REPORT: HIGH-SPEED IDENTIFICATION

By Richa Yadav

OBJECTIVE

The objective of this project is to research and identify algorithms for high-speed identification of a particular voiceprint from a pool of hundreds of voiceprints.

EXISTING APPROACH

The existing approach compares a given voiceprint with all the samples in the database. If the number of voiceprints in the pool grows exponentially, the algorithm will take a lot of time to compare against every single sample.

PROPOSED APPROACH

The proposed approach focuses on compiling a set of clusters of voiceprints (a sub-sample of the entire set of voiceprints). These clusters can be thought of as a reduction in the total number of comparisons that the algorithm needs to make in order to identify the user.

DATA

After browsing across multiple free-speech audio datasets, I decided to use the LibriSpeech dataset (train-clean with 360 hours of speech). This dataset has audio samples of 921 distinct people reading a book in English. The data is available in FLAC files for each user in a separate folder. The following processing steps are used:

- Since the individual FLAC files aren't long enough as desired, a set of 10 initial audio samples for a user are combined using the Sox library to get enough audio to get a richer voiceprint representation. This is then used as training data.
- The last 3 files of a user are combined into one single FLAC file and then used as the testing set.
- The above selection of FLAC files is done to ensure that the FLAC files in the training and testing sets don't overlap and hence avoid data leakage.

After the FLAC files are combined for training and testing sets, the Resemblyzer library is used to convert the FLAC files into a voiceprint representation of a 256-bit vector. To get a voiceprint representation do:
Step 1: Convert the FLAC file for each user into WAV data (array representation).
Step 2: Use the resemblyzer encoder to convert the WAV files into the 256-bit voiceprint representation.

```
for path in wav_fpaths:
    wav_temp = preprocess_wav(path)
    encoded_val = encoder.embed_utterance(wav_temp)
    df = pd.DataFrame(encoded_val,columns=["emded_val"])
```

The training and testing sets are built using the above process for each individual user file. PFB a snapshot of the example of how the datasets look after the above data transformation.

file	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
train-clean-360\audio_samples\data\tra	0.01629	0	0	0	0	0.029195	0.008131	0.011694	0.110153	0.003031	0.143263	0.040228	0.000164	0.007499	0.000414
train-clean-360\audio_samples\data\tra	0.12024	0	0.049727	0	0.00042	0.029869	0	0.003627	0.129419	0.041159	0.037883	0.089378	0.011517	0.099885	0.011687
train-clean-360\audio_samples\data\tra	0.008594	0.024654	0.00024	0	4.38E-05	0.161903	0	0.01183	0.073376	0.084746	0.125548	0.127763	0.023949	0.036731	0.010723
train-clean-360\audio_samples\data\tra	0.143008	0.020304	0.033165	0	0	0.043921	0.009493	0.051434	0.002535	0.028895	0.109424	0.142354	0.030131	0	0.017354
train-clean-360\audio_samples\data\tra	0.001208	0.000666	0.057746	0	0.035551	0.004786	0.001054	0.000195	0.015824	0.009679	0.068906	0.015587	0.026834	0	0.011361
train-clean-360\audio_samples\data\tra	0.007487	0.000332	0.000362	0	0.003164	0.006451	0.030333	0.014118	0.126251	0.008096	0.093204	0.031386	0.010645	0.000247	0.020467
train-clean-360\audio_samples\data\tra	0.019888	0.003666	0.224752	0	0.017393	0.000809	0.013916	0.016347	0.004576	0.141982	0.06927	0.123291	0.118619	0	0.005489
train-clean-360\audio_samples\data\tra	0.013179	0.000657	0.142908	0	0.007502	0.007379	0.010368	0.067767	0.038182	0.003854	0.138202	0.039681	0.0269	0.000915	0.002297

MODELING

The input data is of dimension 921 rows x 256 columns for the voiceprint representation and the respective file name. Since this is a clustering problem, I set the baseline reference model as the KMeans algorithm. I also tried using KDTree and Spectral Clustering + SVM Classifier to find clusters.

Modelling process:

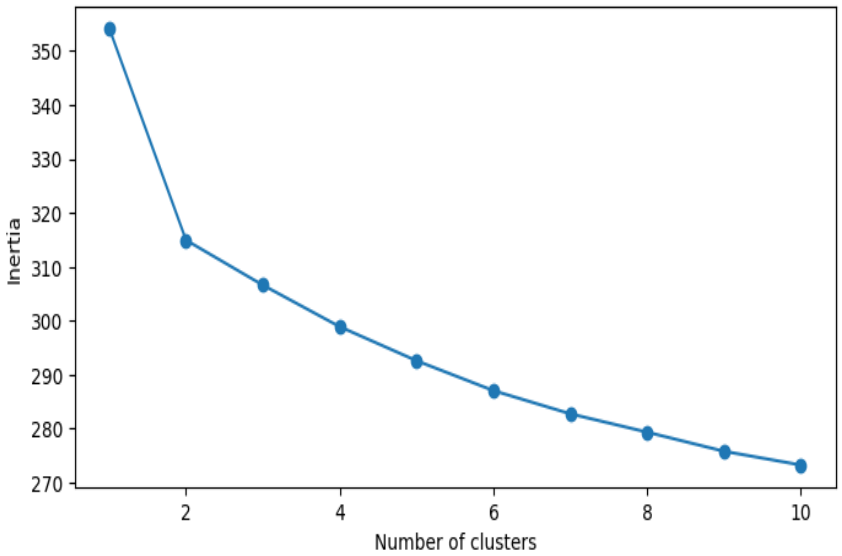
- Each training and testing sample has the "User Name" assigned to their voice print data.
- Use the training set to assign cluster labels to all 921 users.
- Use the fitted model on test data to predict the cluster that the given test sample might belong to.
- Once a cluster for the testing set is identified find the closest match for the sample using cosine similarity against all samples of the training data but for the predicted cluster only.
- The highest cosine similarity sample of training data and testing data is the Identified User after segmentation.

**This approach is used for the KMeans modeling only and Spectral Clustering + SVM Classifier only.*

Baseline Model

The KMeans algorithm from sklearn library is used to create cluster segmentation.

To determine the appropriate number of clusters, I used the elbow method. As per the elbow graph, we can use 4+ clusters.



KDTree model

The KDTree algorithm is used to construct a tree-based search space that can be trained and then run queries to find the top N closest predictions based on the distance metrics used.

Modelling process:

- Each training and testing sample has the "User Name" assigned to their voice print data.
- Use the training set to construct the tree using the KDTree method from sklearn.neighbors for all 921 users.
- Find the top 5 closest predictions using the query method.
- For this approach, the results will be reported in terms of accuracy for up to 5 identifications. The idea is to consider the next prediction incase the first prediction isn't correct.

Spectral Clustering + SVM Classifier

Many clustering algorithms don't have prediction functions to ensure the prediction of the cluster label for unseen data, this approach accounts for the same.

- Used Spectral Clustering to first cluster the training data and then used the SVM classifier to train on the training data with the "cluster label" as the predicted label.
- Applied the trained classifier on unseen testing data to get results.

Note: I also considered the locality-sensitive hashing (MinHash) technique for this problem but I was not able to implement the technique as its generally used in text similarity problems.

RESULTS

The table below summarizes the testing results with the actual results, i.e., the results of the identification of all 921 users from the test set with the training set. Few key points:

- The KMeans algorithm took 2 minutes 19.6 seconds to test all 921 records.
- The KDTree algorithm for up to 5 predictions took around 1.58 seconds to give the query results output.
- The SVM classifier took 0.015 seconds to provide the cluster labels.

Algorithm	#incorrect identifications	Accuracy on last 3 flac files combined
Baseline(Kmeans with 5 clusters)	201	78.18%
Scipy.KDTree Algorithm/w 24 leaves - considering first output	112	87.84%
Scipy.KDTree Algorithm/w 24 leaves - considering second output	80	91.31%
Scipy.KDTree Algorithm/w 24 leaves - considering third output	72	92.18%
Scipy.KDTree Algorithm/w 24 leaves - considering fourth output	65	92.94%
Scipy.KDTree Algorithm/w 24 leaves - considering fifth output	55	94.03%
Spectral clustering and then classification using SVM	128	86.10%

CONCLUSION

As we can see the accuracy is quite promising on the testing set for KDTree and Spectral Clustering with SVM classifier. As the number of records was not much (~900 voiceprints) the scalability of these models on thousands of records can be put to test in future work.