(np.array(bg,dtype=np.double)/float(len(speech))*HSlen)
        ED = np.rint(np.array(ed,dtype=np.double)/float(len(speech))*HSlen)
        for E in range(25):
            HS =  model(input_values).hidden_states[E][0,:,:].detach().numpy()
            for p in range(len(phn)):
                if phn[p] in set(Segs):
                    hs[Segs.index(phn[p])][E] = np.append(hs[Segs.index(phn[p])][E],HS[int(BG[p]):int(ED[p]),:].T,axis=1)
        print(dialect,n/len(lst))

    for s in range(len(Segs)):
        np.save("HS_" + str(dialect) + "_" + Segs[s]+ ".npy",np.array(hs[s]))
```

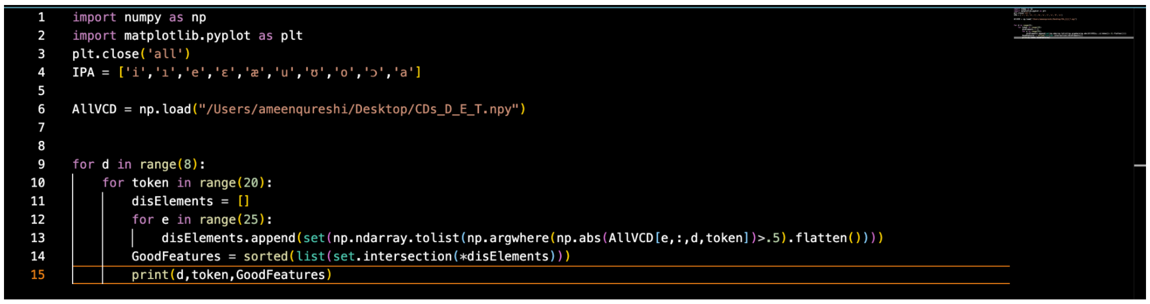
```
R: (not attached)   Ln 12, Col 31   Spaces: 4   UTF-8   LF   {} Python   3.11.3 ('fred': conda)   Go Live
```

laptop and its installation of modules but it was very frustrating to deal with at the moment. I also decided to use this dataset and this approach gives me the ability to work with more variables and also work with dialects. I thought it would be more interesting to look at how dialect affects sonority of vowels and how their sonority affects their cohen's d. I then started looking at the data through the get distinctions by encoder file by calculating Cohen's d. In addition to that, it picks two samples and calculates the thresholds, distinction percentage, and standard deviation. This is really important when looking at the patterns between high sonority stops and low sonority and also vowels vs stops. We can use this to be able to come up with how it affects the model and how sonority plays a role. After that, we use this code below to be given feature indices which we can use for further analysis.



```python
import numpy as np
import matplotlib.pyplot as plt
plt.close('all')

def cohensd(x,y):
    mnx = np.mean(x)
    mny = np.mean(y)
    sdx = np.std(x)
    sdy = np.std(y)
    return((mnx-mny)/np.sqrt((((sdx**2)+(sdy**2))/2)))

Segs = ['iy','aa']
fig, ax = plt.subplots(layout="constrained",num="")


tn = 20
CDs_D_E_T = np.zeros((25,1024,8,tn))

for d in range(1,9):
    DistinctionAll = np.zeros((tn,25))
    HR = []
    for s in range(len(Segs)):
        HR.append(np.load('/Users/ameenqureshi/Desktop/487seg/' + 'HS_' + str(d) + '_' + Segs[s] + '.npy'))
    for token in range(tn):
        hr = []
        for s in range(len(Segs)):
            hr.append(HR[s][:,:,np.random.choice(HR[s].shape[2],100,replace=False)])
        CD = np.empty((25,1024))
        for e in range(25):
            for v in range(1024):
                CD[e,v] = cohensd(hr[0][e,v,:],hr[1][e,v,:])
        CDs_D_E_T[:,:,d-1,token] = CD

        Dist = np.abs(CD) > .5
        DistbyEnc = np.sum(Dist,axis=1)
        DistinctionAll[token,:] = 100*DistbyEnc/1024

    mn = np.mean(DistinctionAll,axis=0)
    sd = np.std(DistinctionAll,axis=0)

    plt.subplot(2,4,d)
    plt.plot(np.arange(25),mn, 'r-')
    plt.fill_between(np.arange(25), mn-sd/2, mn+sd/2,color='r',alpha=.5)
    if d == 1 | d == 5:
        plt.ylabel('# Disting. Features')
    if d > 4:
        plt.xlabel('Encoders')
    plt.title(str(d))
    print(d)


np.save('CDs_D_E_T.npy',CDs_D_E_T)
```

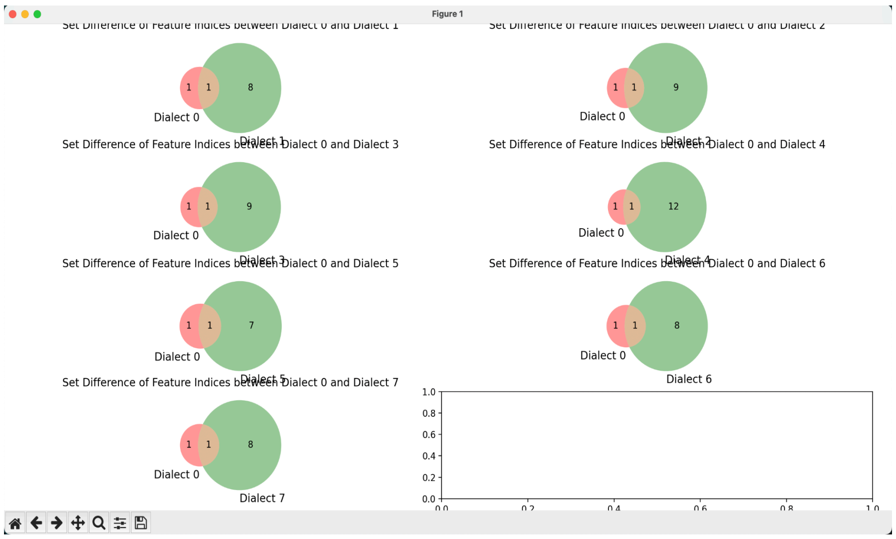
R: (not attached) Ln 18, Col 1 Spaces: 4 UTF-8 LF {} Python 3.11.3 ('fred': conda) Go Live

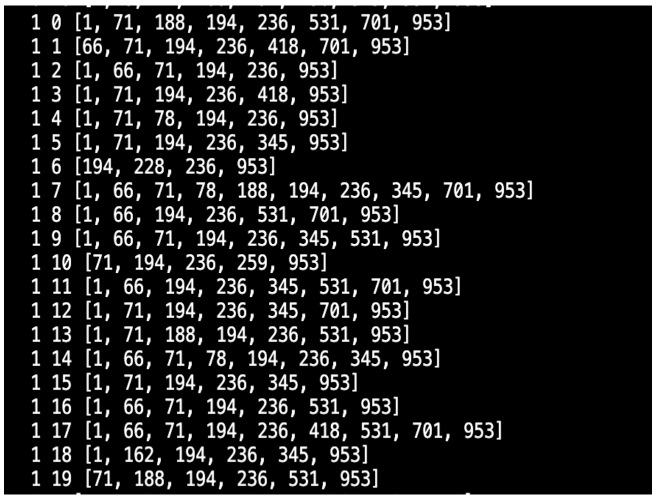


```python
import numpy as np
import matplotlib.pyplot as plt
plt.close('all')
IPA = ['i','ɪ','e','ɛ','æ','u','ʊ','o','ɔ','a']

AllVCD = np.load("/Users/ameenqureshi/Desktop/CDs_D_E_T.npy")


for d in range(8):
    for token in range(20):
        disElements = []
        for e in range(25):
            disElements.append(set(np.ndarray.tolist(np.argwhere(np.abs(AllVCD[e,:,d,token])>.5).flatten())))
        GoodFeatures = sorted(list(set.intersection(*disElements)))
        print(d,token,GoodFeatures)
```

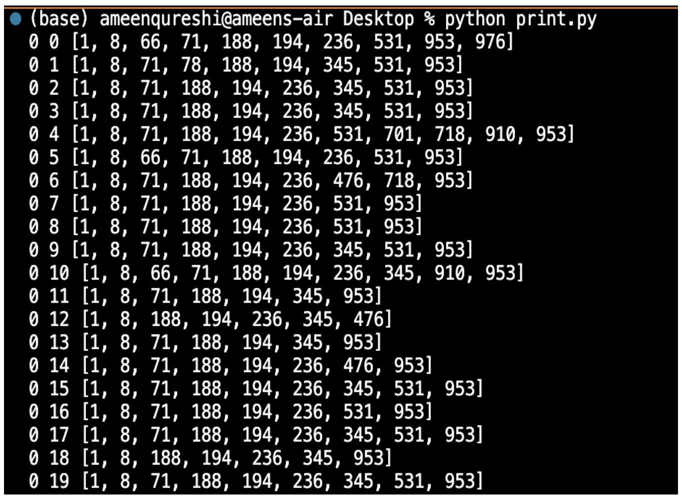
Overlap Between Dialects

To start things off, I wanted to start by looking at the high sonority vowels k and comparing that to the vowel /ih/. /K/ is a high sonority stop, as while it is louder than other stops it still has a complete closure and interruption of airflow. This is in comparison to /ih/ and through being a vowel, it



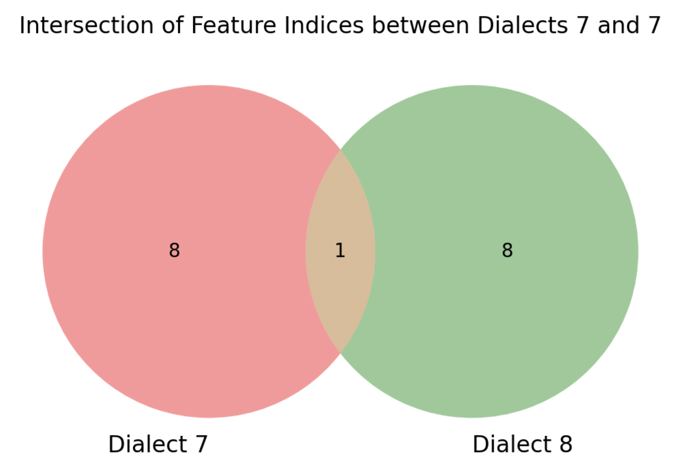


```
 1 0 [1, 71, 188, 194, 236, 531, 701, 953]
 1 1 [66, 71, 194, 236, 418, 701, 953]
 1 2 [1, 66, 71, 194, 236, 953]
 1 3 [1, 71, 194, 236, 418, 953]
 1 4 [1, 71, 78, 194, 236, 953]
 1 5 [1, 71, 194, 236, 345, 953]
 1 6 [194, 228, 236, 953]
 1 7 [1, 66, 71, 78, 188, 194, 236, 345, 701, 953]
 1 8 [1, 66, 194, 236, 531, 701, 953]
 1 9 [1, 66, 71, 194, 236, 345, 531, 953]
 1 10 [71, 194, 236, 259, 953]
 1 11 [1, 66, 194, 236, 345, 531, 701, 953]
 1 12 [1, 71, 194, 236, 345, 701, 953]
 1 13 [1, 71, 188, 194, 236, 531, 953]
 1 14 [1, 66, 71, 78, 194, 236, 345, 953]
 1 15 [1, 71, 194, 236, 345, 953]
 1 16 [1, 66, 71, 194, 236, 531, 953]
 1 17 [1, 66, 71, 194, 236, 418, 531, 701, 953]
 1 18 [1, 162, 194, 236, 345, 953]
 1 19 [71, 188, 194, 236, 531, 953]
```

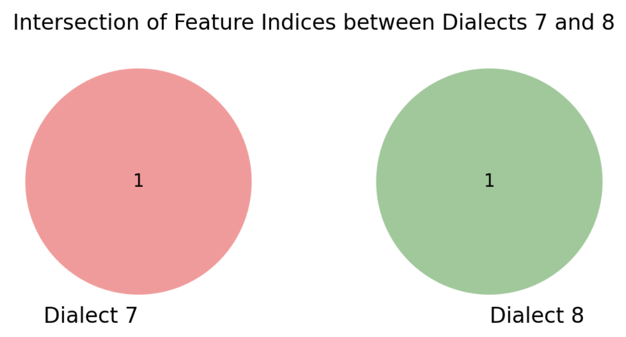


```
(base) ameenqureshi@ameens-air Desktop % python print.py
 0 0 [1, 8, 66, 71, 188, 194, 236, 531, 953, 976]
 0 1 [1, 8, 71, 78, 188, 194, 345, 531, 953]
 0 2 [1, 8, 71, 188, 194, 236, 345, 531, 953]
 0 3 [1, 8, 71, 188, 194, 236, 345, 531, 953]
 0 4 [1, 8, 71, 188, 194, 236, 531, 701, 718, 910, 953]
 0 5 [1, 8, 66, 71, 188, 194, 236, 531, 953]
 0 6 [1, 8, 71, 188, 194, 236, 476, 718, 953]
 0 7 [1, 8, 71, 188, 194, 236, 531, 953]
 0 8 [1, 8, 71, 188, 194, 236, 531, 953]
 0 9 [1, 8, 71, 188, 194, 236, 345, 531, 953]
 0 10 [1, 8, 66, 71, 188, 194, 236, 345, 910, 953]
 0 11 [1, 8, 71, 188, 194, 345, 953]
 0 12 [1, 8, 188, 194, 236, 345, 476]
 0 13 [1, 8, 71, 188, 194, 345, 953]
 0 14 [1, 8, 71, 188, 194, 236, 476, 953]
 0 15 [1, 8, 71, 188, 194, 236, 345, 531, 953]
 0 16 [1, 8, 71, 188, 194, 236, 531, 953]
 0 17 [1, 8, 71, 188, 194, 236, 345, 531, 953]
 0 18 [1, 8, 188, 194, 236, 345, 953]
 0 19 [1, 8, 71, 188, 194, 236, 345, 531, 953]
```

has high sonority due to their open vocal tract configuration and continuous airflow. When looking at this alongside the features indices of dialect 1 and 0. We are able to notice stark differences between them. The overlap suggests that there is a minimal set of features common to both dialects for distinguishing phonemes. This could imply that there may be a core set of properties that are crucial to comprehension across dialects and that leads to some overlap. It also suggests that there are features that are particularly significant in one dialect but not in the other. This can be through how phonetic characteristics/phoneme variations might be emphasized in some dialectics to the point where there are few similar features. Another idea for the differentiation in phonemes between these dialects could rely largely on different phonemes for understanding.



These results are still somewhat supported when we look at the same vowels when approaching different dialects. While there is a small amount of overlap between the dialects it’s hard to tell if sonority is at all related to their overlap. We can look more closely at this by changing up which phonemes we check out. By only changing the /ih/ to /k/ we are able to see that now there is



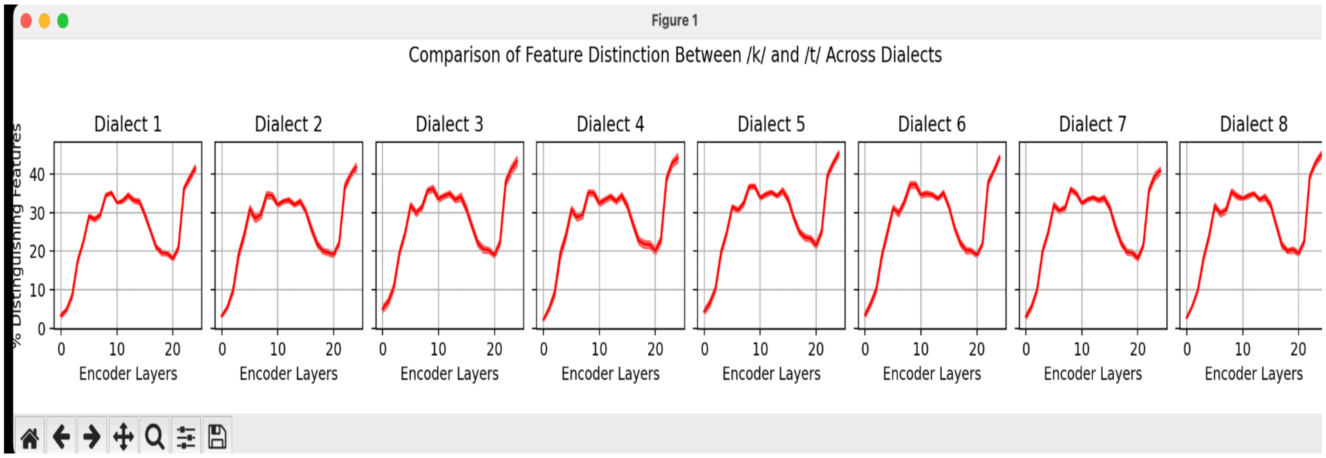
now no overlap between the dialects. This minimal overlap suggests that the features used to distinguish /k/ and /ih/ are quite different, underscoring the sonority contrast between a stop and a vowel.

Switching from /ih/ to /t/ involves comparing two phonemes (/k/ and /t/) that are much closer on the sonority scale—both are voiceless stops. The new Venn diagrams exhibit almost no overlap between the distinguishing features of /k/ and /t/. This near-total lack of shared features highlights how subtle the acoustic and articulatory differences are between these two stops, compared to the differences between a stop and a vowel. The significant reduction in overlapping feature indices when analyzing two phonemes close on the sonority scale (/k/ and /t/) versus a stop and a vowel (/k/ and /ih/) suggests that models and feature extraction techniques are not sensitive enough to pick up on the slight differences between understanding how closely related phonemes differ in subtle acoustic or articulatory ways can be crucial for improving accuracy and naturalness.

The shift illustrates the importance of sonority in phonetic feature analysis. The fact that there is no overlap between two similar phonemes but there is overlap between different phonemes suggests that it does pick up sonority as a factor in feature indices. Though it cannot pick out distinct sets of features important for distinguishing between phonemes of similar sonority (like /k/ and /t/) as opposed to those of differing sonority (/k/ and /ih/). This issue emphasizes the need for new modeling approaches that can capture these subtle differences effectively, particularly in diverse linguistic environments.

Feature Distinctions over Dialects and Encoders

To Look at the comparison between /g/ (a voiced velar stop) and /aa/ (an open back unrounded vowel) with Hubert, I used a line plot to visualize the similarities and differences. Each line plot shows how the distinction between /g/ and /aa/ evolves across the successive encoder layers of the Hubert model. The x-axis represents the encoder layers, and the y-axis represents the percentage of distinguishing features. (The title is supposed to be /g/ and /aa/ not /k/ and /t/)



In some dialects, I can see sharp peaks and troughs, indicating that certain layers are particularly good or particularly poor at distinguishing between /g/ and /aa/. This could be related to how different acoustic and articulatory features of these phonemes are captured or emphasized in various layers of the model. In the model we see a Layer-by-Layer Variability: The plots exhibit variability in the amount of distinguishing features across the encoder layers.

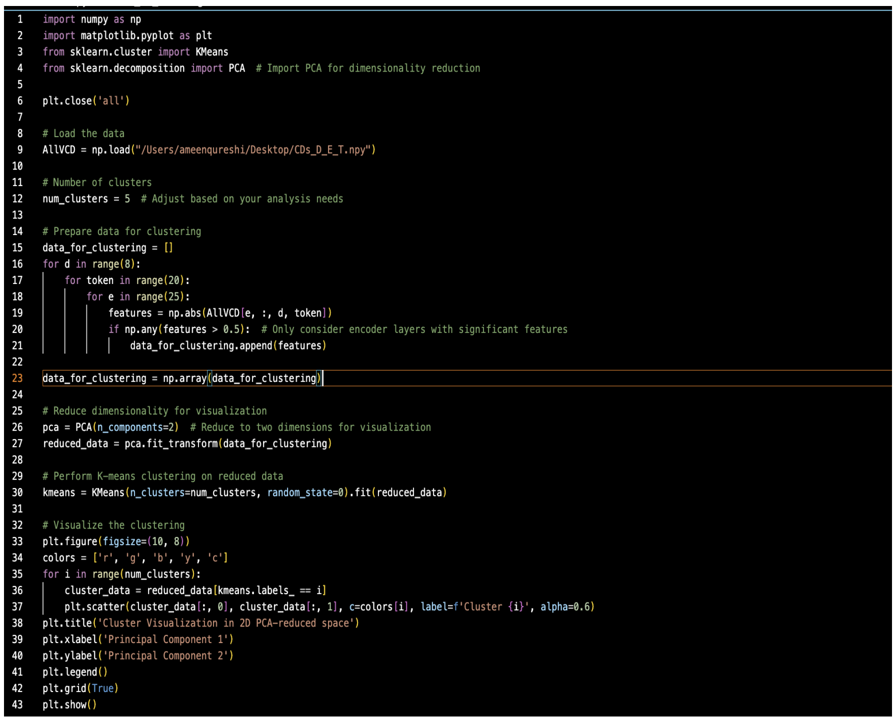
Though at the same time, we see that each plot has a similar pattern with

- Initial Increase: Most dialects show an increase in distinguishing features within the initial layers
- Mid-Layer Peaks and Troughs: Around the middle layers, there are notable peaks and troughs that only slightly differ from each other
- Towards the higher layers, several plots show a resurgence in distinguishing features

These patterns are due to the slight variation in dialect as seen before with how dialect 1 and 0 had different amounts of overlap than 7 and 8. This variability may be due to just that factor. Though it is still possible that due to complexity of the model, the differences may be due to sonority as the vowels are vastly different. Though this visualization may not tell us too much beyond the general differences that I found in the first model. The visualization serves as a valuable tool for analyzing how advanced Hubert manages to capture and differentiate between complex phonetic categories across various dialects. By studying these patterns, we can further refine models to be more sensitive and accurate in real-world applications.

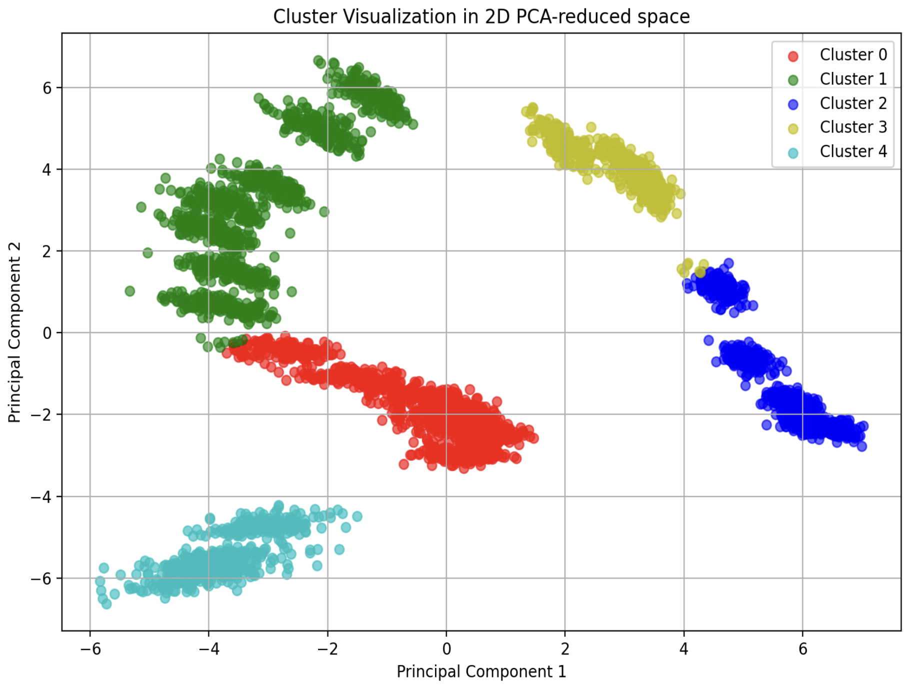
Cluster

In my last approach to looking at the sonority of stops, I decided to approach all 5 consonants with a clustering algorithm as I wanted to see how it would group the 5 different consonants. K-means is a clustering algorithm that partitions the dataset into K distinct, non-overlapping clusters. It does this by minimizing the variance within each cluster, effectively grouping data points that are similar to each other. I also chose to do



```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA  # Import PCA for dimensionality reduction

plt.close('all')

# Load the data
AllVCD = np.load("/Users/ameenqureshi/Desktop/CDs_D_E_T.npy")

# Number of clusters
num_clusters = 5  # Adjust based on your analysis needs

# Prepare data for clustering
data_for_clustering = []
for d in range(8):
    for token in range(20):
        for e in range(25):
            features = np.abs(AllVCD[e, :, d, token])
            if np.any(features > 0.5):  # Only consider encoder layers with significant features
                data_for_clustering.append(features)

data_for_clustering = np.array(data_for_clustering)

# Reduce dimensionality for visualization
pca = PCA(n_components=2)  # Reduce to two dimensions for visualization
reduced_data = pca.fit_transform(data_for_clustering)

# Perform K-means clustering on reduced data
kmeans = KMeans(n_clusters=num_clusters, random_state=0).fit(reduced_data)

# Visualize the clustering
plt.figure(figsize=(10, 8))
colors = ['r', 'g', 'b', 'y', 'c']
for i in range(num_clusters):
    cluster_data = reduced_data[kmeans.labels_ == i]
    plt.scatter(cluster_data[:, 0], cluster_data[:, 1], c=colors[i], label=f'Cluster {i}', alpha=0.6)
plt.title('Cluster Visualization in 2D PCA-reduced space')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.grid(True)
plt.show()
```

this because I am comfortable using clustering through the data science classes I have taken. When looking at the data we see how there are 3 clusters that are focused around the -6-0 and then we see two clusters nearby to each other from 2-6. This is important to look at as it depicts how a program would approach grouping the audio based on their features. One possible



explanation I can think of is that it groups the loudest from the negative and the positive ones are the quietest. If this is true, it would make sense that the blue is most likely /b/ as it is a bilabial stop which differentiates itself from /d/ and /g/. It would make sense that these are grouped semi closely as they fall into the category of voiced stops which are typically louder than their voiceless counterparts. This would also make sense as /k/ and /t/ are voiceless and they tend to be quieter. They are quieter, as they do not need the larynx to vibrate as is the case for voiced stops. One flaw in this idea is that the Y-axis could refer to the pitch and that would throw my predictions entirely off.

Overall through this project, I have a strong idea that sonority makes a difference when a model is creating a hidden representation. This is through the evidence that I presented with overlap between dialects, comparison of feature distinction over the encoders, and through the k-means clustering of the five stops. These all showed slight differences between comparisons of different phonemes suggesting that there is too much evidence to just be a coincidence. With that being said, I still think that while HuBERT represents the pinnacle of speech processing technology. There is still the challenge of fully understanding and quantifying the influence of sonority; its subtle nature makes it an elusive target for computational models. Further research should focus on refining HuBERT's learning algorithm to better process and create hidden representations that are more accurate. These representations should be able to clearly pick apart phonemes easily. Moreover, improving interpretability within these models will enhance our understanding of how sonority is represented and processed currently.

This project, though I did not get an opportunity to work with attention, this project helped me further understand Hidden Representations and helped me to also abridge some of the tools I had to linguistics. I am grateful to have had this opportunity and look forward to helping you with your work over the summer.