CS229 - Final Report
Speech Recording based Language Recognition
(Natural Language)

Leopold Cambier - lcambier; Matan Leibovich - matanle; Cindy Orozco Bohorquez - orozcocc

Abstract

We construct a real time language classifier for communication purposes. Feature vectors are based on Shifted Delta Cepstral Coefficients (SDC). The classifier is constructed by a maximum likelihood estimator based on Gaussian Mixture Model (GMM) density estimations of the feature vectors. As expected, classification error decreases with model complexity to a certain limit. We observe that the optimal number of gaussians varies significantly across languages, and there is clustering pattern in the confusion matrix.

I. Introduction

How to recognize a spoken language without any semantical analysis?. This is a common problem in fields where language detection has to be immediate, and therefore complex algorithms to identify phones and then words are not feasible. For example, modern day communication requires service providers to be accessible to many parts of the globe, where English is not a commonly spoken language. In case there is need to provide information, or converse with a client, there is great urgency in detecting a language in which the client can communicate.

Another application is national security. Intelligence agencies collect copious amounts of cellular data. In many cases, it is imperative to identify the language spoken as soon as possible, for the data to be valuable. Finally, technology applications should fit in a globalized world, and that includes satisfy the demands of polyglot people, who maintain interactions in different languages during a daily routine. The applications should be able to identify and switch from one language to another, depending on the demand of the user. Just think in Stanford students whose first language is not English and want to communicate with both family and classmates.

Notice that as an additional outcome, a language classification algorithm can aid in mapping relations between languages, and distinctive features that would not necessarily be thought of as ones, specially for those languages that do not have a written representation.

With this set of motivations, we construct a real time language classifier which identifies any speech audio sample to a language within a given database. For this purpose, we receive as input an audio sample of spoken language (which can be as small as 210 ms sample), we extract a set of features called Shifted Cepstral coefficients and then we use a maximum likelihood estimator based on Gaussian Mixture Model (GMM) to identify the most suitable language among the trained ones.

II. Problem

A. Description

For the space of audio signals

$$\mathcal{S} : [0, T] \to \mathbb{R},$$

construct a classifier

$$\mathcal{C} : \mathcal{S} \to \mathcal{L}$$

i.e., given an audio sample s, classify it to ℓ, a member of a known set of languages $\mathcal{L}$

$$\mathcal{C}(s) = \ell \in \mathcal{L}.$$

$\mathcal{C}$ is constructed using a training dataset $\mathcal{D}$,

$$\mathcal{D} \subset \mathcal{S} \times \mathcal{L}, \quad \mathcal{D} = \{(s_i, \ell_i) | s_i \in \mathcal{S}, \ell_i \in \mathcal{L}\}.$$

B. Stages

The work itself is constructed of 3 stages

1) Constructing the feature vectors for every sample in the training set using Shifted Cepstral Coefficients.
2) Training a model for each language, and cross validate to select the best fit.
3) Compute test error.

III. Related Work

A comprehensive review of historical methods and approaches to spoken language recognition appears in [1]. Human language recognition has been the topic of research for many decades. Human listening experiments suggest that there are two broad classes of language cues people rely on in identifying a language: the pre-lexical information, such as intonation and pitch and the lexical semantic knowledge based on identifying words, syntax and context. Research shows that pre-lexical recognition predates lexical recognitions, as infants are able to identify a language long before they have a solid grasp of language vocabulary and syntax.

Accordingly, there is a choice of the features to use when trying to classify a spoken language: they are

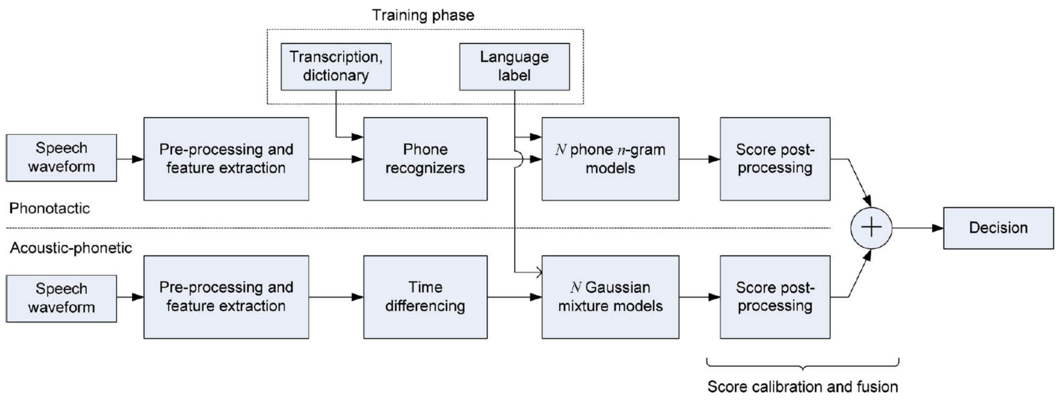
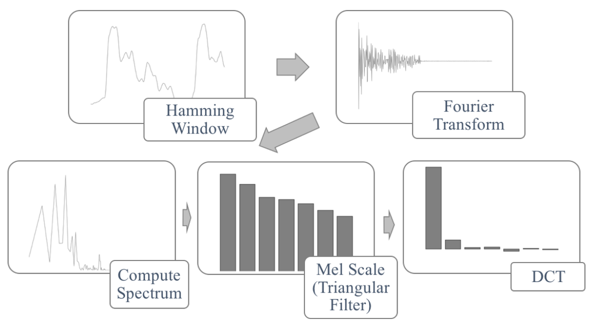
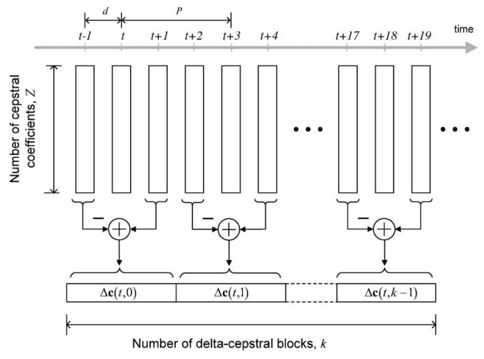


Fig. 1: Classification schemes for acoustic and phonotactic methods [1]



(a) Cepstral coefficients


d
P
time
t-1
t
t+1
t+2
t+3
t+4
t+17
t+18
t+19
Number of cepstral coefficients, Z
Δc(t,0)
Δc(t,1)
Δc(t,k−1)
Number of delta-cepstral blocks, k

(b) SDC coefficients [1]

Fig. 2: Extraction of Feature Vectors diagrams

roughly divided into “acoustic” - physical sound patterns associated with a language, which relate to pre-lexical data and “phonotactic” - constrained syllable structures, more associated with grammar and syntax structure. Acoustic features are easier to describe as they relate to primordial traits of the language, while phonotactic features rely on particular grammatical structures. As we were interested in finding a classifier that would work well on a heterogeneous class of exotic languages, we wished to make fewest assumptions about the grammatical structure of the languages (tonal/ admissible consonants etc.). Therefore we chose to focus on Acoustic features.

There is a plethora of classification methods based on extracted features. However, as demonstrated in Fig 1, the most effective methods seem to be based on N-gram modeling for phonotactic features (recognizing phones in a speech segments and assessing the probability of a sequence of N such phones given each language in dictionary), and mixture of Gaussian density estimation for the conditional density of the features. As we chose acoustic features, we based our classification on a mixture of Gaussians density estimation.

IV. Feature Vectors — Cepstral Coefficients

Cepstral coefficients have been used in language processing for a while, since they seem to match the human auditory perception. i.e. signals with highly different Cepstral coefficient are most likely to be attributed to different sources — for example in speaker classification.

Cepstral coefficients are usually computed over a phone, which is a of a duration of 10ms on average for human speakers. Cepstral coefficients for every 10 ms sample are constructed by (also see figure 2a)

$$\begin{aligned}
\text{Hamming Window}: \; & x_{\text{Ham}}(t) = (x^{*}\text{Ham})(t), \\
\text{Fourier Transform}: \; & \hat{x}(f) = \mathcal{F}(x_{\text{Ham}}(t)), \\
\text{Transition to Mel Scale}: \; & \hat{x}_{\text{Mel}}(f) = \hat{x}(\mathcal{M}(f)), \\
\text{DCT of logarithm}: \; & x_{\text{Cep}}[i] = \text{DCT}(\log \hat{x}_{\text{Mel}}(f))[i].
\end{aligned}$$

We can also construct Shifted Delta Cepstral coefficients (SDC) by looking at the variation of the Cepstral coefficients over adjacent samples (see figure 2b),

$$x^{k}_{\text{SDC}}[i] = x^{k+1}_{\text{Cep}}[i] - x^{k-1}_{\text{Cep}}[i]. \quad (1)$$

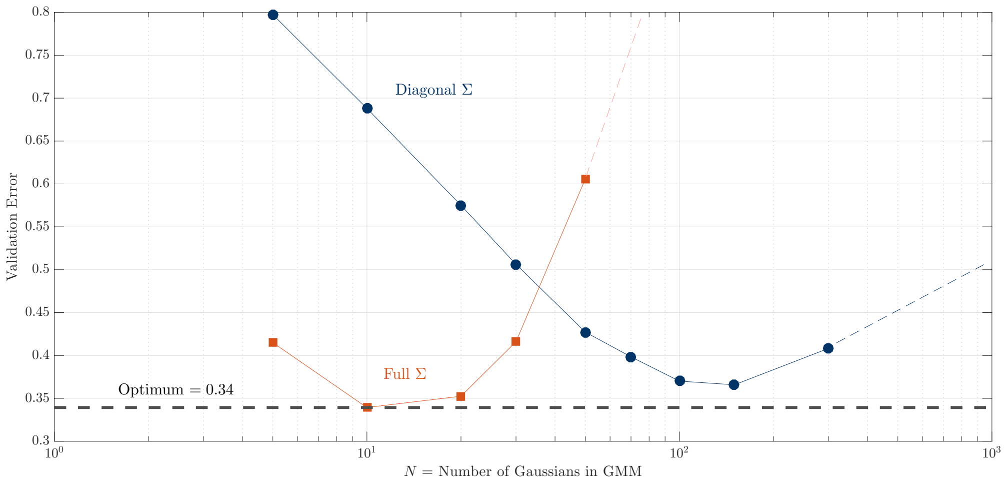


Fig. 3: Validation error as a function of N

V. Dataset

We train our model $\mathcal{C}$ with a data base from a Hacking competition from the webpage topcoder [3]. This data set contains information from 176 different languages, each one with 376 samples of 10 seconds. Each of these samples is a recording of daily conversation. One particular characteristic of this data set is the unusual presence of exotic languages. We divide each 10s sample into 210ms samples. For each 210 ms sample we construct 21 7-vectors Cepstral coefficient for each 10 ms segment. We then use each 3 adjacent Cepstral coefficients to construct a 7-vector SDC coefficient. Thus we have a 49 feature vector for each 210 ms sample. $46 \times 376 = 17296$ data samples, to be used for training, validation and testing (where we divide them at the 10s level). We first isolate 10% of the dataset for testing, and then divide the remaining 90% into 70% for training and 30% for validation.

VI. Training Model

In the training we use a GMM mixture model: every language is modeled as a sum of Gaussians. However, we do not assume every Gaussian is associated with a different label. Rather, this is an extension of the GDA analysis to multiple Gaussians, such that the training algorithm can model generalized distributions. We assume that the prior distribution of the languages is uniform, which is the case in our training set, which mainly contains exotic languages.

We train the model on $0.9 \cdot 0.7 = 63$ % of our dataset to get

$$p(x|\ell) = \sum_{i=1}^{N} w^{(i)} p(x|\mu_\ell^{(i)}, \Sigma_\ell^{(i)}) \tag{2}$$

where x is the 49- feature vector, constructed in the training set from a 210ms sample, $w_\ell^{(i)}, \mu_\ell^{(i)}, \Sigma_\ell^{(i)}$ are our trained weights, means and covariances, and N is the number of Gaussians in the model.

VII. Validation

Given the trained parameters in (2), we calculate the conditional probability of each 10s sample $x^{(i)}$, in our validation set to be drawn from a specific language by

$$p(x^{(i)}|\ell) = \prod_{j=1}^{46} p(x_j^{(i)}|\ell) \tag{3}$$

$$\hat{\ell} = \arg\max_{\ell} p(x^{(i)}|\ell) \tag{4}$$

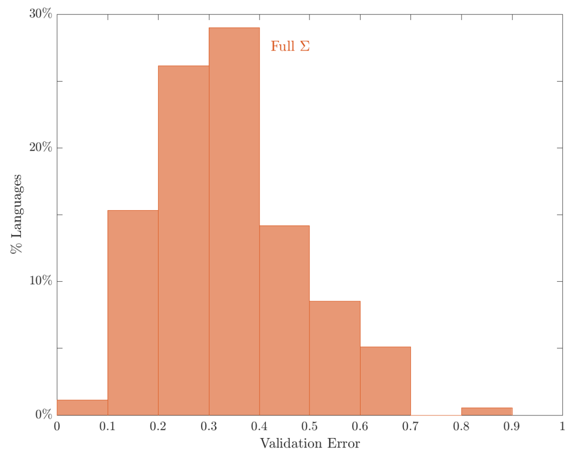
where $x_j^{(i)}$ is the the j-th 210 ms segment of $x^{(i)}$. We store the probabilities in logarithm to avoid machine precision issues.

VIII. Results

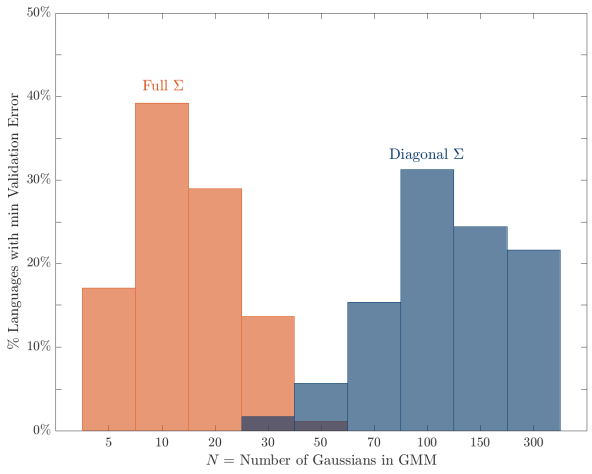
A. *Validation error and hyperparameters optimization*

To explore different models, we trained our model assuming both a full and diagonal covariance matrix. The validation errors for various values of N are depicted on Figure 3 for those two models.

Both models present the same bias-variance behavior. We can see that the "full" model runs into problems after $N = 20$. Given that for $N = 20$ we would have roughly $20 \times 49 + 20 \times 49^2 \approx 50,000$ variables to solve for while we have $\approx 18,000$ data samples per language in total, the problem quickly becomes over determined for the full Gaussian model (for instance, the covariance matrices learned become singular for $N = 100$). Using diagonal covariances alleviates this issue to some extent,



(a) Distribution of validation error for $N = 10$



(b) Distribution of the optimal N across languages

Fig. 4: Analysis of the validation error for each individual language

allowing us to go up to $N = 300$ before reaching singularity issues. As we can see on the graph, the smallest validation error is reached for $N = 10$ with the full covariances matrices. In order to use more Gaussians for training the models, we need to increase the number of data samples.

Since we are training the model for every language independently, we are also interested in how the error spreads between different languages. First, as we can see in Figure 4a, for $N = 10$ using full covariance matrix, we have a unimodal distribution of the error among the languages and the error $[0, 1]$. The same pattern holds as we change N or the type of covariance matrix. Therefore, this suggests that some languages are more susceptible to inaccurate modeling, either because the GMM assumption is not accurate enough at low N, or because the set of features does not capture distinguishable characteristics of them.

On the other hand, we also analyzed the impact of choosing a uniform N for all languages. In the original model, we fit all languages with the same number of Gaussians N, and if this parameter were independent of the language, then we would expect that all languages get their minimum validation error at the same N. Nevertheless in Figure 4b, we can see the distribution of the optimal N across languages. For example for models with full covariance matrix, only 40% of the languages get the minimum error in the optimal selection of $N = 10$. The same phenomenon happens for diagonal covariance matrix, where the optimal $N = 150$, but only 25% of the languages attains its minimum error at this N, whereas more than 30% of the languages have minimum error when $N = 100$. This analysis show that in order to improve our predictions, we need to make N variable among different languages.

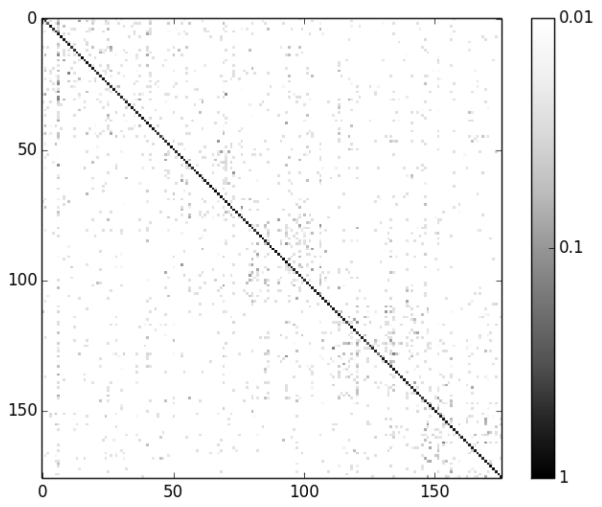
B. Test error and confusion matrix

After selecting the best model in section VIII-A ($N = 10$ with dense confusion matrix), we run the model on the 10% remaining data. We obtained a test error of 0.34, which is consistent with the validation error obtained earlier, suggesting good generalization of the model.

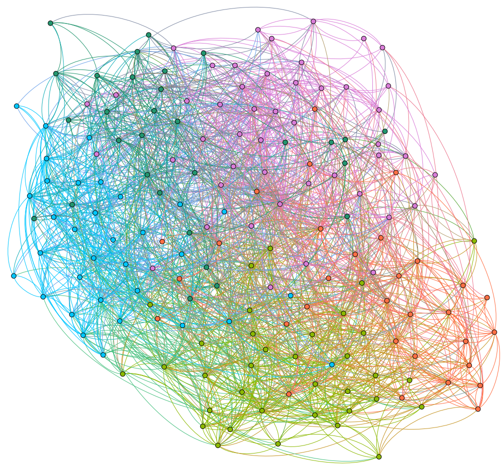
To analyze the error by language, we computed the confusion matrix of the test set. In this matrix, every (i, j) entry corresponds to the fraction of cases when given language i it is predicted as language j. To calculate this value, we work with the score vector defined in (3) per each sample, and then we normalize along all the samples. To have a better visualization, we use colors in log-scale, as shown in Figure 5a.

The confusion matrix is a standard technique to identify if the misclassification error is uniform along the sample, or if there exists subfamilies or clusters, where the error resides. Given the large amount of classes (nearly 200), without any relative order among them, plotting the confusion matrix to find clusters is useless, unless we have by chance a suitable permutation. In order to get this permutation, we understood the confusion matrix as the adjacency matrix of a directed graph. Under this assumption, clusters of misclassified languages are communities in the graph.

Therefore, once we computed the confusion matrix, we used a community detection algorithm [2] to find five communities of languages. Once this is done, a suitable permutation of the confusion matrix gives us Figure 5a and the associated graph is depicted on Figure 5b. This shows that the error, when any, is not uniform and tends



(a) Permuted Confusion matrix



(b) Communities graph

Fig. 5: Cluster analysis in test error for $N = 10$ with full covariance matrix, with highlighted communities

to cluster, i.e., the algorithm tends to confuse similar language together within a community.

IX. Conclusion

In this project, we implemented a real time language classifier for communication purposes. After building the features using SDC and using a GMM model with MLE estimator for classification, we obtained a test error of 0.34, which has to be compared to the performance a random estimator would have on the 176 classes of the dataset. Looking at the data, we see that there is some pattern in the classification, in the sense that when the model misclassifies a language, it does not do so at random but tend to classify the language within some community.

X. Future work

Considering the results from section VIII-A, one way to improve the algorithm would be to allow the number of Gaussian to vary for each language. This should likely allow us to improve the fit and decrease the validation and test error.

Additionally, as figure 3 illustrates, the model quickly become overdetermined because of the important number of unknowns and the limited size of the dataset. One way to improve the model would then be to find a larger dataset to be able to fit more complex models which may improve the accuracy of the algorithm.

Another interesting feature would be to compute a measure of confidence when predicting a language (confidence interval). This could be coupled with a measure of the quality of the GMM fit for each language. For example, for a given sample, we could return the probability of that sample being well classified, and a set of (e.g. 5) likely languages, with a score and confidence interval for each of them.

The described approach can be based in the clustering effect of the classification described in section VIII-B. A hierarchical clustering algorithm, where different communities of languages are treated separately, could lead to some improvements. In this model, the goal would be to first classify an audio sample among different communities of languages, which ideally (but not mandatory), correspond to the geographical region or the same linguistic family. After this step, we would find the specific language to which it belongs, looking only among the distinct community. This may also help improve performances.

Lastly, other direction for future work include making the model available to targeted users. This includes developing an interactive interface that easily recognizes languages, and later on, additional components can be added such as real time recognition of language, even it is switched along time.

References

[1] Li, Haizhou, Bin Ma, and Kong Aik Lee. ”Spoken language recognition: from fundamentals to practice.” Proceedings of the IEEE 101.5 (2013): 1136-1159.

[2] Blondel, Vincent D., et al. ”Fast unfolding of communities in large networks.” Journal of statistical mechanics: theory and experiment 2008.10 (2008): P10008.

[3] topcoder. ”Maraton Match: Spoken Languages 2” (2015). Last visit October 2016: https://community.topcoder.com/longcontest/?module=ViewProblemStatement&rd=16555&pm=13978