Project Assignment: Language Processing

Course of ”Data Science Lab: Process and methods”

January 2023

Savelli Claudio
S317680
Politecnico di Torino
Data Science and Engineering

Spaccavento Bruno
S314908
Politecnico di Torino
Data Science and Engineering

***Abstract*—In the following report, a classification of audio commands for a digital assistance device is proposed. The processes that were performed on the audio files for data cleaning, feature engineering, feature extraction and data augmentation are described next. The random forest classifier and the SVM classifier were used as classifiers. At the end the obtained results are discussed and possible improvements are proposed.**

***Index Terms*—Natural language processing, classification, Python, RFC, SVM**

I. Problem Overview

The goal of the competition deals with classifying certain commands that have been given to a digital assistant.

Within the dataset, the commands given are divided between two attributes, ”action” and ”object”, the former being ”change language”, ”activate”, ”deactivate”, ”increase”, ”decrease”; the latter being ”none”, ”music”, ”lights”, ”volume” and ”heat”.

The combinations within the dataset, however, are only a subset of all possible combinations of action and object. Specifically, analysing the combinations present, there are 7, and they are as follows: ”change language none”, ”activate music”, ”deactivate lights”, ”increase volume”, ”increase heat”.

Within the dataset there is some additional information, relating to the interlocutor’s original language and the language they currently speaks for work/study, their level of English, age and gender.

The dataset is divided in two parts:

- A *development set*, containing 9854 labeled recordings.
- An *evaluation set*, containing 1455 unlabeled recordings.

The development set was needed in order to build a classification model to correctly label the points in the evaluation set.

From Fig. 1, which represents the number of labels within the development set, it can be seen that the problem is very unbalanced, in fact there is a large difference between the most frequent combination, ”increase volume” = 2559, and the least frequent, ”deactivate lights” = 535. This imbalance led to the use of a data augmentation system, which will be described in detail in the following paragraphs.

Some of the files (300 of them) are with a sample rate of 22050, while all the others are 16000. For this reason it was decided to use, instead of using *scipy.io.wavfile.read*, *librosa.load*, which allows to convert files with a rate different from the default rate, which was set to 16000, being the most frequent one considering development set and evaluation set, into the rate defined.

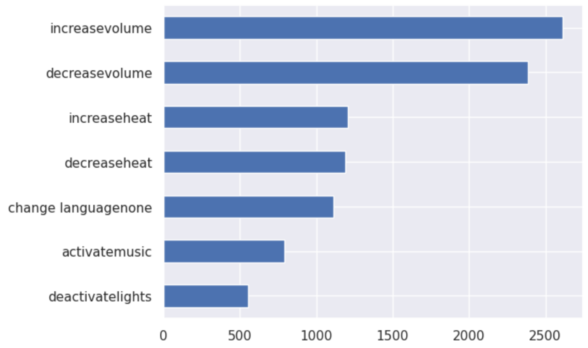

Fig. 1: Distribution in the development set of the different combinations of labels.

II. Proposed approach

A. Data preprocessing

Within the evaluation set, all the data to be analysed were done by native English speakers, which is not the case within the development set. For this reason, it was decided to analyse in detail these elements that differed from the evaluation set, in order to see whether it was still possible to obtain useful information from the latter or not. The distribution of this variation, as can be seen in Fig. 2, is very unbalanced, so that there are 9642 English (United States) against a total of 212 divided between English (Canada), French (Canada), Spanish (Venezuela) and Telugu. By manually listening to some of these audio files, it was noticed that some of them simply had a different accent from traditional English, while others (such as Telugu), are totally different. For this reason, given the data augmentation and the low number in relation, it was decided to eliminate these ’outliers’ and proceed to data processing.

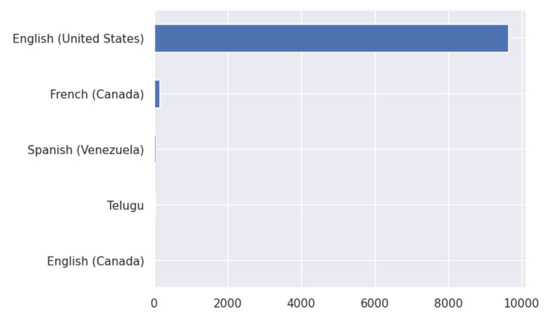

Fig. 2: Distribution in the development set of the different First Language Spoken.

As can be seen from Fig. 3, which displays the distribution of the lengths of the audio files analysed, it is possible to see that there are some outliers with a much longer length in seconds than the average. In fact, the longest audio is 20 seconds, while the 95th percentile of the distribution of audio file lengths is 3.58 seconds. Listening manually to the longer audio files, it was noticed that some of them were people mispronouncing the command and then, when finished, giving the right one, while others consisted of long silences followed by the request for the command or vice versa. For this reason, a silence removal function was used which cuts out the parts where the noise goes below 10db, which is considered silence [1]. After doing this, a new graph of the distributions was generated (Fig. 4), in which it can be seen that the longest audio is about 7 seconds with a 95th percentile 1.632 seconds.

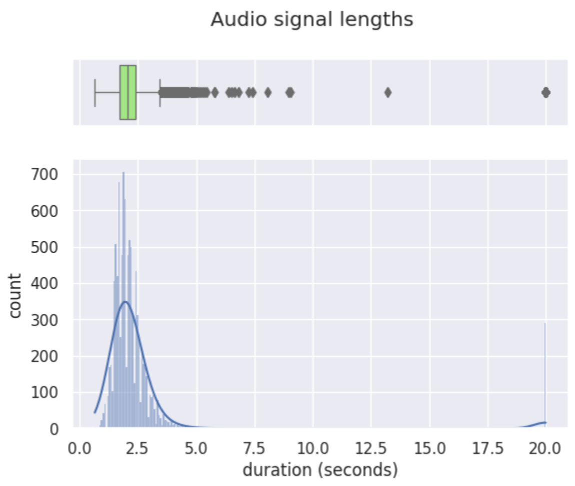

Fig. 3: Boxplot and distribution of the audio files' lengths **before** removing silence.

Given the large time difference between the outliers and the 95th percentile, again taking into account data augmentation a posteriori and after listening to some of the individual audios manually and realising that extracting information from them was difficult, it was decided to eliminate these outliers (amounting to 456).

In addition, in analysing the dataset it was noticed that some

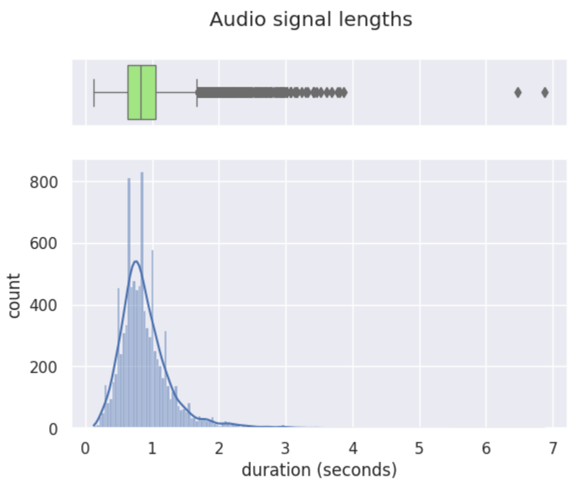

Fig. 4: Boxplot and distribution of the audio files' lengths **after** removing silence.

of the audios were less than 0.2 seconds in length, a time with which it was considered it impossible to issue a command. For this reason it was decided to manually listen to these audio files that were in fact just noise, and since these were useless for classification if not harmful, it was decided to remove them.

After doing so, to make the length of the audio files the same for all, all files shorter than 1.632 seconds were padded by adding zeros at the end to reach the desired length.

Data augmentation

With regard to data augmentation, as described in the previous paragraphs, it was exploited to solve the problem of unbalancing.

To do this, two data augmentation methods were implemented on the audio files with respect to the time domain, namely a variation of speed and pitch. In the first case, the speed of the audio file sampled was changed by a factor between 0.9 and 1.1, and then cut and padded after the augmentation. As far as pitch is concerned, the audio file pitch would have been multiplied by a factor between 0.9 and 1.1.

The method implemented for data augmentation, therefore, takes into account the entire dataset, counts the labels and, taking into account the label of maximum cardinality, equalises all others by generating half of the data by changing the pitch and the other half by changing the speed.

Feature extraction

Feature extraction was performed exclusively in the frequency domain, given the best results obtained both in the literature and in past projects.

1) Mel Spectrogram: Instead of performing a normal spectrogram extraction, an extraction of the spectrogram was performed following Mel's scale. In fact, Mel's scale is well suited to the problem's context, as it is designed to reflect the perceptive capacity of the human ear, which does not perceive sound in a linear manner with respect to frequency, but rather in a similarly logarithmic manner [3]. A mel spectrogram is, in fact, a spectrogram where the frequencies are converted to the mel scale. Next, the mel scale was converted to decibels. Finally, the PCEN, or per-channel energy normalisation, was applied, which is designed to suppress background noise and emphasise foreground signals and can be used as an alternative to the decibel scaling that was previously done [5]. Fig. 5 shows how the longest audio of the development set is transformed.

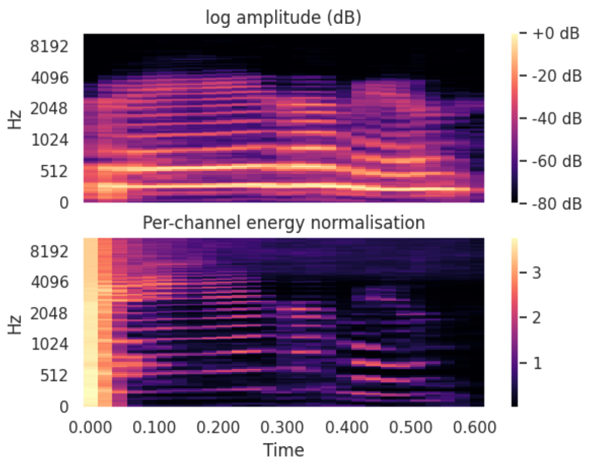

Fig. 5: Comparison between the normal log mel spectrum and its PCEN of the longest audio in development set.

2) MFCCs: The Mel-Frequency Cepstral Coefficients represent the extraction of the spectrum of a sound in a short period always based on the mel scale transformation. As described in the accompanying article [4], it is commonly used in speech recognition as people's voices are usually on a certain range of frequency and different from one to another.

3) Delta and Delta2 MFCCs: MFCCs are represented with a two-dimensional graph (like the spectrogram). Delta MFCCs and Delta2 MFCCs are the first and second derivatives of the MFCCs described above, respectively. They are used to show how MFCCs changes over time in each audio file.

4) RMS: The root-mean-square shows the energy and power of the audio file over time [4]. This can be very useful in speech recognition as the volume at which the person speaks varies depending on the emphasis they want to give and what they are saying, having moments of peak and moments of silence depending on the sentence being announced.

5) ZCR: In audio files the signal, representing the pressure on the microphone, oscillates between -1 and 1. The Zero-Crossing-Rate represents the frequency with which this analysed audio file varies from positive to negative and vice versa [4].

6) Spectral Centroid: The spectral centroid is a measurement that given a signal indicates the frequency band in which there is more concentrated energy. It measures the so-called brightness of the sound, that is how open or dull the sound is. Since this feature is related to timbre, it could be helpful for an eventual classification to better focus on the voice of the speaker rather than background noises.

7) Spectral Bandwidth: The spectral bandwidth is a derivative measurement of the central spectroid. In fact it is a kind of variance of the former and helps to describe the perceived timbre.

Some of these features, such as the Mel Spectrogram, MFCCs and their derivatives, can be represented as two-dimensional graphs, while others can be represented as one-dimensional arrays. To maximise the information obtained from the listed features, it was necessary to divide them into submatrices. For the Mel Spectrogram, the matrix was divided into $n \times n$ submatrices. For MFCCs and their derivatives, on the other hand, the matrix was divided into $m \times n$ submatrices, where m is the number of extracted coefficients, which is usually a constant value. Finally, the one-dimensional arrays were simply divided n times. Of all these 'chunks', the mean and std were evaluated and used as a parameter for classification.

B. Model selection

The following algorithms have been tested:

- *Random forest:* classificate by generating a number of decision trees, each of which is unrelated to the other. In fact, each tree considers all the elements, but only takes a subset of features and evaluates the tree by selecting these. After generating the different trees, it ranks the unranked instances through a voting system.
- *SVM:* classification model whose objective is to find the class separation line that maximises the margin between classes, where margin means the minimum distance from the line to the points of the two classes.

C. Hyperparameter tuning

The hyperparameter tuning phase takes place at two different times. First, within the preprocessing and feature extraction phase, in which it was evaluated the number of significant divisions to divide the matrices and arrays obtained from feature extraction to gather some information afterwards After doing so, in the second phase some informations were extracted from created chunks. For completeness, it would be possible to test a different number of divisions not only for rows and columns, but also for each different feature analysed, described in the feature extraction chapter. On the other hand, for the sake of time, it was decided to set the number of divisions of columns equal to that of rows, evaluating, for SVM and RF, the best value of n.

To do this, a test was performed by varying the parameter n on the standard classifiers of RF and SVM, obtaining the f_1 scores described in Fig. 6. As can be seen from the figure, at n = 18 both classifiers reach stability and peak. For this reason, this value was selected to perform the subsequent hyperparamenter tuning.

Before starting the testing and validation phase, the principal component analysis (PCA) method was used to simplify and decrease the number of features that were previously extracted

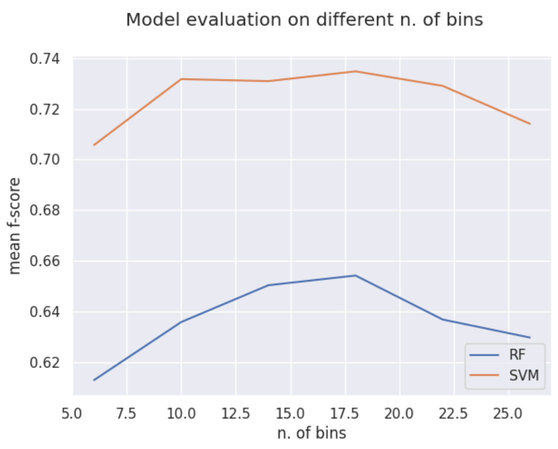

Fig. 6: Plot of the RF and SVM accuracies by varying the value of n.

during feature extraction. This step was crucial from an efficiency point of view, making the code much faster and leaner. The results led to a net lowering of the number of features evaluated within the classifiers, in fact through PCA it is possible to see how only the first 60 features, starting from the initial 2196, are enough to describe with 0.999% accuracy the dataset obtained after the feature extraction, as can also be seen in the Fig. 7. For this reason in the following tests, only these 60 features have been considered and not all of them.

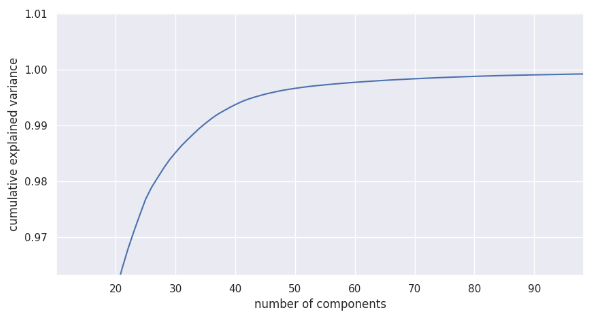

Fig. 7: Cumulative explained variance having n = 18.

After doing so, the best n for RF and SVM was selected, and performed the hyperparameter test based on Table 1. The normal validation and testing process was followed, dividing the development set into train and test in an 80/20 ratio ensuring that all the sets maintained the original proportions of the categories to be predicted. As data augmentation was applied, a hold-out validation was preferred to a cross-validation as this, in conjunction with data augmentation, would have been computationally prohibitive. The weighted f_1 score was used to evaluate the model and hyperparameters so that the unbalanced test set was taken into account. In fact, data augmentation can only be carried out within the training set to avoid the so called data leakage [7]. A close correlation between higher accuracies and higher f_1 scores was noticed during this phase.

Model	**Parameter**	**Values**
Preprocessing	n	$6 \rightarrow 26$, *step* $= 4$
Random Forest	n_estimators	150, 200, 250
	criterion	gini, entropy, log_loss
	max_features	sqrt, log2
	bootstrap	False
SVM	C	10, 100, 1000, 10000
	gamma	scale, 1, 0.1
	kernel	rbf, poly
	class_weight	None, balanced

TABLE I: Hyperparameters considered

III. Results

Selecting $n = 18$ for both classifiers, after validation, the best hyperparameters obtained for RF are as follows: {'bootstrap': False, 'criterion': 'gini', 'max_features': 'sqrt', 'n_estimators': 250}, with a f_1 score ≈ 0.7, while those of the SVM are these: {'C': 10, 'class_weight': 'balanced', 'gamma': 0.1, 'kernel': 'rbf'} with a f_1 score ≈ 0.8. It can be seen that SVM returned much more accurate results than RF.

While performing hyperparameter tuning with and without data augmentation, it was noticed that the accuracies obtained are quite similar, with the difference being that the classifiers obtained with data augmentation are much more robust, generating more or less similar results as the number of hyperparameters varies, whereas those without data augmentation are more variable and dependent on the latter.

Given the great difference between SVM and RF, it was decided to evaluate the two best sets of hyperparameters of the SVM, the first of which has already been described, the second of which is as follows: {'C': 100, 'class_weight': 'balanced', 'gamma': 0.1, 'kernel': 'rbf'}.

In the public leaderboard, the results obtained with these two sets of hyper-parameters are, in order, 0.874 and 0.872.

IV. Discussion

The in-depth analysis and study of the different types of feature extraction that can be performed on audio files has, together with data augmentation, led to more than satisfactory results despite using a limited range of classification methods.

A number of possible aspects to consider in order to achieve an even more effective model are listed below:

1) : The use of neural networks, such as convolutional, recursive or recurrent neural networks, is an approach often recommended in the literature to achieve excellent results in NLP problems [6].

2) : Using a more powerful computer, k-fold can also be used with data augmentation to achieve even more robust results.

3) : In hyperparameter tuning, the division of all obtained matrices and arrays computed during the feature extraction was made using the same parameter n, when potentially each of those could have been divided using a different value.

The results obtained, while positive, still have much room for improvement, as the proposed classification problem is non-trivial. Nevertheless, it cannot be denied that the results obtained are quite satisfying.

REFERENCES

[1] Elena McPhillips, ”Noise levels of everyday sounds”, https://www.audicus.com/noise-levels-of-everyday-sounds/.

[2] Edward Ma, ”Data Augmentation for Audio”, https://medium.com/@makcedward/data-augmentation-for-audio-76912b01fdf6.

[3] Leland Roberts, ”Understanding the Mel Spectrogram”, https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53

[4] Olivia Tanuwidjaja, ”Get To Know Audio Feature Extraction in Python”, https://towardsdatascience.com/get-to-know-audio-feature-extraction-in-python-a499fdaefe42.

[5] librosa.pcen, https://librosa.org/doc/main/generated/librosa.pcen.html.

[6] Olga Davydova, ”7 types of Artificial Neural Networks for Natural Language Processing”, https://medium.com/@datamonsters/artificial-neural-networks-for-natural-language-processing-part-1-64ca9ebfa3b2.

[7] Aashish Nair, ”An Introduction to Data Leakage”, https://towardsdatascience.com/an-introduction-to-data-leakage-f1c58f7c1d64.