SLP - Group Nº 5 Second Lab Report

João Amoroso
96875

João Teixeira
96879

***Abstract*—In this paper, we try to solve a Speech Identification task that consists on identifying between 6 languages (*Basque*, *Catalan*, *English*, *Galician*, *Portuguese* and *Spanish*), the language spoken in a audio file. The solution developed has two phases: feature extraction and classification. In feature extraction, we tried different approaches such as *CMVN*, *VAD*, *SDC*, *Double deltas*, etc.... In the classification, we trained a GMM for each language in order to see which is the most likely language that a certain audio corresponds.**

***Index Terms*—VAD, CMVN, Delta, Double-Delta, Speech Classification**

I. Introduction

In this lab assignment we trained a model to identify the language of the audio provided.

Our model follows the pipeline present in Figure 1.

For the feature extraction, we tried combinations of different techniques that will be described later on.

For the classification, we trained a Gaussian Mixture model for each of the 6 languages. Due to time constraints, we didn't dive deeper in this phase. If we had more time, we would like to try other models like SVM or CNN.

Fig. 1. Pipeline of our model

II. Feature Extraction

A. Pre-Emphasis

After looking at some papers, we found out that in speech processing it is usually a good practice to apply a high pass filter to the signal before doing any kind of processing. So, we decided to apply a high pass filter that follows the following formula:

$$\gamma(0) = s(0)$$
$$\gamma(t) = s(t) - s(t-1), \forall x \in 1, ..., N$$

B. Deltas

As learned in the theoretical lectures, deltas are a great way to analyze transitions between features. As speech is a time-variant signal we tried to use them, which gave us a huge improvement in the accuracy.

However, the approach mentioned above only "sees" a frame at a time and doesn't give any information about the causality between frames. The SDC is a known technique that tries to solve this problem.

C. Shifted Delta Cepstrum (SDC)

This strategy, as mentioned above, makes use of deltas to compute features. Essentially, this technique gathers a set of deltas per frame in order to add a causal relation between them, that in the normal way isn't possible.

This approach gave us a significantly increase in the accuracy of the model.

D. Wiener Filter

We decided to test Wiener filter to mitigate possible noise present in the audio files. This filter could be used in both the signal or the MFCC. After some experiments in both of the features, we realised that this feature was only degrading our results.

E. Spectral Centroid

This feature tries to extract the "centre of mass" of each frame present in the audio. It has a *correlation with the brightness of the sound* [1]. having this in mind, we thought that it was a valuable feature to take in account.

F. Spectral Roll-Off

This feature measures the frequency that is below a percentage of the total spectral energy (85% in our case). As this is calculated frame by frame, we thought that this could improve our model since maybe it can give less importance to small noises that might be in each frame.

G. Zero Crossing Rate

As discussed in the previous lab assignment, zero crossing rate is the rate at which a signal transitions from positive to zero to negative or negative to zero to positive. We thought of using it since it gives information about the signal behaviour.

H. Voice Activity Detection (VAD)

Voice Activity Detection is the technique used to detect if there is presence or absence of human speech in a certain signal. As mentioned in the lectures, this feature is essential in speech process, so that the machine learning model only learns about the speech itself without "having to care" about the silence.

We tried different strategies to achieve a good algorithm that can remove unvoiced audio.

1) Root Mean Square (RMS): This approach consists on going through the signal with a frame and compute the *RMS* for each frame. After this mapping, we go through these *RMS* values and compare them with a threshold (that was experimentally calculated) in order to distinguish between voice and unvoiced frames. See Figure 2.

2) Sum Of Squares: This approach consists in going frame by frame through the signal and compute the Sum of Squares of each frame. After these calculations, we removed the frames that have the values lower than the threshold (empirically determined as in the last approach). See Figure 3.

3) Energy Threshold: Similarly to the previous approaches we iterate over the signal frame by frame and compare the first coefficient of the MFCC (since it is highly correlated to the energy of the frame) with some threshold. See Figure 4.

*4) Guassian Mixture Model (*GMM*):* This approach makes use of a GMM with two components to try to discover the voice and unvoiced distribution. The steps are as follows:

- Create a GMM and train it with the first MFCC coefficient.
- After the model is trained, it loops through the MFCC time frames and predicts to which component it belongs.
- Check which component has the lower energy, which is accomplished by choosing the lowest mean
- Remove the frames that belong to the component that corresponds to the unvoiced frames.

We trained the GMM with 2 components and with 100 maximum iterations.

As we can see in the figure, we achieve a nice removal of unvoiced parts. This approach was also the most versatile, since the model adapts to each signal individually. See Figure 5.

*I. Cepstral Mean and Variance Normalization (*CMVN*)*

The CMVN is *a computationally efficient normalization technique for robust speech recognition* [2]. This method is the last step of the feature extraction pipeline.

J. OpenSmile

We also tried to use OpenSmile [3] using *eGeMAPSv02* that uses 88 features. Unexpectedly, we got very poor results (about 20% of accuracy in the *dev* dataset).

III. Classification Model

For the classification model, we trained a GMM for each language that we want to predict. The prediction is obtained by feeding each GMM with the features and see which language is the most likely.

We also wanted to try a SVM but due to the time limit, we didn't had time to explore this model.

IV. X Vectors

For the second task we used the implementation from speech brain that is able to generate a embedding given a signal.

These embeddings are fed to a GMM with 10 components. With this model we obtained 82% accuracy.

We wanted to test more models like CNN, SVM or even a MLP to get better results. Due to time limits this was not accomplished.

V. Results

As mentioned before, we ran our model multiple times with different combinations of techniques, obtaining the following Table I. The parameters used to obtain the values in the table were the ones that came with the jupyter notebook.

Although some of the models did not converged, we chose not to increase the maximum iterations, to be more efficient and the results obtained were good enough to help us make a choice.

VI. Observations

As we can observe in the Table I, the combination of techniques that achieve the highest accuracy is:

- SDC
- VAD
- CMVN

The final model was trained with a maximum iteration of 100 (enough so that all the models could converge), and it achieved a accuracy of 53 %

We can also observe that some combinations of features can have a negative impact on the accuracy achieved. As an example, the approach Wiener+SDC+VAD+CMVN has an accuracy of 49% and the approach Pre-Emphasis+SDC+VAD+CMVN has an accuracy of 51% which are both good, but when combined, they achieve a worse accuracy (47%). But, if we add Centroid+Roll-Off+ZCR to the latter combination, we obtain one of the highest accuracies achieved (51%).

VII. Conclusion

In conclusion, the final pipeline obtained is: SDC, VAD and CMVN for feature extraction and a 6 GMMs with 64 components and 100 maximum iterations for the classification model.

We can state that the combination of certain features can have a negative impact on the accuracy achieved (*e.g.*, Wiener+SDC+VAD+CMVN vs Pre-Emphasis+SDC+VAD+CMVN vs Pre-Emphasis+Wiener+SDC+VAD+CMVN).

Having in mind the accuracy obtained with the X-vectors (83%) and the accuracy obtained with our model (53%), we can also state that, despite the techniques used to extract features from the audio, the approach of extracting a embedding of an audio with an x-vector is far superior, since the model responsible for generating the embeddings is a pre-trained model that has trained over a huge datasest.

We would have wanted to train our final model with the full dataset, but due to resources limits (namely RAM) we could not do it. Despite of that, we think that the model trained with the partition *train100* is a good approximation of a fully trained model.

References

[1] https://en.wikipedia.org/wiki/Spectral_centroid
[2] https://en.wikipedia.org/wiki/Cepstral_mean_and_variance_normalization
[3] https://www.audeering.com/research/opensmile/
[4] https://huggingface.co/speechbrain/spkrec-xvect-voxceleb

TABLE I
ACCURACIES OBTAINED WITH DIFFERENT COMBINATION OF STRATEGIES

Pre-Emphasis	Wiener	Delta Order	SDC	Centroid	Roll Off	ZCR	VAD	CMVN	Accuracy (%)
X	X		X	X	X	X	X	X	51
X			X	X	X	X	X	X	47
			X	X	X	X	X	X	44
X			X		X		X	X	47
			X		X		X	X	48
			X	X	X	X	X	X	46
							X	X	37
		X					X	X	43
			X				X	X	52
	X	X					X	X	39
X		X					X	X	42
X	X	X					X	X	38
	X		X				X	X	49
X			X				X	X	51
X	X		X				X	X	47

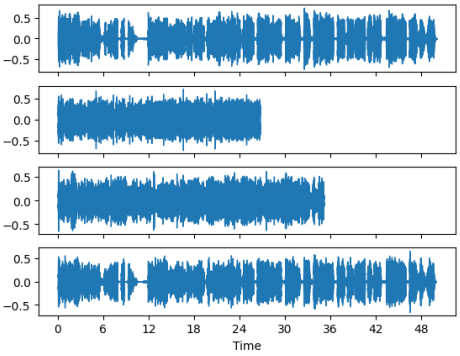

Fig. 2. Comparison between the original sound and the result after applying VAD-RMS with different values for threshold: *2)* 0.1, *3)* 0.05, *4)* 0.001

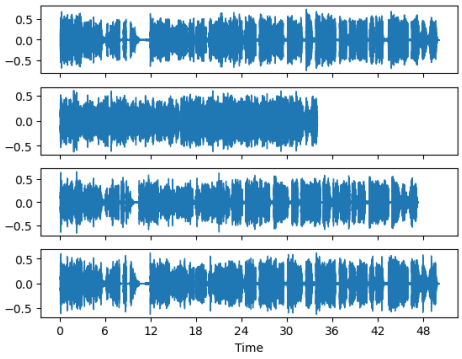

Fig. 4. Comparison between the original sound and the result after applying threshold with different values: *2)* -200, *3)* -300, *4)* -400

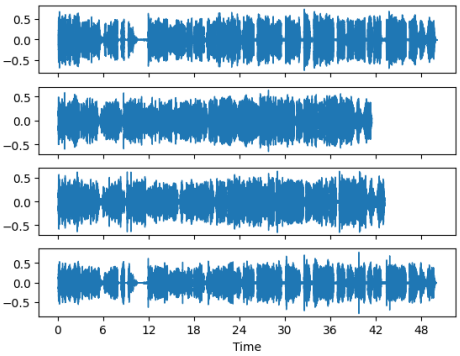

Fig. 3. Comparison between the original sound and the result after applying VAD-SS with different values for threshold: *2)* 0.1, *3)* 0.05, *4)* 0.001

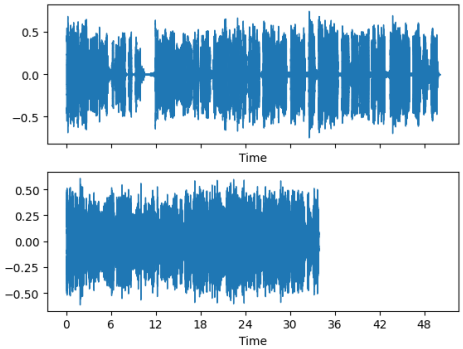

Fig. 5. Comparison between the original sound and the result after applying VAD-GMM