A study for command speech recognition using MFCC

Simone Clemente
Politecnico di Torino
Student Id: s309291
s309291@studenti.polito.it

Fabrizio Gallo
Politecnico di Torino
Student Id: s315338
s315338@studenti.polito.it

***Abstract*—In this report we address a speech recognition classification task through the implementation of the MFCC analysis. The entire pipeline, from the preprocessing phase to the hyperparameters tuning, is illustrated. We compare several models discussing the obtained results.**

I. Problem overview

The proposed task is a classification problem on a data set containing information on spoken commands, which are often used for digital assistants and Iot applications. The goal is to correctly classify each command through a label composed by the *action* and the *object* which it is referred to.
The data set is composed by:

- A development set of 9,854 objects characterized by information related to the speaker (fluency, first language, current language, sex, age range), the audio in .wav format and the action and object labels.
- An evaluation set of 1,455 objects with the same structure of the development apart from the labels that has to be predicted.

Each record is composed by 2 parts: the audio signal and some additional attributes referred to the speaker.

TABLE I
KEY ATTRIBUTES DESCRIPTION

Feature	Type	Cardinality Dev	Cardinality Ev
Gender	Nominal	2	2
Age Range	Ordinal	3	2
Current Lang	Nominal	4	1
First Lang	Nominal	5	1
Fluency	Ordinal	5	1
Speaker Id	Nominal	87	10

From analysis on the data set it results that no development or evaluation data is missing, while on the other hand, the cardinality of some attributes in the evaluation set decreases compared to the development one (Table I). According to this analysis, all the features related to the language level have to be discarded, while the gender information can be kept for further processing.
Observing the distribution of length of the audio files we can highlight a log-normal curve: through this visualization analysis we can detect some outliers in the upper and lower bound (Figure 1).

It is important to highlight that the sampling rate is not constant, varying between 16 kHz and 22.05 kHz.
Each audio file contains a time-series of the signal intensity for each instant recorded, hence showing a graph in time domain with peaks and troughs (Figure 3). To better understand the analysis that has to be done, it is crucial to address the distribution of the class labels (Figure 2). The data set is not balanced over the several labels; this aspect could have an influence on the analysis results: as a matter of fact volume commands appear at least twice compared to the others.

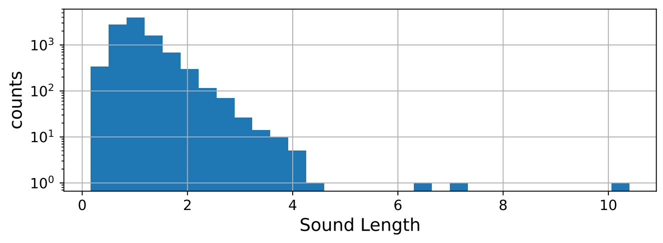

Fig. 1. Distribution of audio signal length

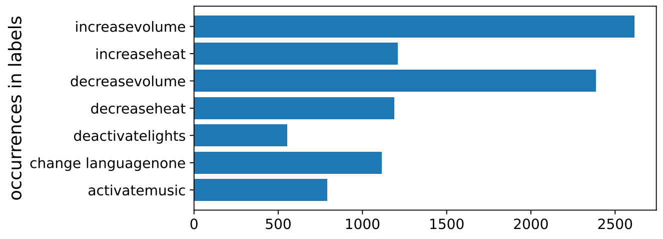

Fig. 2. Distribution of class labels

II. Proposed approach

A. Preprocessing

The files we are working with have several critical aspects which have to be addressed in order to make them suitable for a classification model.
To approach the preprocessing, we perform an audio trim on the original signal: the audio file is presented as an array of variable length and the silences at the beginning and at the end of each file are dropped based on a 20 dB threshold.

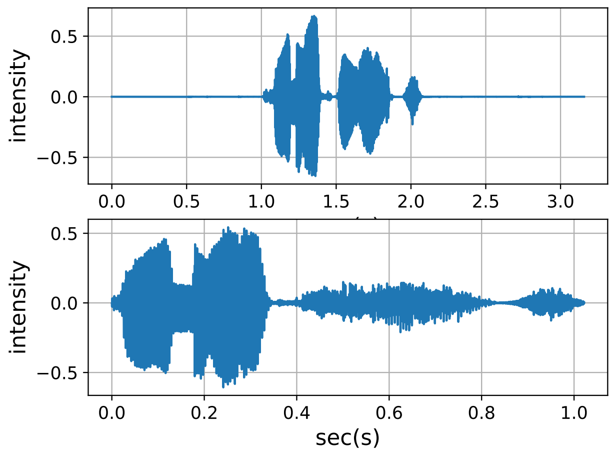

Fig. 3. Comparison between female original (on the top) audio and processed signal (below).

This process helps us reducing the arrays size but, more importantly, we can deal with more uniform audio sequences containing only the spoken part (Figure 3).
Once the silences removal has been performed, we can proceed with outliers elimination: audio files which are too long or too short are removed since their content is not considered useful for the classification task. Then we applied a data augmentation process based on the speaker's gender: women voices have been decreased by 1 octave since they are usually more high-pitched. This technique aims to standardize the files making the voices of the two genders more similar to each other and it should helps with the classification performances [1]. Further methods, such as noise cancellation, have been taken into account but they have been discarded since they had no impact on the data quality.
There are several ways to address the speech classification tasks: we could use time domain or frequency domain, but in order to make the most out of the audio signal, it is suggested to combine them. To merge the two domains a spectrogram can be used, but a more accurate feature selection can be yielded by the MFCC feature extraction. This method recalls the study of how the human ear perceives the sounds [2].

We start from the pre-emphasis part, where the lowest energy frequencies are enhanced to highlight a part of the audio signal that the human ear would perceive more. Following with MFCC road map the audio file is divided into chunks of specified length in time: the number of bins will depend on the FFT window dimension [3]. Once in the frequency domain, the feature extraction passes the split audio content to a log function which transform the signal in Mel frequency scale:

$$mel(f) = 1127 \ln{(1 + \frac{f}{700})}$$

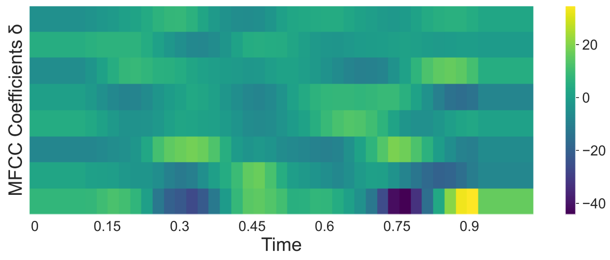

Fig. 4. MFCC first derivative plot

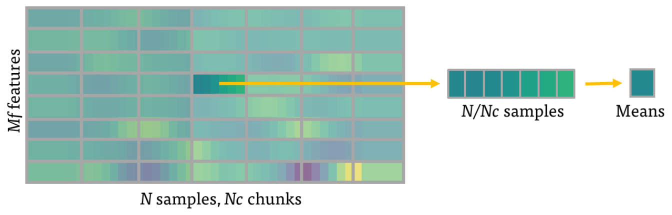

Fig. 5. MFCC derivative matrix chunks division

Subsequently, we exploit the IDFT function, an inverse transformation which highlights the importance of the low frequencies cited before. This part is used to retrieve the so called *"cepstrum"* coefficients, which are calculated comparing the audio data with a Mel bank of maximum 12 filters.

Along with the features extracted from the time-frequency analysis, the MFCC adds a feature about the energy of the audio signal for a total of 13 original features.

$$Energy = \sum_{t=t1}^{t2} x^2[t]$$

We decided to use the first derivative of the MFCC (Figure 4) since it is a method which is frequently used in this kind of tasks providing interesting results [4]. In order to insert the obtained matrices inside a data frame structure, we need to transform them into 1-dimensional arrays. Since each matrix has a different amount of columns, we divide each row in Nc equal length bins in order to extract the mean from each one. After that we reshape the matrix as a single-dimension array which can be fed to the models (Figure 5). A normalization of the data is required as we are going to work with models which requires it: we insert a Max-Min Scaler, which scales the data set based on a range from -1 (Min) to +1 (Max). The last step is represented by the features selection: a PCA is applied to the data set to choose the most important features, reducing the processing time and increasing data quality [5]. After this process we can build a data frame structure in which each row represents the features of the correspondent audio file.

B. Model selection

Different classification algorithms can be considered in order to address this problem:

- ***K-Nearest Neighbors***: this approach computes all the distances from the data points in the training set, assigning the most common label out of the k nearest point. The idea could be useful because the data set is numerical and it is also normalized. The performance of the method is comparable to other techniques, but it is unstable with a large number of features because of the curse of dimensionality.
- ***Random Forest Classifier***: this algorithm exploits more trees that are developed from random bootstraps of the records in the data set; a random selection of subfeatures is analyzed at each split in order to find the best configuration. This method results robust to outliers and it is able to work with large number of features.
- ***Extra Trees Classifier***: this classifier is very similar to the Random Forest but, instead of computing the best split point, it chooses it randomly (the best of the randomly generated splits is then evaluated and selected). Furthermore, by default it does not use bootstrapping but it considers the whole data set. Performances are usually similar to Random Forest Classifier but there are reduced bias and less variance in the results. The computational time is reduced and it can be interesting for data exploration.
- ***SVM***: this method is largely used for speech recognition tasks. The classifier is built based on a risk minimization goal. The model defines an hyperplane that is powerful to separate clearly each class [6].This can be done by a linear or non-linear model based on the data provided. This Kernel-based technique guarantees a zero-error on the separation of each class. This makes the SVM a potential powerful machine to achieve the goal of this task.

To coherently elect the right model, an hyperparameter tuning is performed for each classifier through different grid searches and trials.

C. Hyperparameters tuning

To perform the hyperparamters tuning selection we need to work on the choices of:

- The value of Nc and Mf mentioned during the preprocessing.
- The parameters for the different methods previously stated.

The first values are chosen through a grid search done using a sample algorithm for data exploration. Instead, for what concerns the grid search for the classifiers, it has been established a different sequence of values to be tested. The approach to verify the hyperparameters has been made with a cross validation with size 5.

TABLE II
HYPERPARAMETERS CONSIDERED

Model	Parameters	Values
Preprocessing	Nc Mf	$[8, 10, 12]$ $[25, 30, 35]$
Random Forest	$n\_estimators$ $criterion$ $max\_features$	$[50, 100, 200, 500]$ $'gini', 'entropy'$ $'log2', 'sqrt'$
Extra Trees	$n\_estimators$ $criterion$ $max\_features$	$[50, 100, 200, 500]$ $'gini', 'entropy'$ $'log2', 'sqrt'$
KNN	$n\_neighbors$ $weights$ p	$[5, 9, 11, 13, 15]$ $'uniform', 'distance'$ $[1, 2]$
SVM	$kernel$ C $gamma$	$'linear', 'rbf', 'poly'$ $[1, 5, 10]$ $[0.1, 1, 10]$

III. Results

First of all, we obtain as best values Nc=25 and Mf=8, for a total of 200 original features which are reduced to 100 using PCA. Then we proceed with the hyperparameters tuning of each algorithm separately, in order to find the best configuration for each model. Since we were dealing with an unbalanced data set, we consider as scoring metrics not only accuracy (which is the core request of this task) but also Macro F1 score.
In the Table III is provided a summary of our results obtained over 5 rounds:

TABLE III
BEST CONFIGURATIONS AND RESULT

Model	Best configuration	Local Scores
Random Forest	$n\_estimators : 500$ $criterion : entropy$ $max\_features : sqrt$	$Accuracy$ = 0.626 $MacroF1$ = 0.590 $FitTime$ = 31.1s
Extra Trees	$n\_estimators : 500$ $criterion : gini$ $max\_features : sqrt$	$Accuracy$ = 0.624 $MacroF1$ = 0.571 $FitTime$ = 4.41s
KNN	$n\_neighbors : 11$ $weights : distance$ $p : 2$	$Accuracy$ = 0.759 $MacroF1$ = 0.757 $FitTime$ = 0.071s
SVM	$kernel : rbf$ $C : 5$ $gamma : 0.1$	$Accuracy$ = 0.805 $MacroF1$ = 0.801 $FitTime$ = 6.24s

Overall, SVM is the best performer with an average accuracy around 0.8 and its training phase takes a small amount of time. KNN is the fastest and it is the one with the most constant behaviour. Instead, the two tree-based algorithms do not perform well since they show similar accuracy around 0.62: this is not a convenient approach considering that they have a longer training time and low F1 Score.

To summarize, KNN provides a stable result throughout the different trials but SVM performs generally better in any case. The distributions of the results of the different models are clearly displayed in the Figure 6.
The public scores strongly improves over the local ones since models are trained over the entire development data set: this results improvement can be observed through a learning curve (Figure 7).
After completing all the local runs, the two best algorithms have been tested on the public leaderboard: SVM provided the best overall public score of **0.942** while KNN reached **0.891**. For comparison we decided to run some tests with the tree-based algorithms: Random Forests reached **0.755** as best result on the leaderboard, while the public score of the Extra Trees Classifier peaked at **0.747**.

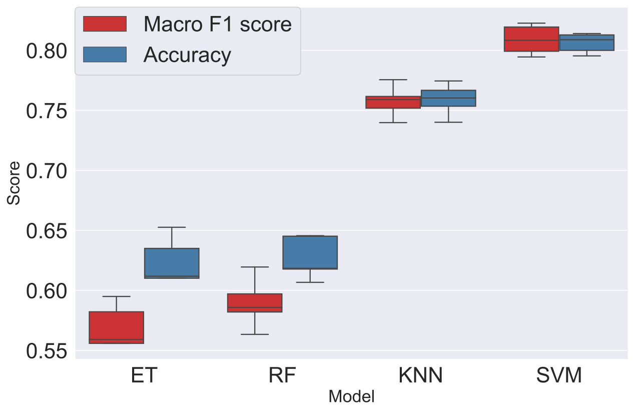

Fig. 6. Models performance distribution

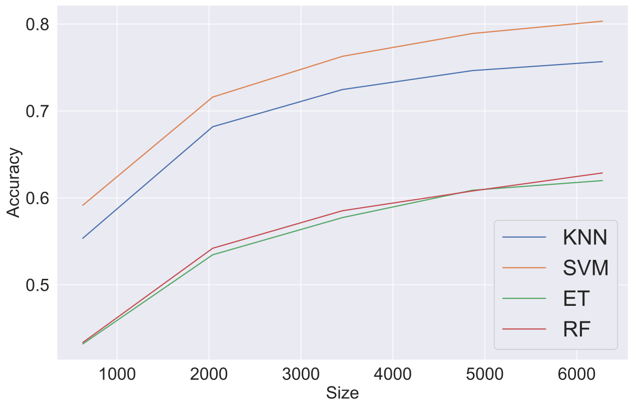

Fig. 7. Models learning curve

IV. Discussion

The proposed approach obtains results that far outperforms the naïve baseline defined in the public results board.

Even though we were provided with an unbalanced data set, the model achieves a F1 Score which is in line with the accuracy meaning that each class is predicted well. Here are some points worth considering for further improvements and implementations:

- The adoption of the MFCC first derivative increases the accuracy by around 2%. Further analysis on the second derivative could lead to better results, but in our configuration the first derivative alone looked like the best option.
- Considering the nature of the task in which we have an action and an object, CNN could be indicated as the best method: thanks to neural network we are able to divide these two parts focusing more on the semantic of the sentence and less on its structure. As a matter of facts, this is the method which is usually implemented in this branch of speech recognition. However, these models require a huge hyperparameter tuning and the training phase could take a long time: in our case we considered a trade off with performances obtaining a training phase around 10/15 seconds (running on a 7th gen intel i7 processor).

To conclude, we are satisfied with the results we obtained and we believe that they are in line with the scope of this task. Our idea was to select a method able to find a trade-off between performances and computational complexity: the use of the SVM in pipeline with the MFCC analysis seems to address this assignment in a suitable way.

References

[1] R. Sinha, S. Shahnawazuddin, and P. S. Karthik, "Exploring the role of pitch-adaptive cepstral features in context of children's mismatched asr," in *2016 International Conference on Signal Processing and Communications (SPCOM)*, pp. 1–5, 2016.
[2] S. D. Dhingra, G. Nijhawan, and P. Pandit, "Isolated speech recognition using mfcc and dtw," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 2, no. 8, pp. 4085–4092, 2013.
[3] W. Han, C.-F. Chan, C.-S. Choy, and K.-P. Pun, "An efficient mfcc extraction method in speech recognition," in *2006 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 4 pp.–, 2006.
[4] F. Zheng, G. Zhang, and Z. Song, "Comparison of different implementations of mfcc," *Journal of Computer science and Technology*, vol. 16, pp. 582–589, 2001.
[5] M. Cutajar, E. Gatt, I. Grech, O. Casha, and J. Micallef, "Comparative study of automatic speech recognition techniques," *IET Signal Processing*, vol. 7, no. 1, pp. 25–46, 2013.
[6] R. Thiruvengatanadhan, "Speech recognition using svm," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 9, pp. 918–921, 2018.