Intent detection classifier for audio-based applications

Riccardo Zanchetta, Valerio Mastrianni
Politecnico di Torino
Student id: s313344, Student id: s308781
s313344@studenti.polito.it, s308781@studenti.polito.it

***Abstract*—In this report, we aim to tackle the problem of intent detection in audio-based applications. The proposed approach consists in extracting the mel-frequency cepstrum for each audio signal and split it into a specific number of chunks. Using the mean and resizing the different matrices to a common shape we were able to provide an input for two classification models. The proposed approach achieves good results and outperforms the baseline proposed for the project.**

I. Problem overview

The aim of this project is to classify audio files contained in a dataset, made by speech recordings of people talking with vocal assistants. The final goal is to correctly identify the underlying intention in each recording.
The dataset is divided in two parts:

- a *development set*, containing 9854 recordings labeled with the action and the object of the audio file.
- an *evaluation set*, containing 1455 recordings without labels.

Both datasets contain additional information about the speaker, such as: self-reported fluency level, first language spoken, current language used for work/school, gender and age range. We will need to use the developing set to build a classification model to correctly identify and label the recordings in the evaluation set.
Some consideration can be made analyzing the development set. First of all, the problem is not well-balanced: for each of the 7 actions to classify there are not an equal number of recordings as show in Figure 1. Second, the sample width is of 16 bits for the majority of the files but in some cases it is equal to 22 bits. Third, the language of the speakers in not always the same: in the dataset is either English or Spanish.
All the signals have a different duration as shown in Figure 2. The large difference of signals length is due to the phrase formulation being more or less articulated and the presence/absence of periods of silence.

II. Proposed approach

The proposed approach is a well-established method, for which it is possible to find several references in literature [1], [2]. The audio processing pipeline was developed in different stages. First of all, a lot of data preprocessing was needed in order to solve issues caused by the presence of outliers and improving the audio signal for further analysis. Dealing with audio signals is not always that easy, a basic knowledge of

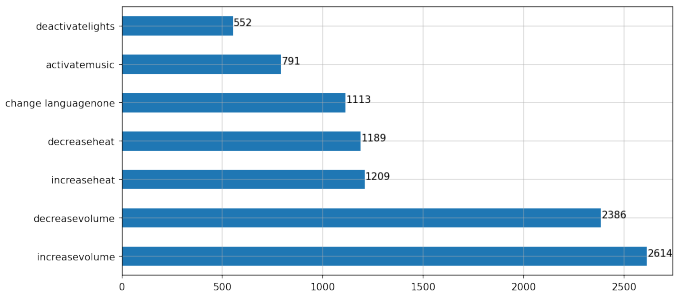

Fig. 1. Distribution of action/object on training dataset

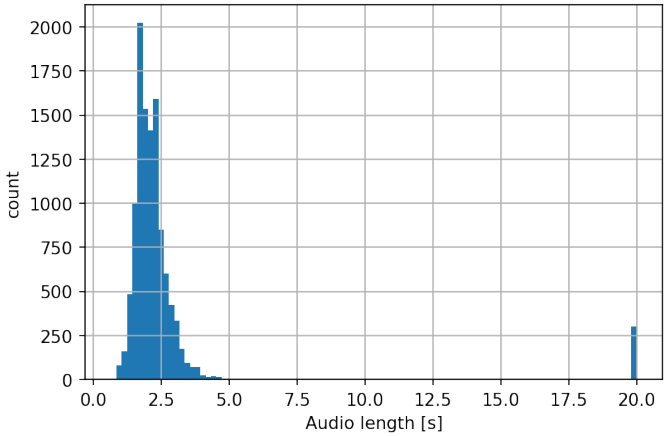

Fig. 2. Distribution of the duration of the recordings

audio features is required. In Figure 2a is reported a typical waveform of an audio file. It's clearly visible that the peaks and the valleys are linked with pronounced words. In order to achieve good results it is crucial to select and extract some features from the raw audio file.

A. Preprocessing

As stated before, the files into the dataset present some outliers and are distributed in a heterogeneous way. Before performing the feature extraction a preprocessing phase is needed. It involves cleaning and preparing the audio files for analysis, and includes the following steps:

- **Noise reduction**: The first step in audio preprocessing is to reduce or remove background noise from the audio

TABLE I
LABELS OF AUDIO FILES

Action	Object
Change Language	-
Activate	Music
Deactivate	Lights
Increase	Volume
Decrease	Heat

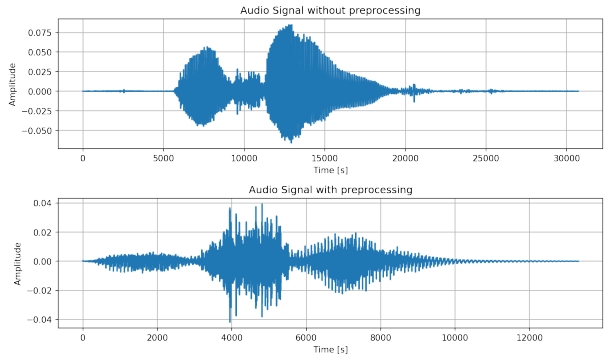

Fig. 3. [2a] on top, [2b] on bottom
Representation of a recording in the time domain

files. This can be done using various techniques such as spectral gating [3]. The raw audio files were recorded with low quality microphones and there is a lot of background noise such as traffic.

- **Silence removal**: the duration of the audio files presents some outliers as shown in Figure 2, after a further analysis, we noticed that in most cases, the audio files with a duration greater than 5 seconds contained tail silence. We applied the "trim" function provided by *Librosa*, to remove the silence and set the silence threshold at 20db.
- **Pre-emphasis**: by emphasizing the high-frequency components of the audio signal, pre-emphasis can improve the frequency resolution of the signal, that helps to better capture the characteristics of speech. We computed Mel-frequency cepstral coefficients (MFCCs) [4], that are calculated on the log-power spectrum of the audio signal. By emphasizing the high-frequency components of the audio signal, pre-emphasis can improve the accuracy of the log-power spectrum and thus the accuracy of the feature extraction. Most of the speech energy is concentrated in the high-frequency range, as a result pre-emphasis helps to amplify these high-frequency components, making the speech signal stand out more.

The selection of the feature set to be used is a significant issue for all recognition systems. Pitch, energy, durations, tunes, spectral characteristics, intensity, and other factors have all been explored in several researches [5].
The frequency domain contains the features that are most useful for speech processing. In the frequency domain contrary to the time domain, the spectrum analysis of the speech signal is more reliable and simple. The spectrum magnitude seems to be far more important to the hearing process than the phase or time characteristics. When dealing with machine learning models there are issues with data structure, since we need to feed our model with standard-size data, we can not insert data with different shapes into the model. The raw data and the features computed on them are of variable size.
MFCC, is a matrix with a constant number of rows but a variable number of columns, depending on the length of the audio file. To obtain uniform data without losing too many information we applied a technique that is very common in image processing: resizing [6]. Given an input matrix $M \times N$ we can split it into submatrices $m \times n$ and then compute some statistics on each chunk. After many trials we noticed that extracting both mean and standard deviation, did not improve the accuracy of the model and, for that reason, we extracted only the mean values.
The resize function allows us to handle input matrices of different cardinality through interpolation. This technique is useful to avoid information loss that fundamentally would occur if padding techniques were chosen. After plotting the distribution of the number of columns N of each MFCC matrix, we noticed that the mean value was 136, for that reason we resized all the MFCCs to that value. At the end of the resizing process we obtained identical matrices of M rows and N columns, where M is equal to the number of cepstrums computed by MFCC and N is equal to 136. In Figure 4 is reported an example of how a matrix of large cardinality with n columns (n is a number that changes for each audio file, depending on the duration) is converted into a matrix of smaller cardinality with a constant number of columns and rows.
The last step was the encoding of the categorical values previously cited, we used the one-hot encoding to deal with it. This process was a good workaround because it allowed us to store all the data needed without increasing the dataframe.

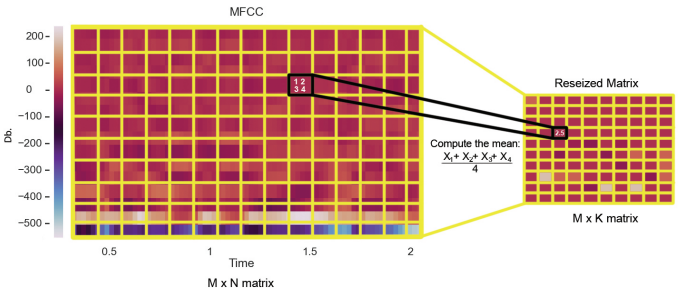

Fig. 4. Visual example of how the resize process works. The matrix on the left side is a MFCC, with M rows and N columns. M is constant, N is variable, depending on the duration of the audio file. The Resize phase converts the matrix $M \times N$ to another matrix of $M \times K$, both M and K are constant.

B. Model selection

Most of the commercial speech recognition algorithms are based on Deep Learning and Neural Networks, in this project we explored some well known Machine Learning models, achieving different results. In the first phase, we applied Random Forest [7], after many trials and hyperparameters tuning we achieved better results using Support Vector Machine (SVM) [1]. The choice of the right model is a crucial aspect for the project, in fact it can heavily affect the accuracy of the prediction. In literature there are several approaches to this problem, an interesting one is shown in Aravind Ganapathiraju and Joseph Picone's work [1]. Support Vector Machines (SVMs) are a powerful, discriminatively-trained technique that has been shown to work well on a variety of tasks. However, they are typically only applied to static binary classification tasks. An SVM is one type of classifier that directly estimates decision surfaces instead of simulating a probability distribution over the training set of data. SVMs have shown strong performance on a number of well-known pattern recognition issues. The definition of an SVM classifier is in terms of the training examples, however, not all training instances are used in the classifier's definition. In practice, the proportion of support vectors is small, making the classifier sparse. The complexity of the classifier depends on the data set itself. On the other hand, the complexity of systems like neural networks and HMMs is often predefined or selected through a cross-validation procedure. Data that can only be distinguished using a nonlinear decision surface are usually involved in classification difficulties in the real world. In this instance, a kernel-based transformation is used for input data optimization.

C. Hyperparameters tuning

The Hyperparameter tuning has been one of the hardest phases of this project. We used an 80/20 train/test split on the development set and run a random search on it. At the beginning we noticed that the accuracy score was very low on both the investigated models (SVM and Random Forest), Random Forest (RF) was performing quite better than SVM. With 1500 trees in RF we achieved results on average 15-20% better than SVM. Analyzing RF's hyperparameters, we first started using n_estimators = 500 and then increased it until we reached a plateau around n_estimators = 1500. The maximum depth of the tree can greatly affect the performance of the model. A deeper tree can capture more information, but it is more likely to overfit the training data. We started setting max_depth = 10 to ensure the model to generalize well, but also avoid overfitting.

As stated before, on the first tests the SVM was led to poor results, that was both caused by a wrong way of preprocessing and transforming the data, and also because of inappropriate hyperparameters. At the beginning we computed the mean values of MFCC for each row in the matrix, in that way the lack of information was too large and the model was underperforming. After changing the preprocessing pipeline we achieved better results also on SVM. The most performance changer parameters are C, which controls the trade-off between maximizing the margin and minimizing the classification error, and kernel. A small value of C results in a wider margin but leads to more classification errors, while a larger value will result in a narrower margin and fewer errors. When C is big, the model tries to fit the training data as well as possible, even if that means fitting the noise in the data, which can lead to overfitting. Also in this case we observed that a greater C made the accuracy better but for C greater than 12 we reached a plateau. Another crucial hyperparameter is kernel. Kernel is a function that takes two inputs and returns their inner product. The kernel trick is a technique used in SVMs to transform the input data into a higher-dimensional feature space, where it is possible to find a linear decision boundary, a common choice for a kernel function in speech recognition is the radial basis function (rbf) kernel. It allows the model to capture non-linear relationships between the features and the target variable.

Another key aspect is the tuning of all those parameters that belong to the preprocessing phase, as we noticed that even a small change in them brings a big jump in accuracy score.

For computing MFCC, we applied the function provided by Python speech features, and since our dataset was made of short audio files we needed to modify standard parameters. Generally, a window length of around 20-30 milliseconds is suitable for speech signals, but for very short speech segments (less than 3 seconds) a window length of around 10 milliseconds resulted in being more appropriate. The window length determines the time resolution of the MFCCs, setting it to 10 milliseconds provides a higher time resolution and increases the capacity of capturing the rapid changes in the speech signal that occur within a short time frame. To avoid information loss we preferred to set window step to 5 milliseconds. The window step is usually set to be less than the window length, so that the windows overlap at least of 50%.

TABLE II
HYPERPARAMETER TURNING

Model	Parameter	Values
Preprocessing	*M,N*	*M=13 →26, N=13 → 136*
Random forest	*max_depth* *n_estimators*	{None, 200, 250, 500, 1000, 1500} {gini, entropy}
SVM	*C* *kernel*	{1,5,10,12,15,20} {rbf, sigmoid}

III. RESULTS

The tuning of the parameter N (N = Number of columns) is summarized in Table II. In Figure 5 we can immediately see that the SVM is more stable than RF as N increases. Both classifiers achieve satisfactory performance for N = 70, which we can select as a reasonable value (larger values would affect the SVM with no meaningful improvement for the random forest). With this value of N, we ran the random grid for both random forest and SVM. We decided to perform a randomized search

instead of a grid search due to computational complexity and long time processing required for the operation.

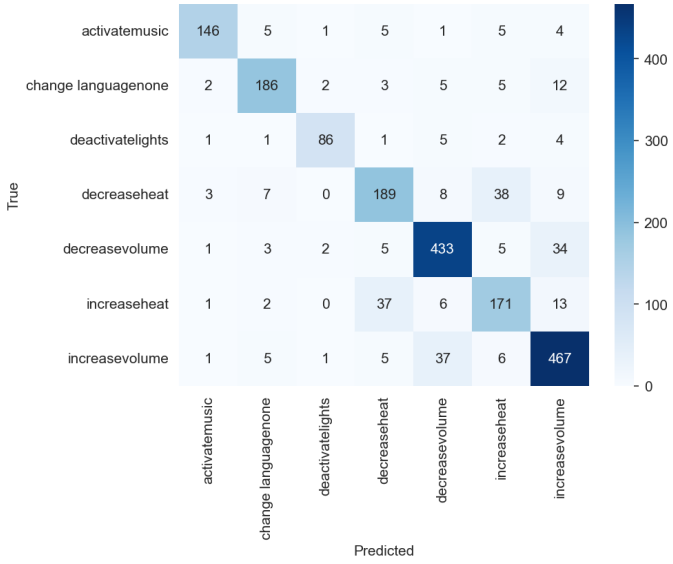

Fig. 5. Confusion Matrix. On the diagonal are reported the values that have been correctly labeled by the algorithm

The best configuration for RF was found for criterion=entropy, max depth=10 (*accuracy* score ≈ 0.67), whereas the best configuration for the SVM was found for C=12, kernel=rbf (*accuracy* score ≈ 0.87). The two classifiers achieve different results with SVM outperforming RF.
We trained SVM on all available development data. Then the models have been used to label the evaluation set. The public score obtained is 0.919. Since these are the first scores obtained using the evaluation data, there should be no overfitting. In Figure 5 is shown the confusion matrix that highlights some problems of classification: when there is an action (increase/decrease) involving an object, the model tends to make more errors than usual, leading to a poor classification.

IV. Discussion

The proposed approach achieved good results, the overall accuracy is quite high considering that the training of the model was performed on labels and not on audio transcription. There is a strong correlation between the accuracy of the model and the size of the MFCC matrix, therefore a further exploration of the optimal size of the matrix could lead to even better results.
As stated in the previous paragraphs, the hyperparameter tuning is a crucial task for the model, a grid search instead of a random grid would find the optimal tuning. In the proposed approach only RF and SVM models have been studied, there are several references in the literature that perform the same task using different techniques, such as Hidden Markow Model (HMM) [8] and Convolutional Neural Networks (CNN) [9].

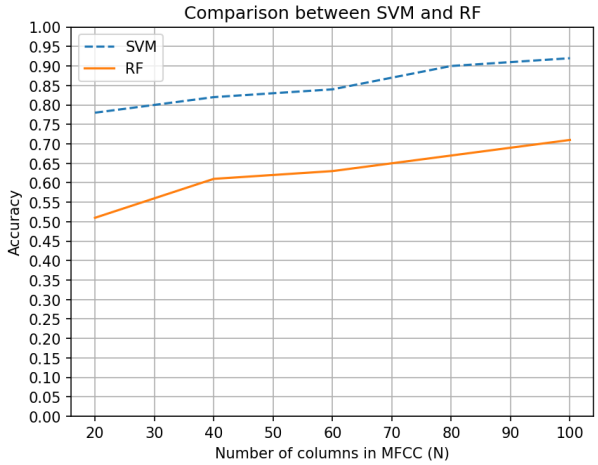

Fig. 6. Accuracy change of the two explored models as the number of columns N in the MFCC matrix changes.

CNN's are known to outperform all the other models in speech recognition tasks, despite having longer training time. A possible approach could use CNN's applied to the MFCC images obtained in the preprocessing phase to then train a CNN.
The classification task was successfully performed with a minor error rate, however the project leaves room for improvement that could be obtained with a larger dataset and CNN.

References

[1] A. Ganapathiraju, J. Hamaker, and J. Picone, "Hybrid svm/hmm architectures for speech recognition.," in *INTERSPEECH*, pp. 504–507, Citeseer, 2000.

[2] N. Smith and M. Gales, "Speech recognition using svms," in *Advances in Neural Information Processing Systems* (T. Dietterich, S. Becker, and Z. Ghahramani, eds.), vol. 14, MIT Press, 2001.

[3] J. M. Inouye, S. S. Blemker, and D. I. Inouye, "Towards undistorted and noise-free speech in an mri scanner: Correlation subtraction followed by spectral noise gating," 2014.

[4] M. R. Hasan, M. Jamil, M. Rahman, *et al.*, "Speaker identification using mel frequency cepstral coefficients," *variations*, vol. 1, no. 4, pp. 565–568, 2004.

[5] U. Shrawankar and V. M. Thakare, "Techniques for feature extraction in speech recognition system: A comparative study," *arXiv preprint arXiv:1305.1145*, 2013.

[6] P. Parsania, D. Virparia, *et al.*, "A review: Image interpolation techniques for image scaling," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 12, pp. 7409–7414, 2014.

[7] D. S. Siroky, "Navigating random forests and related advances in algorithmic modeling," 2009.

[8] B. H. Juang and L. R. Rabiner, "Hidden markov models for speech recognition," *Technometrics*, vol. 33, no. 3, pp. 251–272, 1991.

[9] D. Palaz, R. Collobert, *et al.*, "Analysis of cnn-based speech recognition system using raw speech as input," tech. rep., Idiap, 2015.