Intent recognition problem analysis

Giulia Monchietto, Andrea Ruglioni
Politecnico di Torino
Student ids: s315312, s309568
giulia.monchietto@studenti.polito.it, andrea.ruglioni@studenti.polito.it

***Abstract*—In this report, a machine learning model for intent recognition is proposed. The model is able to recognize key words from wav audio files by using Mel-frequency cepstral coefficients (MFCCs) as features, Principal Component Analysis (PCA) to reduce features number, and K-Nearest Neighbors (KNN) algorithm as a classifier. It achieves an accuracy of 0.762 on the public leaderboard, showing potential for future improvements.**

I. Problem overview

Intent recognition is an important task in natural language processing (NLP) and speech recognition. Its goal is to determine the underlying purpose of a part of speech. This is a challenging task because the audio files contain little background noise and people speaking with different accents and variations in speaking style. Additionally, audio data are not easy to work with as they require specialized preprocessing techniques in order to extract useful features.

The dataset given has been divided in two parts:

- Development set, containing 9954 instances, 7 predictive attributes and the target, *intent*.
- Evaluation set, containing 1455 instances with the same structure besides the target variable.

The split has been made in order to train a model using the first one, and evaluate its accuracy using the second one.

Each row of the dataset contains information about the person speaking, from his/her *speakerId*, to his/her *gender*, *ageRange*, *self-reported fluency level*, *first Language spoken* and the *current language used for work/school*, and the *path* to the wav file. The *intent* refers to the purpose of the speaker and it is a categorical response variable assuming 7 possible values, some of which are "activate music", "increase heat", and "deactivate lights". Its distribution can be seen in figure 1, and it can be observed an imbalance between the counts.

From a brief exploratory data analysis, it is immediate to observe that there are no missing or duplicated values. Then it can be noted that 10834 out of the total 11309 speakers are native english (from United States). Thus, they account for nearly 96% of the data, making the remaining speakers negligible and not numerically significant. Therefore, we could cut off the 3 attributes regarding languages or alternatively consider the non native speakers as outliers and exclude them from the development set, focusing our attention to the recognition of the native ones intents. Both options have been tested, bringing no significant difference in the results, so we will proceed with the first one.

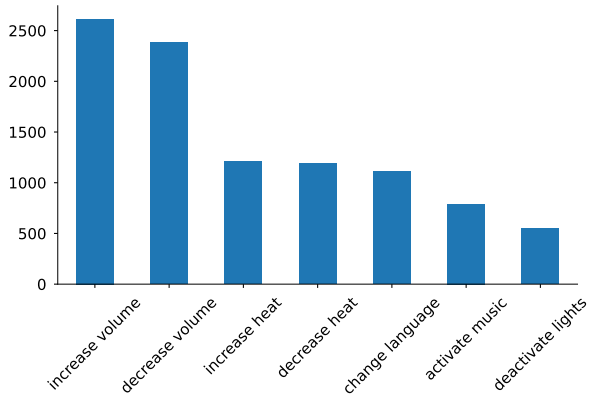

Fig. 1: distribution of the response variable

On the other hand, *gender* and *ageRange* will be kept since speakers in the same bucket could have similar tone colour, making the recognition easier.

For what concerns the wav audio files, they have been sampled at the same rate, 22050 Hz, which is a standard value for speech recording when perceived quality is unimportant, but clarity must be maintained, providing a good trade-off between quality and memory load. Moreover, the audio lengths are different, having a mean duration of about 2.6 seconds. In order to extract an equal number of features from each recording, it will be necessary to properly adjust each one of them.

Listening to some audios, it is interesting to note that the intent does not exactly coincide with the person statement. For instance, the speaker could say "volume up" instead of "increase volume". This increases the complexity of the problem because there is not temporal overlapping between speakers belonging to the same class.

II. Proposed approach

A. Preprocessing

Firstly, the categorical features *gender* and *ageRange* have been one-hot-encoded (adding up respectively 2 and 3 features) while *speakerId* has been discarded because it had 97 unique values. Therefore, its encoding would have ramp the number of features up by 96, increasing the computational cost and time efficiency without actually improving the accuracy.

Moving our focus towards the audio files, they were preprocessed by trimming, cutting, and padding. The first step

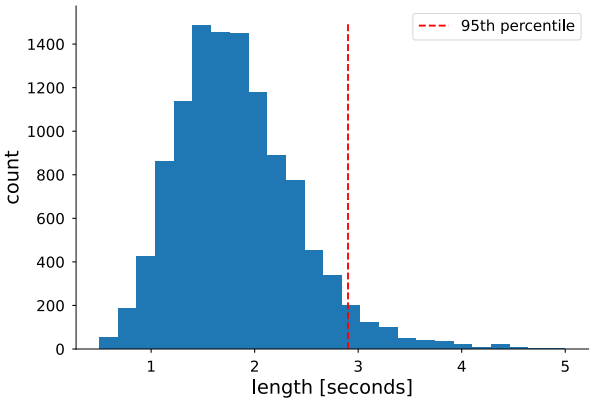

Fig. 2: histogram of recordings' length

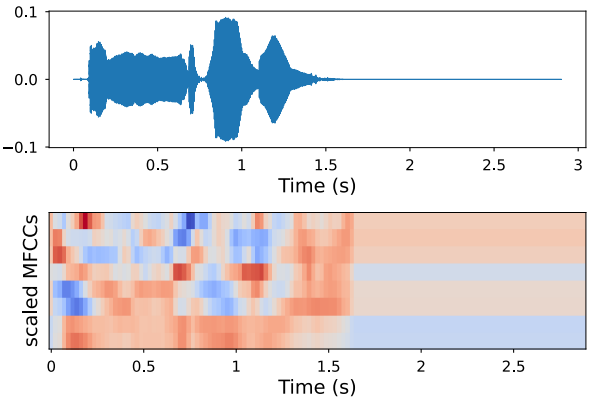

Fig. 3: wave plot and the relative scaled MFCC

involved removing any trailing and leading silences or background noises, making possible to cut the mean duration to about 1.8 seconds without any loss of relevant information. Next, the audio files were cut and padded. This has been done to ensure that the features extracted are comparable and of the same length, obtaining as a result consistent data for scikit-learn. Going into detail, the 95th percentile p of the lengths has been taken, indicating the length such that 95% of data are shorter than p. We got $p = 2.9$ seconds. In figure 2 is shown the distribution of the recordings' length with the 95th percentile.

The cut and pad method refers, respectively, to the discard of the recording part over p and to the addition of trailing zeros (which represent silence) to recording with length less than p.

The next step was to compute the Mel-frequency cepstral coefficients (MFCCs) [1]. MFCCs are the "de facto" standard for speech recognition because they provide a good representation of the spectral properties, still having low complexity. However, they are not very robust at the presence of noise. Therefore, it is common to normalise their values to lessen their influence, and so we did. The term ceptral refers to the cepstrum, which is a tool to investigate periodic structures in frequency spectra. Instead, Mel-frequency indicates that the frequency studied are equally spaced on the mel scale, which is a non-linear scale that is based on the way the human ear perceives different frequencies of sound.

In figure 3, in the upper plot can be seen clearly the effect of the trimming and padding, while the bottom figure shows its scaled MFCC. The function `librosa.feature.mfcc` has been used to get the coefficients. Its default number of coefficients returned is `n-mfcc = 20`. Anyway, it could be considered an hyperparameter, and the best value found is 8.

At the end, Principal Component Analysis (PCA) was used to reduce the number of features. In fact, the MFCC built from each recording is represented by a 125×8 matrix, producing the overwhelming number of 1000 features plus the 5 from *gender* and *ageRange*. PCA is a technique used for feature extraction and dimensionality reduction, which can improve the computational efficiency of the model. It works by transforming the original features into a new set of features, called principal components, which are linear combinations of the original ones. These principal components are chosen such that they capture the most important information in the original features, explaining the most variance possible. In our case, it narrows the features down to 162 still explaining the 90% of the variance in the development set.

B. Model selection

The following two models have been deepened because researches have proved their efficiency for speech recognition systems [2]:

- The K-Nearest Neighbors (KNN) algorithm is a simple, yet effective, classifier that can be useful for recognizing patterns in data. It is a non-parametric method, which means it does not make any assumptions about the underlying data distribution. This makes it well-suited for a problem like intent recognition where the data may be very complex. Additionally, KNN algorithm is computationally efficient and easy to implement.
- The Support Vector Machine (SVM) is a very versatile model whose basic idea is to find the best linear decision surface, that separates different classes. Their importance is due to the kernel trick, which virtually transforms the data in a higher dimensional space, where it is linearly separable, obtaining a non-linear decision boundary in the original feature space. For speech recognition models, SVMs are often used with a radial basis function (RBF) kernel. Moreover, the SVM has been trained with a one vs one approach consisting in fitting one classifier per class pair. At prediction time, the class receiving the most votes is selected.

C. Hyperparameters tuning

In this section is reported the process of tuning the previous models' hyperparameters using a grid search 5-fold cross-validation method. It involves training the model multiple times with different combinations of hyperparameters, and

TABLE I: parameters grid

Model	Parameter	Values
SVM	kernel	['linear', 'poly', 'rbf']
	C	[0.1, 1, 10]
KNN	n neighbors	[5, 6, 7, 8, 9, 10]
	weights	['uniform', 'distance']

TABLE II: Results on development set

Model	Accuracy	Best parameters
SVM	0.639	kernel: 'rbf'
		C: 10
KNN	0.667	n neighbors: 6
		weights: 'distance'

evaluating the performance of each combination using cross-validation. In our case, the performance score of interest is the accuracy. The possible parameters' values can be seen in table I.

In general, k-fold cross-validation is a method of evaluating the performance by dividing the development data into training and validation sets, with size respectively $(k-1)/k$ and $1/k$. This process is repeated k times with different partitions of the data, and the average performance is used to evaluate the model. Therefore, k training and testing processes have to be performed which could be computationally expensive for slower models like SVM.

In addition, there is the hyperparameter `n-mfcc` from the feature extraction section, which will be tested on the best model found.

III. Results

In this section we will show the results obtained using the procedures described in the previous sections. For testing purpose k-fold cross validation on the development set has been used, leading to the average accuracies in Table II.

The results show little difference in accuracy, with KNN algorithm performing slightly better.

Subsequently, we have validated our KNN and SVM algorithms with the best hyperparameters configurations found, changing MFCCs filters number. We have tested 7 different values: 4, 6, 8, 10, 12, 14, and 16. The optimal number found has been 8 in both cases, with accuracies displayed in figure 4.

After model selection and hyperparameters tuning processes, the two best models have been evaluated on the public evaluation test set. The **SVM** model achieved an accuracy of **0.728**, while **KNN** model achieved an accuracy of **0.762**.

IV. Discussion

The results prove the potential of using MFCCs, PCA and KNN algorithm for intent recognition task. The preprocessing steps such as features selection and normalization were crucial

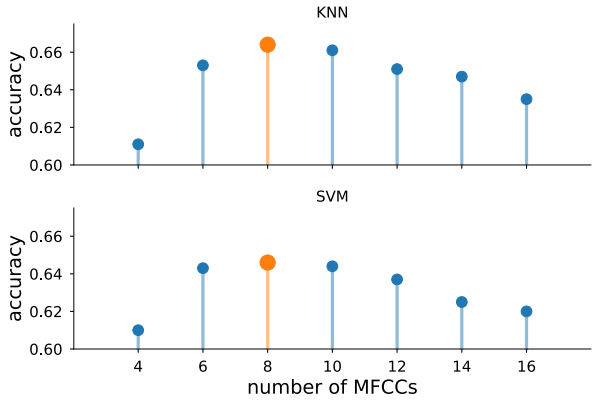

Fig. 4: accuracy varyng n-mfcc

TABLE III: results on public leaderboard

Model	Accuracy	Best parameters
SVM	0.728	kernel: 'rbf'
		C: 10
KNN	0.762	n neighbors: 6
		weights: 'distance'

for achieving good performance. MFCCs in particular show great speech recognition performances in a clear environment, while they have poor robustness to noise signals. In order to enhance robustness, mean and variance normalization has been performed on each MFCC feature. In fact normalization procedures enable to lessen the influence of variability and noise in each recording.

For what concerns MFCCs extraction hyperparameters tuning, results show that higher numbers of coefficients, implying greater features quality, does not always lead to higher accuracy. Therefore, a simpler extraction using 8 coefficients has been preferred, in order to not overfit training data.

After the preprocessing stage, PCA has been fundamental to reduce feature vector's size while retaining important information. Thus allowing shorter algorithms training time, preventing overfitting and avoiding the curse of dimensionality.

In model selection the best KNN results were compared with SVM model outcomes, with the latter showing potential as benchmark for future works. In fact, inference for SVM model takes $O(d)$, where d is the number of features, since determination of which side of a hyperplane a given point lies on is only needed. On the other hand, KNN inference requires computation of distances between new objects and already classified objects. Therefore, for fast classification purposes SVM could help achieve a more efficient performance with slightly worse accuracy results.

In conclusion, a great accuracy gap has been obtained between evaluation set and validation on development set testing. The behaviour could be due to the fact that model training during hyperparameters tuning has been performed on

4/5 of the development set. While before testing on evaluation set, the model has been trained on the entire training set, leading to higher accuracy.

Outcomes also indicate that there is a room for improvement and further research could be done to improve the performance of the model. For example, other feature extraction techniques could be explored, such as DWT to improve noise robustness, or more complex normalization methods could be employed to reduce the influence of low energy components in MFCCs.

REFERENCES

[1] S. Samad, T. Idebeaa, S. Majeed, and H. Husain, "Mel frequency cepstral coefficients. feature extraction enhancement in the application of the speech recognition: A comparison study," *Journal of theoretical and applied information technology*, 2015.

[2] J. Weston and C. Watkins, *Support vector machines for multi-class pattern recognition*. Esann, 1999.

[3] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "Librosa: Audio and music signal analysis in python," in *International Society for Music Information Retrieval Conference*, pp. 561–566, 2015.

[4] M. Labied and A. Belangour, "Automatic speech recognition features extraction techniques: A multi-criteria comparison," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 8, 2021.