Intent Detection Classification Problem

[redacted]

Luca Pesce
Politecnico di Torino
[redacted]

***Abstract*—In this article we present a potential solution to the Intent Detection Classification Problem. The suggested method entails, as first step, the preprocessing of the audio files assigned, followed by the retrieval of several features, such as the extraction of MFCCs (Mel Frequency Coepstral coefficients). The features retrieved were consequently inserted in two classification algorithms, enabling the performance to be more accurate than the naive baseline and to yield generally positive outcomes.**

I. Problem overview

The subject of the proposed competition is the classification of audio files in wav format present in a dataset. Specifically, the goal is to identify the intention expressed by the speaker in each audio.

The dataset is divided into two parts:

- A development set consisting of 9854 audio files in wav format whose intent is known.
- An evaluation set consisting of 1455 audio files with unlabelled intent to be classified.

With the first set we train the model that we will use to classify the second and consequently evaluate its performance. In addition, for each audio we also have other information, such as the speaker's gender and age range, which will not be taken into account for model construction.

By using the function *Librosa.load()* on the audios in the development set, we obtain the frequency of sampling and an array of numeric values. About the first variable, we can confirm that all recordings share the same sampling frequency of 22050 Hz, corresponding to one half the sampling rate of audio CDs. It is an ideal sampling frequency to work with as it is used for lower-quality PCM and MPEG audio and for audio analysis of low frequency energy. The array of numbers retrieved from *Librosa.load()* represents the amplitude in linear scale of the signal at a specific moment in the recording: by plotting some of the audio signals, as the one represented in figure 1, we see that most of them are characterized by periods of silence or low noise both at the beginning and at the end, and that in general they have different lengths. Therefore we have to modify the arrays by removing silences and adjusting them all to the same length. This allows us to go and extract the same number of features for each audio.

II. Proposed approach

A. Preprocessing

As said before, we have to modify the arrays by removing silences and adjusting them all to the same length: thanks to the function *librosa.trim()*, set with parameter **topdB**=30, which was found to be more efficient after different trials, we trimmed the values corresponding to the initial and ending silences, obtaining a more faithful and meaningful array. The length of all the audio files were still different, meaning that the arrays of the features that were going to be extracted would have been of different sizes: to prevent this we extracted the length of the longest audio array between both development and evaluation set and set it as a variable (*maximal_length* = 279552).

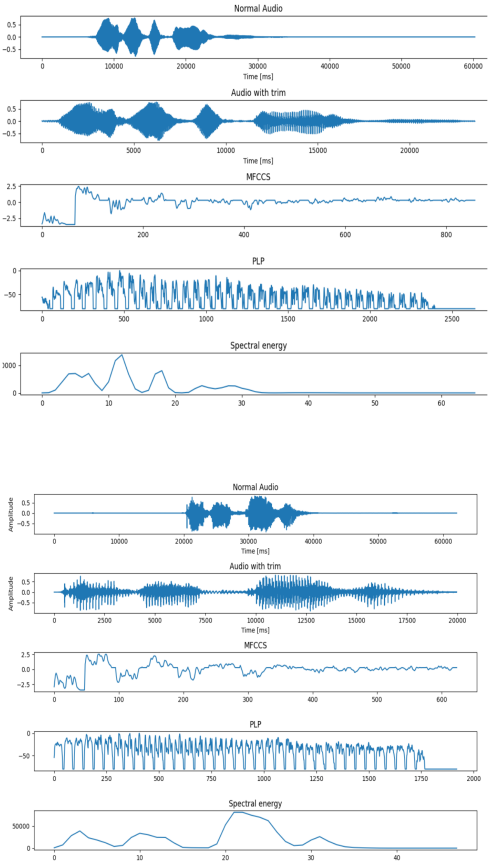

Fig. 1. Difference of all features between two audio signals

Then we calculated the difference between any audio and that specific long audio file: finally we used the *numpy.pad* function to all the other audios, in order to add constant

values (zeros) for the whole length of the difference previously calculated.

Moreover, due to the intrinsic nature of the task of speech recognition, we were initially driven to the research of features related to the voice of singular speakers: therefore, we chose to extract the pitch and the formant of the speaker of any audio. The pitch is the relative highness or lowness of a tone as perceived by the ear, and it can be associated with a number, which depends on the number of vibrations per second produced by the vocal cords. The formant, on the other hand, is the broad spectral maximum that results from an acoustic resonance of the human vocal tract. Both of these features can be extremely valuable, in fact they are utterly characteristic and distinctive for different people, due to the natural morphosis of their phonatory apparatus.

Regarding the composition and structure of the audio signal, we decided to choose other three different features, due to the fact that it was observed, from plotting the audio signal in the frequency domain with a dB-scale, that some frequencies were providing much more energy than others. Firstly, we extracted and normalized the MFCC (*Mel-frequency cepstral coefficients*), which are a set of coefficients that represent a short-term power spectrum of a sound in a compact, low-dimensional form: they turn out useful in speaker-independent speech recognition systems, being able to effectively capture the speech signal's spectral envelope and highlight its most important features.

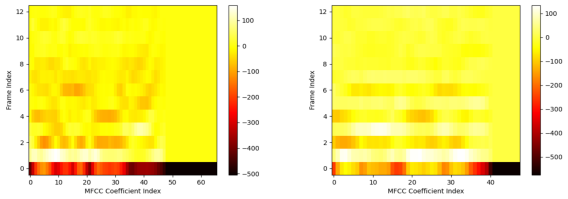

Fig. 2. MFCCs heatmap of two audio signals

For the number of MFCCS (which is a parameter of the function), after several trials we found out that 13 was the optimal number. The second feature is *Spectral Energy*, also known as short-term power spectral density, and it is used for speech recognition [1] because it represents compactly amount of energy present in different frequency bands of the speech signal, which can be used to differentiate between different speech sounds and to capture the prosodic and phonetic information of speech.

Finally, to the set of features it was added PLP (*Perceptual Linear Prediction*). It is a spectral analysis technique that models the spectral envelope of speech signals in terms of a linear prediction filter: its coefficients represent the spectral envelope of the speech signal, which can be used as features for speech recognition systems [2]. For calculating these values, the number of MELs (which is strictly related to MFCCs calculated before) was set to 13 in analogy to the parameter chosen before. It is possible to see, in figure 1, the evolution

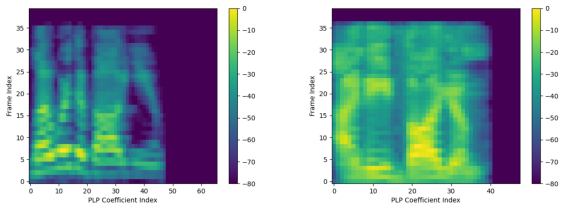

Fig. 3. PLP heatmap of two audio signal

of the audio signal preprocessing pipeline, comparing two different audio signals to better understand how peculiar the features extracted can be, related to a specific audio file: in particular, in figure 2 and 3 can be seen the differences of the signals with the MFCCs and the PLP plotted on a heatmap.

B. Model selection

For the classification issue, we chose two models. Firstly, we approached SVM, which, alongside with Neural Networks, is one of the best performers in terms of accuracy and efficiency: through a transformation of the data points, it is able to detect the hyperplane with the maximal margin in order to separate the different classes. Due to the large number of audio to classify and features to interpret, we matched it with a PCA decomposer on both the development set and evaluation: without it, the computation time would have been highly vast. The second model we used was the Random Forest classifier, which ensures the presence and cooperation of multiple decision trees to make predictions. We opted for this strategy not only because it performs well in terms of accuracy and efficiency, but also is able to outmatch the Decision Tree classifier regarding problems related to overfitting. As mentioned before, also this model was matched with a PCA on all the features chosen.

C. Hyperparameters tuning

Regarding hyperparameter tuning, it was done two times with *Randomized Search* over specific parameters [3]. The first search was regarding only the parameter *n_estimators*, which is the number of trees in the forest, and it was set in range (between 100 and 2000) with a number of iterations equal to 50. The second search we used also applied the tuning to *n_estimators*, but it was set on a smaller range, between 50 and 200, while *min_samples_split*, which is the minimum number of samples required to split an internal node, was set on a range between 1 and 20, *min_samples_leaf*, which is the minimum number of samples required to be at a leaf node, was set on a range between 1 and 10, and *criterion*, which describes the function that measures the quality of a split, was set to test all the three possibilities (*"gini", "entropy"* and *"log_loss"*). Since we wanted the Randomized Search to be as close as possible to a Grid Search, without having to check all the possible combinations, we set the number of iterations for the Randomized Search to 200.

III. Results

Before overgoing the process of *hyperparameter tuning* we tried both the models with default settings: Random Forest was outperforming SVM (0.59 against 0.21): even when the n parameter of PCA (setted at the beginning equal to 150) was modified, the SVM performance was clearly smaller, so we kept the focus of the research only on the first model. When we got to apply the hyperparameter extracted on the first Randomized Search on the Random Forest, which was *n_estimators* = 1500, we got as result an accuracy of 0.62, which was more performing than the one achieved with default settings. Later on, we applied the results of the second Randomized Search, which in particular were: *n_estimators*=115, *min_samples_split*=4, *min_samples_leaf*=1, *criterion*= *"gini"*. The accuracy of the Random Forest classifier, unfortunately, decreased by 0.03 (and it was comparable to the default settings), so we chose to comply with the first extraction of hyperparameters (*n_estimators* = 1500): in figure 4 are underlined the differences between the models just mentioned.

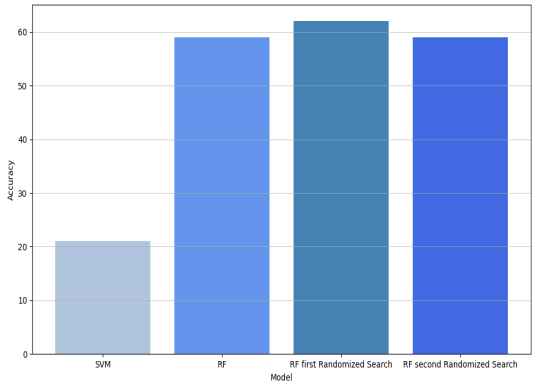

Fig. 4. Differences of models in terms of accuracy

In order to be able to increase again the accuracy score, we worked on the features and on the number of components of PCA. To start, all the accuracy results achieved before were involving all the features, that where concatenated on a set, in order to have a series of values for each audio file (approximatively 15 thousand): several tries were done to find the best combination. Initially, the pitch and formant features were eliminated, due to the fact that without them, the model was providing an accuracy of 0.66. Consequently, we tried the three possible combinations between the remaining distinctive features, which were in order MFCCs, Spectral energy (SE) and PLP, but none of them was performing as good as the subset containing the three of them: figure 5 sums up the different attempts and their singular accuracy scores. The last attempt to increase the score was tried on the PCA: by doing a for cycle which was testing different values of PCA (in the range between 10 and 200 for the number of components), we were able to detect that n=60 was the optimal solution, achieving the **final accuracy of 0.73**.

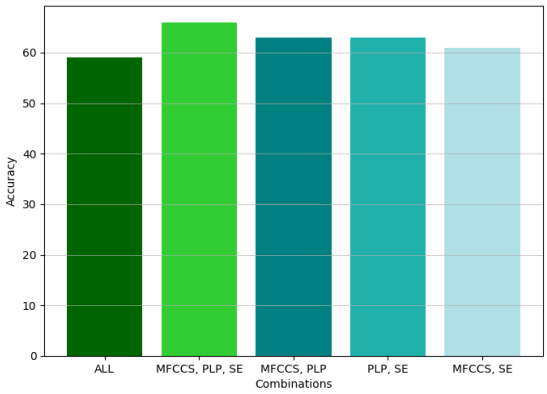

Fig. 5. Differences of combinations in terms of accuracy

IV. Discussion

Thanks to the exploitation of features that are specific of the singular audio signal and are also characteristic in term of time flow, the suggested process is able to provide an accuracy score that is higher than the naive baseline. Unfortunately we were able to perform acceptable scores only on one model, but with further examination of parameters and hyperparameters regarding the SVM classifier (that has been rejected as soon as the results of the Random Forest were definitely preferable), also this model could reach acceptable and finer results. Though its accuracy is sufficient, there is still a considering room of improvement to the process, and some other approaches that can be implemented to increase the score could be:

- Instead of two Randomized Search on a limited hyperparameters set, it could be exploited a Grid Search process on a much larger set to tune, with even larger ranges of values.
- The development of a process consisting in Convolutional Neural Networks or Recurrent Neural Networks, which could both be adequate for the task of speech recognition in this classification problem.

References

[1] J. Shen, J. Hung, L. Lee, Robust Entropy-based Endpoint Detection for Speech Recognition in Noisy Environments. 5th International Conference on Spoken Language Processing (ICSLP 98) Sydney, Australia, 1998.

[2] I. Mporas, T. Ganchev, M. Siafarikas, N. Fakotakis, Comparison of Speech Features on the Speech Recognition Task. Journal of Computer Science 3 (8): 608-616, 2007.

[3] W. Koehrsen, Hyperparameter tuning the random forest in python. Medium, Towards Data Science, 2018.