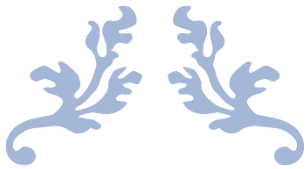


SONG GENRE ANALYSIS

KDAG Task 2

An Introduction to unsupervised learning, clustering and NLP

MUSAIB BIN BASHIR

Vectorization

The dataset has 3 keywords for each song, keyword 1 is the instrument, keyword 2 is mood and keyword 3 is more like vibe of song. Since these 3 are independent variables for each song, so we make a separate vocab for each keyword and use that to implement vectorization. I chose Bag Of Words (BOW) over Term Frequency Inverse Document Frequency(TF-IDF) as our emphasis is on presence, not rarity and TF-IDF downweighs common terms, which isn't ideal in our case where all keywords

```
Vocab keyword 1:
{'violin', 'piano', 'guitar', 'synth',
'banjo', 'brass'}
Size of vocab: 6

Vocab keyword 2:
{'mellow', 'nostalgic', 'upbeat',
'happy', 'energetic', 'sad', 'calm',
'angry', 'emotional'}
Size of vocab: 9

Vocab keyword 3:
{'fast', 'melodic', 'heavy', 'acoustic',
'slow', 'distorted', 'rhythmic',
'twangy', 'upbeat', 'danceable'}
Size of vocab: 10
```

are equally relevant. And also, our document isn't that complex, so BoW is a simpler and better option in this case, our dataset has fixed vocabularies for each keyword type, so rarity is not a critical factor.

Implementing BoW gives us 3 matrices, of sizes (147,6), (147,9), (147,10) respectively.

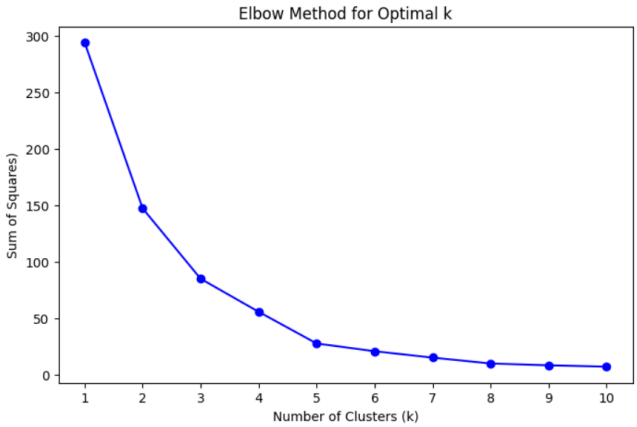
```
[[0 0 1 0 0 0]
 [0 0 0 0 0 1]
 [0 0 0 0 1 0]
 [0 0 0 1 0 0]
 [0 0 0 1 0 0]]
First 5 entries in the BoW
matrix of keyword 1
```

Dimensionality reduction

Using Principal Component Analysis, we reduce the 3 matrices from their initial dimensions to a final dimension of (147,2).

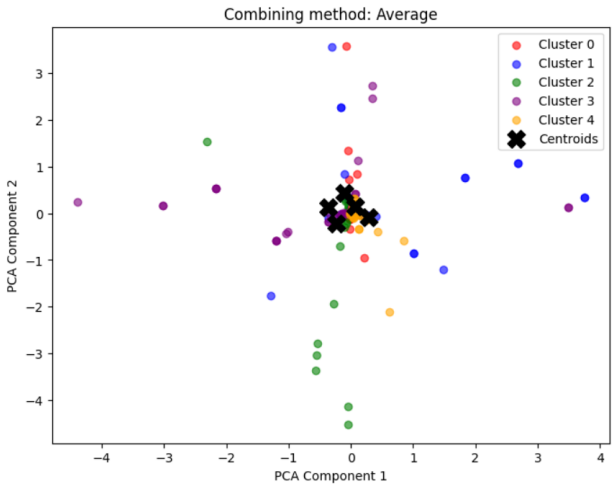
Combining Embeddings & Clustering

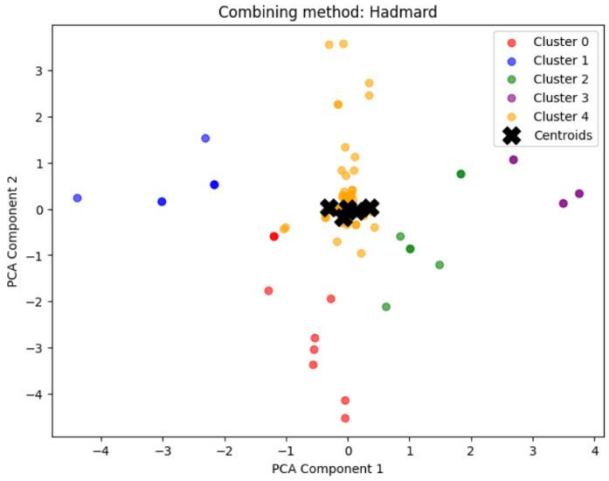
To combine the 3 matrices of keywords we have multiple options like, simple average, Hadmard product, weighted average, cosine similarity.
Weighted average is ignored, as a good estimate of the weights can be done by a simple neural network, but due to library restriction using that would make this unnecessarily complex. So we try rest of the methods and generate the final embedding.
Then we use k-means clustering algorithm to make k clusters.

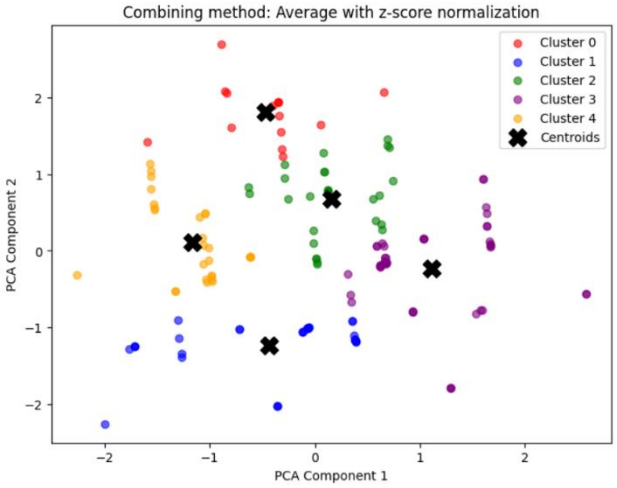


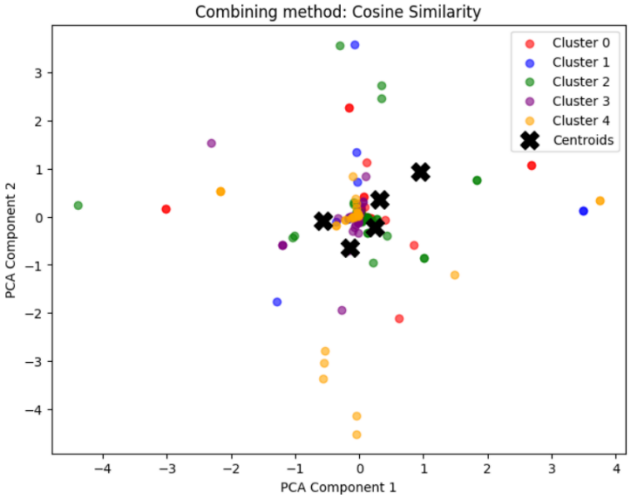
Using the elbow method we see the optimal value for k is 3, but since we have 5 genres in total, we use k=5, to generate 5 clusters, each one of which would be later assigned to a unique genre.

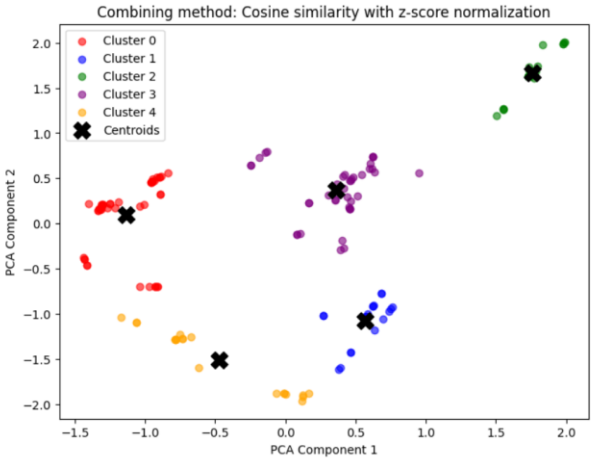
For each of the combining method we run k-means algorithm to evaluate which one gives the best clusters. Here, by best we mean well separated clusters where each cluster has atleast few points.





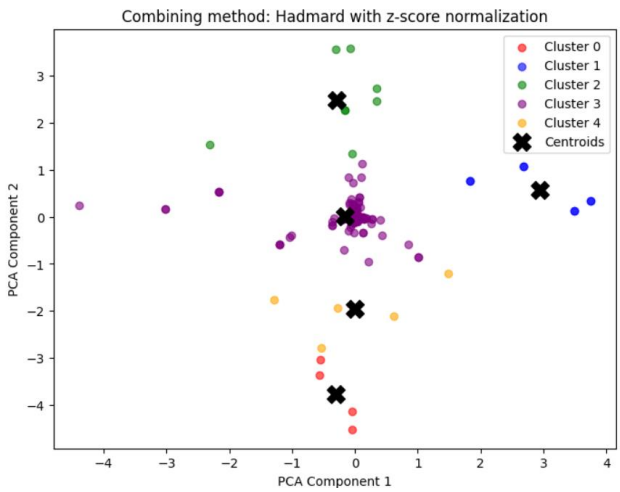






From these scatter plots we realise the final embeddings are too close to each other, so to make the data work better with clustering we use z-score normalization.

z-score normalization has improved distinctive separation of clusters, and we can see cosine similarity looks the best among the 3 methods. So we proceed ahead with this method.



Analysis

First we build the ground truth matrix (or confusion matrix) for the clusters obtained from the previous methods.

	classical	country	hip-hop	pop	rock
0	8.695652	26.086957	10.869565	21.739130	32.608696
1	14.285714	19.047619	23.809524	19.047619	23.809524
2	35.000000	20.000000	15.000000	25.000000	5.000000
3	20.000000	10.000000	35.000000	25.000000	10.000000
4	25.000000	25.000000	15.000000	10.000000	25.000000

And we also compute the $\mathrm{Silhouette\ Score}$ which comes out to be 0.6250, showing clusters are fairly separated.

Now we move on to label the clusters based on the frequency matrix

	classical	country	hip-hop	pop	rock
0	4	12	5	10	15
1	3	4	5	4	5
2	7	4	3	5	1
3	8	4	14	10	4
4	5	5	3	2	5

From this, if we take the max count genre as the label the Cluster to Genre Mapping we obtain is:

{0: 'rock', 1: 'hip-hop', 2: 'classical', 3: 'hip-hop', 4: 'classical'}

Hip-hop & classical appear twice whileas country & pop isn't mapped to any cluster.
So we do a tradeoff here, we assign cluster cluster 1 to pop and cluster 4 to country. As cluster 1 has nearly similar number of hip-hop and pop songs. Same logic for cluster 4. It should be noted that although this step would decrease accuracy, we are doing so that our predictions include all 5 genres.

New Cluster to Genre Mapping:

{0: 'rock', 1: 'pop', 2: classical, 3: 'hip-hop', 4: 'country' }

Prediction

Using the centroids we obtained from k-means clustering, we can take keywords of new songs, process them using the keywords vocab we have already built, and apply PCA, cosine similarity and then z-score normalization. Then using this new embedding we measure the algebraic distance of each point from each centroid and assign it to the cluster whose centroid is closest to it. We use the cluster to genre mapping to print the finally predicted genre.

Keywords: ['piano', 'calm', 'slow']
Genre: country

Keywords: ['guitar', 'emotional', 'distorted']
Genre: hip-hop

Keywords: ['synth', 'mellow', 'distorted']
Genre: hip-hop

Bonus

Accuracy Measurement:
We input the z-score normalized embeddings and using distance from the centroids of already trained k-means model, we get a label for each of the 147 inputs. Then we use the Cluster to Genre Mapping and compare that with the truth label.
Here are the results obtained:

Accuracy using initial genre mapping: 31.2925%
Accuracy using new genre mapping: 30.6122%

Footnote

Elbow method calculation use scikit library that's why it isn't in the .ipynb file.