Speech recognition - Intent detection

Florentin-Cristian Udrea
Politecnico di Torino
s319029

Sara Rosato
Politecnico di Torino
s317547

***Abstract*—In this report we introduce a possible approach to the Intent detection problem starting from the audio samples without using speech to text techniques. In particular, the proposed approach consists leveraging both temporal and frequency domain information by extracting the log-Mel spectogram and the first thirteen Mel-frequency cepstral coefficients (MFCCs) of each audio signal and split it into a specific number of blocks from which are computed summaries described as statistical features. The information will then be fed to a two model sequential pipeline in order to predict the final intent. The proposed approach outperforms a baseline defined for the problem and obtains overall satisfactory results.**

I. Problem overview

The problem proposed is known as *audio intent detection*. It involves identifying the underlying intention or purpose of a statement, in our case, given as a audio registration. The provided dataset is split in two distinct parts:

- a *development* set, containing 9854 recordings and their associated intent (described as two separate features: **action** and **object**, the final intent would be their string concatenation).
- an *evaluation set*, containing of 1455 recordings, but without the associated intent.

The objective is to use the development set to build a classification process able to correctly label the evaluation set.
In both the sets, other than the audio recording, we also have several other information about the speaker:

- self-reported fluency level
- first language spoken
- current language used for work/school
- gender
- age range

II. Proposed approach

A. Data preprocessing

The first aspect analyzed was checking the missing values values in our dataset. Fortunately our dataset didn't have any, so we proceeded with considering other aspects.

After a fast exploration of the value distribution of these attributes, it becomes clear that only one of their values has statistical significance. For istance, the value distribution of the first language spoken, plotted as a histogram in fig. 1.

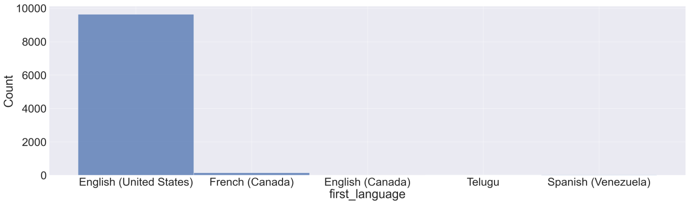

Fig. 1. First language distribution among speakers.

This pattern repeats also for *fluency level* and *current language*. Since the less significant attribute values also coincided with less fluent english speakers, we decided to mark them as outliers, keeping only the speakers with the following attribute values:

- *self-reported fluency level*: native or advanced
- *first language spoken*: English (United States)
- *current language*: English (United States)

Age range also presented such a distribution, speakers of age *"65+"* were mostly underrepresented in our dataset, yet it didn't denote a lesser capacity in speaking the english language, so we preferred to keep all records.

On the other hand the *gender* attribute is equally distributed therefore we kept all records.

At the end of the data cleaning process we removed what we considered being outliers in our dataset and managed to keep its size comparable with the one we started from, going from 9854 records to 9081 records.

Next we focused on audio files and their characteristics. The first step was to consider the frequency rate at which they were recorded: most of them at 16 kHz, but not all, since the rest were recorded at 22.5 kHz frequency rate. In order to have a uniform dataset we brought all audio files to 16 kHz through a **resampling** process.
Since the audio files were recorded in different contexts, many of them had different volume levels (expressed as amplitude standard deviation, as shown in fig.2). A **normalization** step was then considered almost essential.

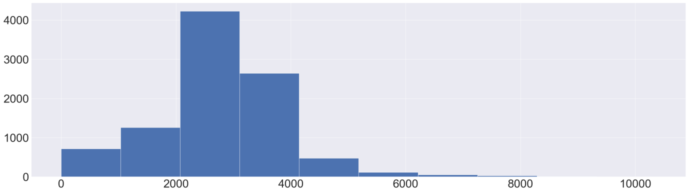

Fig. 2. Standard deviation distribution of audio files before preprocessing.

The last step was the examination of audio file lengths (fig.3). As the figure shows, our dataset presented outliers having a duration of 20 seconds and, by inspecting the longest audio (fig.4), it becomes obvious that the issue is the silence at the beginning and at the end of the recording. By **trimming** the audio files, we removed the problem.

In order to ensure the robustness from outliers of our dataset we further ignored audio files of length bigger than after-trimming 95% quantile, which stated at 1.92 seconds.

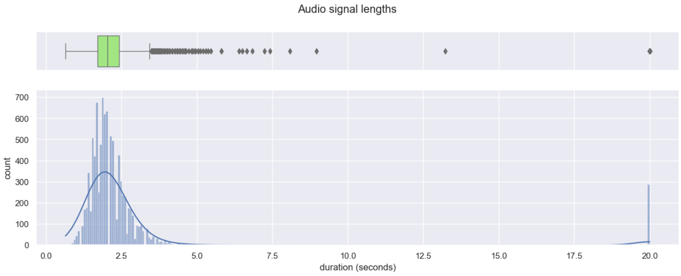

Fig. 3. Audio length distribution before preprocessing (seconds).

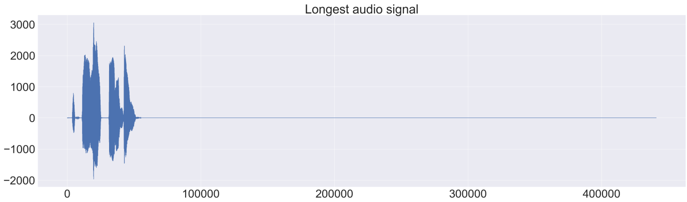

Fig. 4. Longest audio signal before preprocessing represented in time domain.

The preprocessed audio files were still unusable for a classification algorithm. A feature extraction was needed in order to bring out relevant data about the signals that was usable for the chosen algorithm. We opted for a mixed approach, consisting of both **frequency domain and time domain statistical features**. By using the **log-Mel spectogram** we could ensure the representation of the two domain features.

A spectogram is a matrix that contains the frequancy power magnitude at a certain time. It is represented as in fig.5:

- on x axis: it contains the time dimension
- on y axis: it contains the frequency dimension
- the color: contains the power magnitude of the frequency described by y axis at the time described by x axis

Furthermore the power magnitude is converted in logarithmic scale as it is more representative of human perception.

Since the spectograms (as matrixes) had different dimensions based on the duration of the audio file we needed a way to standardize the number of features that we input into the algorithms. In order to do that we **split the spectogram matrix into n^2 blocks** (splitting rows in n groups and columns in n groups as shown in fig.6) and **computed for each block its mean and standard deviation**. This way we wouldn't just obtain an *uniform number of features*, but also *a feature reduction*. We will consider n as a hyperparameter and will discuss its choice in section *II.E*.

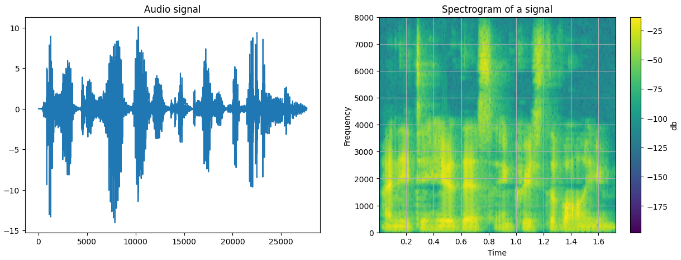

Fig. 5. Audio signal: time domain and log Mel spectogram representation.

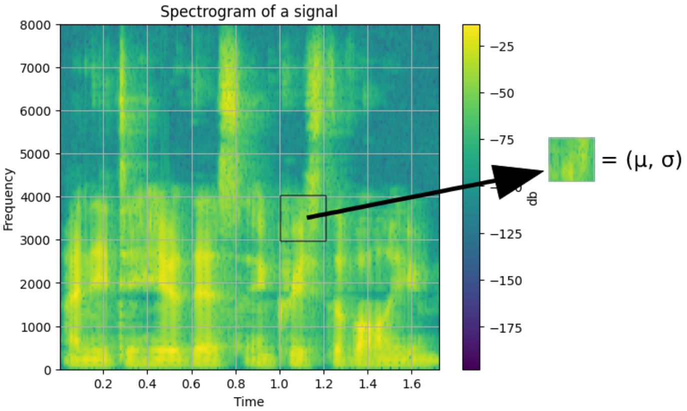

Fig. 6. Representation of spectogram splitting and feature extraction.

We further enriched our dataset by extracting the **Mel-frequency cepstral coefficients (MFCCs)** out of each audio file. It is a feature representation commonly used in speech recognition problems to extract relevant information from audio signals. MFCCs are derived from the power spectrum of a speech signal, and they capture the characteristics of the speech signal that are most relevant to the perception of speech by the human ear. The coefficients were represented as an array of values of variable length based on the length of the audio signal. In order to uniform the number of values we used the same technique used for Mel-spectrogram: splitting the array in n equal parts and extracting the mean and standard deviation out of each split. The number of coefficients used was 13 as they are commonly considered to contain the most relevant information [1].

As n grows, the number of features grow quadraticly. A large number of features can decrease the accuracy of our model as some less meaningful features may act only as 'noise' (this phenomenon is known as 'curse of dimensionality' [2]). For this reason we applied the **principal component analysis (PCA)** in order to reduce the number of features. PCA asks for the number k of features to which we want to reduce our dataset to and keeps the k features that best separate the different classes. This is accomplished by linearly transforming the data into a new coordinate system where (most of) the variation in the data can be described with fewer dimensions than the initial data. We will also consider k as a hyperparameter and it will be discussed in section *II.E*.

B. Model selection

As previously mentioned the class (target of our classification pipeline) is described as two separate attributes: **object** which has 5 different values and **action** which has also 5 different attributes. Since the objective is to predict the combination of action and object we studied how many combinations are there present in the dataset.

TABLE I
OBJECT - ACTION COMBINATIONS

Object	**Action**
volume	increase decrease
heat	increase decrease
change	language
activate	music
deactivate	lights

As shown in table 1 there are only 7 possible combinations. Furthermore only *volume* and *heat* have two actions associated while the rest have only one. This let us thinking that by predicting the correct object, the action was almost certain, so one approach would be using a **pipeline of two classifiers**, the first one would predict the object and the second one would predict the action having a dataset containing also the object predicted. Considering that the prediction would be a categorical value, it would be surely necessary to transform it through an encoder (in this case we can use the one hot encoder because the number of objects is small).

Another approach would be to try and predict directly the combination of action and object, but since the number of single objects is smaller than the number of combinations it would have more data per each class, so we expect to have a better accuracy using the pipeline described in the first method.

The algorithms tested for both steps are the following:

- *Random Forest*: it is an ensemble method, using multiple decision trees (trained on random subsets of the dataset) and giving the prediction by using majority votation. It generally avoids the problem of overfitting generated by simple decision trees. Since random forest works on one value at the time, there is no need to normalize the dataset. We use random forest algorithm as it has been shown to work well on audio data [3].
- *SVM*: the algorithm applies (generally non linear) transformation to the data points then maximum margin hyperplane that separates the classes. Since the margin is calculated as distance between the points and the hyperplane a normalization of the dataset tipically increase result quality. This model has also been shown to work well on audio data [4].

The dataset uses statistical frequency attributes which are not understandable to begin with, so the problem of model understandability is not to be taken in consideration. Furthermore we did not use the speaker's information (age, gender) as it has no correlation with their intent.

C. Hyperparameter tuning

There are three sets of parameteres that require tuning:

- spectogram number of blocks: n
- PCA number of components: k
- SVM and RF parameters

As MFCCs and spectogram contain different information we decided to use the same number of splits n. This way we have splits that refer to the same part of the audio signal and give insights from different points of view.

For simplicity we assume that feature extraction and selection parameters (n and k) are independent from model parameters, so we will firstly analyze the performances of the models having default parameters on all n and k combinations, then will analyze models hyperparameters on the best combination found. The process will be repeated for both models: first on the object classifier, then on the action classifier having a dataset with correct (not predicted) information about objects. The hyperparameter values used are presented in table II.

TABLE II
HYPERPARAMETERS VALUES

Model	**Paramter**	**Values**
Spectrogram and MFCCs	n	$6 \rightarrow 42$, step 3
PCA	k	[10,30,70,100,300,500,700]
SVM	C $kernel$	[1, 5, 10, 50, 100] $[rbf, linear, sigmoid]$
RFC	*max_depth* *criterion*	[None, 2 5, 10, 50] [*gini, entropy*]

In order to ensure that our model is not overfitting we used as a metric the mean of the **k-fold cross validation** accuracy scores, using $k_{cv} = 5$. This way we can assume that our hyperparameters are not optimizing for the test dataset only.

III. RESULTS

Unsurprisingly, both classifiers gave best results for the same feature extraction parameters. In fig.7 are described the accuracies of the object classifier for both algorithms and, even if values change slightly, the trend is the same also for the action classifier.

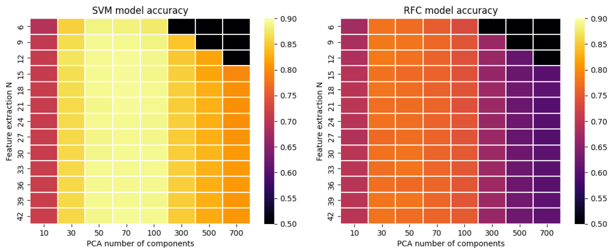

Fig. 7. Accuracy heatmap of default models SVM and RFC addressing object classification under the changing variables n and k

The best combination of feature extraction parameters resulted as $(n, k) = (12, 70)$ used on the SVM algorithm, with the following accuracies:

- Object-classifier: 0.893
- Action-classifier: 0.884

A further inspection of the neighbourhood of the best parameters showed that results were almost the same for $n \in [10, 15]$ and $k \in [50, 80]$. In this case the best result was the configuration $(n, k) = (15, 80)$ used on the SVM classifier that would bring the accuracy up to 0.894, so we kept it.

At this stage the SVM classifier was already outperforming the Random Forest Classifier, which had mean accuracy of around 0.78 in both object and action classification.

After the model hyperparameter tuning the SVM accuracy was slightly better and we assessed that best results were given by:

- Object-classifier: SVM(C=5, kernel='rbf') → accuracy=0.904
- Action-classifier: SVM(C=5, kernel='rbf') → accuracy=0.900

The Random Forest Classifier, instead, didn't get any improvement from the hyperparameter tuning process, so we decided that the two SVM configurations shown above would be implemented in the final pipeline.

If we consider the accuracy as the probability of computing the right result and assume the two probabilities as independent, the final accuracy we could expect was $a_{final} \approx 0.813$, value given by multiplying the accuracies of the two models. In fact the accuracy given by predicting the evaluation set and submitting the result on the leaderboard was $a_{leaderboard} \approx 0.823$ which is comparable. The slight improvement (with respect to the expected value) can be explained as the final models had the entire development dataset to train on, which may have boosted the performance (other than the inexact assumption that the two probabilities are independent).

IV. Discussion

The final pipeline was composed by two SVM models used sequentially, the second one working on the initial dataset enriched by the first model's prediction. It worked on a modified dataset that leveraged both temporal and frequency features by using the log-Mel spectogram and MFCCs.

In order to assure the quality of our pipeline we decided to compare it with a naive classifier, in particular one that labels every record as the most frequent class. Our pipeline was, indeed, outperforming the naive classifier as it had a mean cross-validation accuracy of $a_{naive} = 0.249$.

Even if the final accuracy score is satisfactory when compared with the baseline ($a_{baseline} = 0.334$) and with the naive classifier($a_{naive} = 0.249$) we believe that in the context of a bigger project where more time and resources can be allocated there are more advanced techniques that could be considered in order to create a better performing pipeline, such as:

- using **more complex models**, such as convolutional neural networks (CNN, that perform and optimize feature extraction internally) or recurrent neural networks (RNN, that work really well on sequential data such as audio files), in order to increase the accuracy of the pipeline at each step.
- taking into consideration **other features** that may be extracted from the audio in the dataset, such as: *Spectral Centroid* (represents the center of gravity of the audio spectrum and is commonly used as a measure of the "brightness" of the sound), *Zero Crossing Rate* (the rate of sign-changes along a signal, often used as a feature for speech and audio classification) and so on.
- improving the dataset using **audio data augmentation techniques**, such as: adding noise, changing the pitch, changing the tempo, stretching the audio file and so on. This would be particularly beneficial in our case, since it would even out the difference in distribution of the outcomes

References

[1] T. Davis and K. K. Paliwal, "Unsupervised feature extraction techniques for speech signals," Signal Processing, *IEEE Transactions on*, vol. 42, pp. 3006-3018, 1994.

[2] "Curse of dimensionality and feature selection," by Guyon, I., Weston, J., Barnhill, S., Vapnik, V. (2003). Available at: https://link.springer.com/article/10.1023/A:1023784627191

[3] F. Saki, A. Sehgal, I. Panahi, and N. Kehtarnavaz, "Smartphone-based real-time classification of noise signals using subband features and random forest classifier," in 2016 *IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 2204–2208.

[4] P. Dhanalakshmi, S. Palanivel, and V. Ramalingam, "Classification of audio signals using svm and rbfnn," *Expert Systems with applications*, vol. 36, no. 3, Part 2, pp. 6069 – 6075, 2009.