Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Pronunciation Assessment at Phone Level for Second Language Learning

Tesis presentada para optar al título de

Licenciado en Ciencias de la Computación

Leandro Ariel Matayoshi

Directora: Dra. Luciana Ferrer

Buenos Aires, Octubre 2018

ABSTRACT

Technological advances of last decades have allowed the development and expansion of Computer-Assisted language learning (CALL) systems. These systems assist second language learners in different tasks regarding grammar, vocabulary and pronunciation. In the current work, we focus on *Pronunciation Assessment*, a particular subfield of pronunciation. Pronunciation Assessment consists in determining whether a recorded speech was correctly or incorrectly pronounced. The analysis is performed at a predefined level, such as sentence, word or phone level.

Currently, whenever performing pronunciation assessment, the most reliable estimates are obtained from paragraphs or long sentences. On the other hand, the smaller the unit (and therefore the smaller the amount of information in the speech segment), the less precise is the estimate of the assessment. However, pronunciation assessment systems that operate at shorter levels, such as phone level, not only can point out specific errors produced by the students but also can be used by children that still have difficulties in pronouncing long sentences. For these reasons, in the current work we will focus on phone-level pronunciation assessment methods.

The more standard methods in the literature for pronunciation assessment at phone level usually involve using generative approaches based on Gaussian Mixture Models. Usually, for each phone two individual GMMs are trained: one using the correctly pronounced instances of that phone and the other one using the incorrect instances. A standard way to make the assessment is to compute the Likelihood-Ratio between the two models. In a previous work in the pronunciation assessment field at phone level, a discriminative approach based on *Support Vector Machines* (SVM) trained on special features called *supervectors* was explored, leading to slightly better results than generative models such as *Gaussian Mixture Models* (GMMs). Supervectors are derived from adapted GMMs that are trained using all the available instances for a given phone.

In the current work, we use as reference and baseline system the SVM model trained on supervectors in order to explore new features in the phone-level pronunciation assessment field. Even though both GMMs and supervectors summarize the low level acoustic information of the speech segment, they don't provide information about the temporal dependencies of the features. Because of that reason, in the current work we study features that model explicitly the

dynamics of the acoustic features over time. In order to do so, each feature is modeled independently by a parametric function, from which the dynamic features are extracted. Two different parameterization techiniques are studied: *Legendre Polynomials* and *Discrete Cosine Transform* (DCT). The objective is to analyse if the proposed dynamic features carry complementary information to supervectors features.

We train and test the baseline and the proposed methods on a Latin-American Spanish speech database. The dataset consists in 2550 utterances adding up to a total of 130,000 phone instances, labeled by expert phoneticians. Recordings are uttered by 206 native American English speakers. Results showed that for a subset of the phones, the combination of supervectors and dynamic features reduce the error compared with using supervectors only, thus supporting the hypothesis that both set of features carry complementary information.

Keywords: Computer-Assisted Language Learning, Pronunciation Assessment, Phone, Support Vector Machines, Gaussian Mixture Models, Supervectors, Legendre Polynomials, Discrete Cosine Transform

RESUMEN

Los avances tecnológicos de las últimas décadas han posibilitado el desarrollo de sistemas automáticos de Asistencia Computarizada para el Aprendizaje de Idiomas (ACAI). Estos sistemas brindan ayuda a estudiantes de segundos idiomas en diversos campos, entre las cuales se destacan la gramática, el vocabulario y la pronunciación. En el presente trabajo nos concentramos en una forma particular de asistencia relacionada con el último campo: la evaluación de la pronunciación, que consiste en decidir si los segmentos de habla presentes en una determinada grabación fueron pronunciados de forma correcta o incorrecta. Dicho análisis puede realizarse a distintos niveles tales como oración, palabra o fono.

Actualmente, las estimaciones más confiables de la evaluación de la pronunciación son obtenidas a nivel de párrafo u oraciones largas, disminuyendo la precisión de los sistemas a medida que se reduce la duración (y por lo tanto la cantidad de información) del segmento de habla a considerar. Sin embargo, los sistemas de evaluación de la pronunciación que trabajan con unidades de habla más cortas, como por ejemplo el fono, permiten poner el foco en errores específicos del estudiante y pueden ser utilizados por niños aún incapaces de pronunciar frases demasiado largas. Por esta razón, en este trabajo nos concentramos en métodos de evaluación de la pronunciación a nivel fono.

Los métodos tradicionalmente utilizados para evaluar la pronunciación a nivel fono están basados en métodos generativos a partir de modelos de mezclas Gaussianas (GMMs). Generalmente, para cada fono se entrena un GMM por clase (pronunciación correcta e incorrecta), aplicando luego técnicas tales como el Cociente de Verosimilitud (*Likelihood-Ratio* en inglés) entre ambos modelos realizar la evaluación. En un trabajo anterior en el área de evaluación de la pronunciación a nivel fono, se exploró un método discriminativo basado en Máquinas de Vectores de Soporte (SVM) entrenado con atributos llamados *supervectores*, que produce resultados ligeramente mejores a los métodos generativos comúnmente utilizados en el campo. Los *supervectores* para cada fono se obtienen a partir de un proceso de adaptación de un GMM global entrenado con la totalidad de las muestras de dicho fono.

En el presente trabajo, tomamos como base y punto de referencia el modelo SVM entrenado con supervectores para explorar nuevos atributos en el área de evaluación de la pronunciación

a nivel fono. Si bien tanto GMMs como supervectores modelan las características acústicas de bajo nivel del segmento de habla a considerar, no tienen en cuenta el comportamiento temporal de las mismas. Por este motivo, en esta ocasión estudiamos atributos dinámicos que modelan de manera directa el comportamiento temporal de dichas características acústicas. Para ello, cada una es aproximada de manera independiente por una función, a partir de la cual se extraen los atributos dinámicos. Dos técnicas de aproximación son evaluadas como posibles alternativas: Polinomios de Legendre y Transformada Discreta del Coseno (DCT). El objetivo es analizar si los atributos dinámicos propuestos tienen información complementaria a la provista por los supervectores.

Entrenamos y evaluamos los métodos base y los propuestos usando una base de datos no nativa de Español Latino, correspondiente a 206 hablantes estadounidenses, estudiantes de Español. La base de datos está conformada por 2550 grabaciones alcanzando un total de 130.000 instancias de fonos etiquetadas por transcriptores profesionales. Los resultados muestran que para un subconjunto de fonos, la combinación de supervectores con los atributos dinámicos efectivamente reduce los errores durante la clasificación, soportando la hipótesis de que ambos tipos de atributos contienen información complementaria.

Palabras claves: Asistencia Computarizada para Aprendizaje de Idiomas, Evaluación de la Pronunciación, Fono, Máquinas de Vectores de Soporte, Modelo de Mezclas Gaussianas, Supervectores, Polinomios de Legendre, Transformada Discreta del Coseno

AGRADECIMIENTOS

A mi directora, Luciana Ferrer, por haberme acompañado en este camino con suma dedicación, atención y responsabilidad, transformando esta última etapa de mi carrera en una experiencia muy linda y enriquecedora.

A Federico Landini, licenciado del Departamento de Computación, por haberme ayudado y compartido la implementación de su código durante la primera mitad de mi tesis.

A mi novia, Claudia, por haber estado a mi lado durante este último año y medio, apoyándome y respetando mis tiempos.

A Manas, por haberme dado la posibilidad de tomarme una licencia para poder dedicarme a la tesis, y por todas las libertades que constantemente me otorgó para que pudiera armonizar el trabajo con la facultad.

A mis compañeres y amigues de la facu: Martín, Dami, Lau, Alex, Sofi, Floppy, Toffo, Pato, Nico, Yanet, Javi por todas las cosas que compartimos en esta aventura de cursar en el DC.

Finalmente, el mayor de los agradecimientos a mi familia por haberme acompañado todos estos años. A mi mamá por enseñarme con el ejemplo, y muy especialmente a mi papá por transmitirme desde siempre la conducta de seguir adelante y superar las adversidades.

CONTENTS

1. INTRODUCTION

1.1 Problem Description and Motivation

Technological improvements of last decades allowed the development and expansion of Computer Assisted Language Learning (CALL) systems. These tools are aimed to help students along the process of second language acquisition. One of the points where the students should focus is pronunciation learning, that usually requires a one-to-one teacher interaction. Automatic pronunciation assessment is very useful in this context because it provides an alternative way of learning with a performance close to human judgement, cheaper and typically available at any time and any place.

When working in automatic pronunciation assessment an important decision to be taken is whether or not considering L1, the native language of the learner, along the process. These systems have improved speech recognition accuracy because they are designed taking into account common known errors between L1 and L2. For example, native american English students that are learning Spanish have difficulties pronouncing the diphtong [eu] (e.g. "eufemismo") or /r/ after [l] (e.g. "alrededor"), [n] (e.g. "sonrisa") and [s] (e.g. "disruptivo"), which should be thrilled in all cases. On the other hand, native Spanish students that are learning English may have trouble pronouncing different kind of vowels that are present in English but not in the former. This approach, however, requires a labeled nonnative speech database which is an expensive and time-consuming task, and it is not feasible in some cases. A database like that is available for the current thesis. It contains collected speech from native american English speakers that are learning Spanish, and it is used to train and test the different models.

In the area of pronunciation assessment, the smaller the unit to be evaluated, the higher the uncertainty in the associated assessment [1]. Currently, the most reliable estimates are obtained from paragraphs composed of several sentences that can be used to characterize the speaker's overall pronunciation proficiency [2]. However, specific problems can be pointed out when assessing smaller units, which is of great value because it allows the students to focus on specific aspects of their speech production.

The current work is part of a bigger project lead by Dra. Ferrer, with the objective of developing a pronunciation scoring system for argentinian children that are learning English.

The project has 3 main stages: The collection and annotation of a speech database of argentinian children, the development of an *Automatic Speech Recognition* (ASR) system and the development of a pronunciation scoring system that works at phone or word level. The fact that the system works with short speech segments is specially important in the context of a pronunciation scoring system for children, because of the difficulty of children in pronouncing longer segments.

In the current thesis, we based on L1 strategies to explore alternatives to existing techniques in the pronunciation assessment field at phone level. The explored methods could be helpful during the development of the CALL system for argentinian children that are learning English.

1.2 Previous work

In this section we will be reviewing representative works in pronunciation assessment at phone level, which is the topic of interest in this thesis.

One of the simplest ways of assessing phone utterances found in the literature is to use segment durations ([3, 1]), which are obtained through the following procedure. First, a duration distribution is generated for each phone using the native training data. Then, phone durations in frames for nonnative utterances are obtained and its value is normalized to compensate for rate of speech. Finally, phone-segment-duration scores are computed as the log-probability of the normalized duration, using the duration distributions previously mentioned. Phone durations are obtained from Viterbi alignments.

In the same works other two methods, which use *Hidden Markov Models* (HMMs) to obtain spectral matches, are tested. Phonetic time alignments for nonnative utterances are generated from an HMM-based speech recognition system trained with native instances. In order to do that, the text pronounced by the student must be known. This is achieved by eliciting speech in a constrained way, such as reading a predefined text. From these phonetic segmentations two probabilistic measures based on HMMs are computed as scores: log-likelihood and log-posterior probabilities. The underlying asumption is that the logarithm of the likelihood or the posterior of the speech data, computed by the Viterbi algorithm using the HMMs trained with speech from native speakers is a good mesasure of the similarity between the students's speech and native-sounding speech.

For each phone segment the log-likelihood score $\hat{l}$ is computed as:

$$\hat{l} = \frac{1}{d} \sum_{t=t_0}^{t_0+d-1} log\ p(y_t|q_i) \tag{1.1}$$

where $p(y_t|q_i)$ is the likelihood at time t with observation vector y_t given the phone class q_i, d is the duration in frames of the phone segment and t_0 is the starting frame index of the phone segment. The likelihood is computed through a forced recognition phase by using a known ortographic transcription of the speech signal. Duration normalization is done to eliminate the dependency of the pronunciation score on the duration of the phone.

Alternatively, log-posterior scores can be computed for each time t:

$$P(q_i|y_t) = \frac{p(y_t|q_i)P(q_i)}{\sum_{j=1}^{M} p(y_t|q_j)P(q_j)} \tag{1.2}$$

Likelihoods in both numerator and denominator are computed in the same way as in Equation 1.1. The sum over j runs over a set of context-independent models for all phone classes. $P(q_i)$ represents the prior probability of the phone class q_i.

Finally, the posterior score $\hat{\rho}$ for the phone segment is defined as:

$$\hat{\rho} = \frac{1}{d} \sum_{t=t_0}^{t_0+d-1} log\ P(q_i|y_t) \tag{1.3}$$

To test each method, a database of nonnative read speech is transcribed and scored for pronunciation quality by expert human raters. Log-posterior probabilities achieves the highest correlation with human ratings, outperforming log-likelihood and normalized duration scores.

A very similar approach to log-posterior probabilities named *Goodness of Pronunciation* (GOP) was developed at the same time as log-posteriors and is investigated in some works [4, 5, 6]. The quality of pronunciation for any phone p is defined to be the duration-normalized log of the posterior probability that the speaker uttered phone p given the corresponding vector of observations $y = \{y_{t_0}, \ldots y_{t_0+d-1}\}$:

$$GOP(p) = \left| log\ \left(\frac{p(y|q)P(p))}{\sum_{q \in Q} p(y|q)P(q))} \right) \right| \Big/ d \tag{1.4}$$

The likelihoods $p(y|q)$ are obtained from the recognizer as a sum of the likelihoods over all observations in the phone. The difference between GOP scores and the log-posterior score

technique previously mentioned is that in GOP, the likelihood for both numerator and every phone in the denominator is computed at segment level as a sum of log-likelihoods per frame over the duration of the phone p, while in Equation 1.2 log-posterior probabilities are computed at the frame level and averaged over the segment's length.

A different technique is explored in [7], which studies the relationship between phonetic mispronunciations that occur in a word and the actual word rating. The research is based on a set of phonological rules representing phonetic mispronunciations between L1 (native language of the students) and L2 (the target language). In this case, the corpus is obtained from chinese learners of English and all the speech data of the corpus is phonetically transcribed by trained linguists. In addition, crowdsourcing is used to collect perceptual gradations of word-level mispronunciations and the WorkerRank algorithm is used to conduct quality control to filter crowdsourced data in terms of reliability. The gradation score for a phonological rule is obtained as the average of the aggregated values of the words (rated via crowdsourcing) containing that rule. On the other hand, gradation for a word is calculated as either the maximum value among all phonological rules that are present in that word or a linear combination of them. Even though the work is focus on the use of phonological rules in word-level mispronunciation gradation, the same approach can be used to evaluate mispronunciation gradation at phone level. In order to derive the phonological rules from the mispronunciations, the previous strategy requires a phonetically labeled nonnative database. Taking that into account, another paper [8] works well as complement to [7], because it develops a speech recognition system for automatic mispronunciation detection capable of transcribing the learner's input speech.

So far, all the aforementioned methods but the last one are based on confidence measures obtained from Automatic Speech Recognition (ASR) systems. Scores measure how closely the utterance of the speaker matches the recognizer's acoustic model. Mismatches result in low confidence scores, which provide a profile of the speakers' production errors. Nevertheless, the accuracy of the assessment based on these confidence scores can be quite low [9]. In addition, ASR systems based on HMMs are both time-consuming to train and extremely vulnerable to acoustic interference and variation in speaking style, and the conventional methods for enhancing ASR performance often require enormous amounts of data collection and annotation, as well as extensive training on representative material. For those reasons, other types of classifiers are explored in many works, specially after the 1990s.

Support Vector Machines (SVMs) are a preferred choice due to their excellent generalization

capability and suitability for 2-class classification problems. Moreover, in contrast to the confidence models described above, which do not take into account misspronounced data, SVMs are trained directly to discriminate the two classes of interest. Of course, this poses a new requirement: both correctly and incorrectly-pronounced data should be available for training.

In [10], SVMs are used to discriminate between good and mispronounced vowels in Dutch. Three different types of features for training the models are evaluated: log-posterior-based scores obtained from HMMs, MFCCs and a set of phonetic features (first three formants, pitch and intensity). Despite the existence of nonnative-speech databases for Dutch language, they were considered too small for the purpose of the research. For that reason, phonemic substitutions that induce vocalic errors are simulated by artificially introducing them in the native corpus. It is worth mentioning that training and testing the models with artificially generated data is not recommended because it can lead to models that do not generalize well to real nonnative data. Having said that, the results show that the best results are obtained by using MFCCs as features of the SVM models, followed by confidence-measure-based scores and finally phonetic features, though substantial improvements can be obtained by combining them.

In a different work [11], SVM is used as the classifier and the log-likelihood ratios between all the acoustic models and the model corresponding to the given phone are employed as features for the classifier. In other words, given the observation y, the feature vector for the classification problem is computed as:

$$[LLR(y|q,q_1),\ LLR(y|q,q_2)\dots LLR(y|q,q_Q)], \tag{1.5}$$

where q is the canonical label of the observation being pronounced, $q_1, q_2 \dots q_Q$ represent the set of acoustic models of all the phones and LLR is defined as:

$LLR(y|q,q_i) = log\ p(y|q_i) - log\ p(y|q)$, where likelihood is computed at segment level as in Equation 1.4 for GOP scores. The reason for choosing these features is that q can be mispronounced as any other phone and the likelihood ratio is a powerful method to detect this kind of mispronunciation. Classifiers trained with these features, however, can only effectively deal with a phone mispronounced as another phone. This type of acoustic models is less effective for partially changed mispronunciations, that are the most frequent mispronunciations in practice. In order to detect all kinds of mispronunciation, a technique called *Pronunciation Space Models* (PSMs) is introduced. The idea is to represent pronunciation variations of different proficiency

levels. The traditional phone-based model q is expanded to $\{q_1, q_2 \dots q_K\}$ by means of an unsupervised algorithm that does not require speech data labeled with proficiency level. This set of models represents K types of pronunciation variations ranging from perfect to totally wrong pronunciations, where K is a tunable parameter. To build the PSM models for a given phone q, all the instances are sorted according to their posterior probabilities obtained from the original acoustic model for q, and then uniformly split into K group of the same size. Finally, an individual new acoustic model is trained with the observation from each group via maximum likelihood estimation. The feature vector dimension of Equation 1.5 is therefore increased from Q to $Q * K$. Experiments are carried out on the Mandarin mispronunciation detection task for native Chinese speakers with various dialect accents, and data labeled as "correct" or "mispronounced" by experienced human annotators is used to train and test the models. Results show that SVM based on log-likelihood ratio features outperforms GOP scores (used as baseline), and some improvements are obtained when adding PSMs.

A last example of SVM-based research is found in [9, 12]. A landmark-based SVM is introduced and compared with a confidence scoring method over 10 phonemes where L2 English learners, whose native language is Korean, make frequent errors. Landmark theory relies on the fact that humans can perceive distinctive features using only spectral features extracted from the time frame including and adjacent to a landmark (sudden signal change). A particular SVM for each phoneme is trained. Vowel SVMs are trained using one or more frames from the vowel center. Frames from both onset and offset (prevocalic and postvocalic position) are selected and used in the training of consonant SVMs. The confidence score method shows a similar performance to SVM, though by combining the two scores, statistically significant improvements are achieved for almost all phonemes.

A different strategy for discrimination of Dutch velar fricative /x/ versus the velar plosive /k/ is studied in [13]. Linear Discriminant Analysis (LDA) over two different sets of features are explored and compared with previous approaches: aforementioned GOP scores and Weigelt algorithm. The latter is based on three simple measures that can be easily obtained: log root-mean-square (rms) energy, the derivative of log rms energy (*Rate of Rise* or ROR) and zero-crossing rate. Weigelt algorithm discriminate plosives from fricatives by using ROR values, based on the fact that the release of the burst of the plosives causes an abrupt rise in amplitude therefore yielding higher values compared with fricatives. On the other hand, LDA weights are assigned to each feature to find the linear combination of features which best separates the

classes. Selecting the most relevant features turns out to give an advantage compared to other classifiers. The two LDA methods yielded the best performing scores followed by GOP and Weigelt.

Finally, some pronunciation assessment approaches based on Deep Neural Networks (DNNs), the most popular machine learning technique at this moment, have been developed for the last few years. In [14], the use of context-dependent DNN Hidden Markov models (or CD-DNN-HMMs) is studied in order to improve the speech recognition performance for a better assessment of children English language learners. The experimental results showed that the DNN-based recognition system approach achieved a significant error reduction compared to the GMM-HMM approach, leading to an improvement in the quality of the extracted features and final spoken English proficiency scores. In [15, 16], a CD-DNN-HMM acoustic model approach for English language is also explored as an alternative to conventional GMM-HMM models. The acoustic models are then used to generate GOP scores at phone level and tested in a language learning corpus with manual grading of pronunciation quality. The GOP scores, generated from the DNN-based acoustic models, correlate better with human scores than those generated from the conventional GMM approaches. The main objective of the current thesis is to explore new features for pronunciation assessment at phone level. For this reason, we choose to focus on well-established modeling approaches rather than DNN-based approaches which are still under development.

Relying on labeled nonnative speech data usually leads to more flexible models with better performance than those trained with native speech when working on pronunciation assessment for specific combinations of L1 and L2. On the one hand, the set of common pronunciation errors tend to be particular for a given <L1, L2> pair. On the other hand, models trained with nonnative speech are better at capturing the subtle differences between the nonnative speech realizations that are considered acceptable versus the nonnative speech realizations that are considered mispronounced, in a similar way that an annotator would do. However, labeled nonnative speech databases are rarely available. Annotation of nonnative speech is an expensive and time-consuming task, and in some cases it is not feasible. Models trained with annotated nonnative speech usually have better performance than those trained with native data, though the cost of nonnative database collection and annotation is very high.

To conclude this section, the papers that set the starting point for the current thesis will be reviewed. They are the two latest works in one of the most important lines of investigation

in the field of pronunciation assessment at phone level. Both use a Latin-American Spanish speech database that includes recordings from native and nonnative speakers, where nonnative speech is produced by speakers whose native language is American English.

The first of these works [17] is a continuation of [1], where log-likelihood and log-posterior scores were proposed as effective methods for assessing pronunciation. In [17], the same log-posterior method (Equation 1.2 and 1.3) is applied using models trained on native data and is used as baseline and compared with a proposed technique based on log-likelihood ratio (LLR). To compute LLRs, the phonetically labeled nonnative database is used to train two different *Gaussian Mixture Models* (GMMs) for each phone class: one model is trained with the "correct" native-like pronunciations of a phone, while the other model is trained with the "mispronounced" or nonnative pronunciations of the same phone. In the evaluation phase, for each phone segment q_i, a length-normalized log-likelihood ratio $LLR(q_i)$ score is computed for the phone segment by using the "mispronounced" and "correct" pronunciation models λ_M and λ_C respectively:

$$LLR(q_i) = \frac{1}{d} \sum_{t=t_0}^{t_0+d-1} [log\ p(y_t|q_i, \lambda_M) - log\ p(y_t|q_i, \lambda_C)] \tag{1.6}$$

The normalization by d allows the definition of unique thresholds for each phone class, independent of the length of the segments. A mispronunciation is detected when the LLR is above the threshold. The LLR method performed better than the posterior-based method for almost all phone classes, specially for those with the highest consistency across transcribers. These results are in line with the fact that models trained on nonnative speech data produce better results than those trained on native speech, as it was previously mentioned.

Finally, the most recent work of the series [2] extends the research proposing two new methods that also explicitly model both mispronunciations and correct pronunciations by nonnative speakers. The LLR method developed in [17] is used as baseline and compared with the new modeling techniques.

The first proposed method also uses LLR based on GMMs trained on nonnative data (Equation 1.6). The difference resides in the way these GMMs are obtained. In the baseline approach, for each phone class two different GMMs are trained: one model is trained with the "correct" native-like pronunciations of a phone while the other is trained with the "mispronounced" or nonnative pronunciations of the same phone. In the new approach, the models for each class for a given phone are obtained by adaptation. The model to which they are adapted is trained using

all the samples from a given phone, ignoring the class ("correct" or "mispronounced"). Bayesian adaptation [18] is used to adapt this class-independent GMM to all the "correctly" pronounced training examples for a phone and obtain the adapted "correct" model for that phone. An analogous process is followed to obtain the adapted "mispronounced" model. For these class-dependent GMMs both the means and the mixture weights are adapted to the class-specific data.

The second proposed method follows a discriminative approach. It uses the class-independent GMM of the previous experiment to create supervectors by adapting this GMM to each phone instance. The supervector for a certain phone instance is obtained by adapting the means and the mixture weights of the original GMM to the acoustic feature vector representing the phone. These supervectors are then used as features to train a linear SVM classifier. To evaluate test instances of a given phone, its supervector is calculated by adaptation of the class-independent GMM to the phone feature vector frames. Finally, the distance of that supervector to the SVM hyperplane is taken as the score for the phone.

The two methods presented in [2] show better results than the mispronunciation detection system based on LLR of independently trained GMMs developed in [17], which at the same time outperforms the standard method that uses phone log-posteriors as scores [1]. This is one of the most important lines of research in pronunciation assessment at phone level, positioning the methods described in [2] among the best systems in the field. For that reason, both generative and discriminative models are used as baseline and as a starting point in the work here presented. A more detailed explanation of these systems will be provided in the next section.

2. METHOD

In this section we will review the way the pronunciation assessment system works and explain the concepts and techniques on which it is based. The diagram below shows the overall architecture of the system:

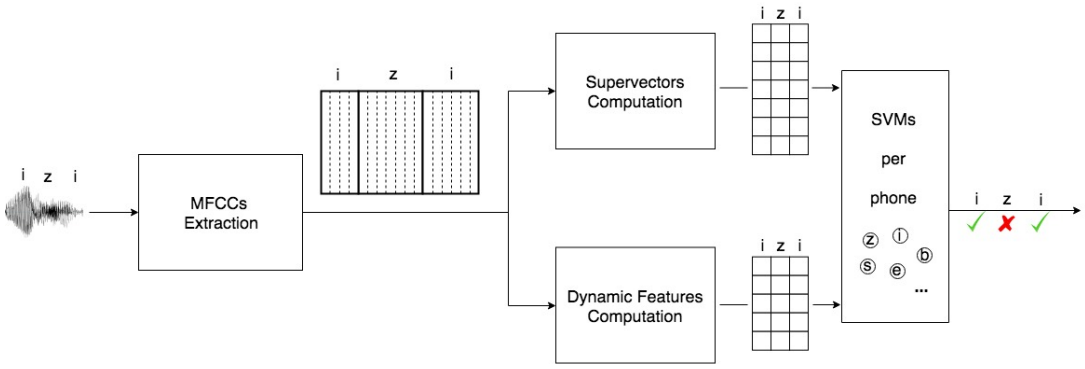


Fig. 2.1: General System Architecture

Given a recording, the objective is to determine whether each of the phones present in the utterance was pronounced correctly or incorrectly. In order to do so, the flow goes as follows:

1. The first step is to compute the audio features from the recordings. This is done in order to summarize the signal information in a measurable way through numeric values. The chosen features are the *Mel Frecuency Cepstral Coefficients* (MFCCs) because they are one of the most standard and widely-spread features in the speech processing field.

2. The MFCCs are then split in segments according to the phones to which they belong. Each segment is at the same time divided in frames, that represent short intervals where the audio signal is not supposed to be changing so much.

3. MFCCs of each phone instance present in the utterance are then used to compute the more complex phone-level features on which the SVMs operates. Supervectors are the base and proven to work features, while the Dynamic Features are the experimental features to be studied.

4. SVM classifiers are trained individually for each phone. Two alternative ways of combining the features are studied:

- Training a single SVM from a combination of both Supervectors and Dynamic Features.
- Training two separate SVMs: one using the Supervector features while the other using the Dynamic Features, and then combining the results.

5. Finally an individual score is obtained for each of the instances of the different phones present in the utterance. Phone instances with positive score are labelled as correct while phone instances with negative score are labelled as incorrect. The bigger the magnitude of the score, the more sure the system is about the decision.

2.1 Features

In this section we describe the chosen features for the current study. The features can be classified in two different categories: frame-level features and phone-level features.

Frame-level features are computed directly from the speech signal, and they carry spectral density information. For the current work, the standard Mel Frequency Cepstral Coefficients were chosen.

The frame-level features are then used as base to compute more complex features: The phone-level features. These are the features used to train the SVM and classify the instances. In the current work, two types of phone-level features are studied: *Supervectors* which are obtained from an adaptation process of a Gaussian Mixture Model trained on the MFCCs and the *Dynamic Features*, which are the coefficients obtained from different parameterization methods of the MFCCs values through time. The main objective of this work is to find out whether or not there is a gain in performance when combining the Supervectors with the Dynamic Features.

In the following subsections a detailed description of each of the aforementioned features is provided.

2.1.1 Frame-level Features: MFCCs

The first step in order to build the system is to extract acoustic features from the speech data. The feature extraction process aims at approximating the linguistic content that is conveyed by the speech signal, by identifying relevant aspects such as information about the frequencies involved during the speech, and discarding unuseful properties.

In this work we base our features in the standard and widely used Mel Frecuency Cepstral

Coefficients (MFCCs). They were introduced by Davis and Mermelstein in 1980 and have been state-of-the-art ever since [19]. MFCC coefficients represent the spectral envelope of the speech signal on the Mel-frequency scale, which relates the perceived frequency of a pure tone to its actual measured frequency inspired by the human hearing.

The process to compute the MFCCs coefficients can be summarized by the following steps:

1. Preemphasis: Preemphasis aims at increasing the amplitude of high frequency bands while decreasing the amplitudes of lower bands by means of a high-pass filter. This stage is performed in order to balance the frequency spectrum since high frequencies usually have smaller magnitudes compared to lower ones.

2. Windowing: The MFCCs are computed in time intervals where the audio signal is not supposed to be changing so much. Because of that reason the signal is divided in frames using a 25 ms duration window and 10 ms shift (standard values). A 25 ms duration window provides enough samples to get a reliable spectral estimate: 200 samples per frame at 8 kHz, and at the same time is short enough to minimize the signal changes. On the other hand, a 10 ms shift leads to a 15 ms overlap between consecutive frames, ensuring that the information between adjacent frames is also captured in the middle of another frame.

3. Discrete Fourier Transform: This is the first and main step in regard to spectral analysis. Frequency domain approaches are proven to be among the best options in speech classification tasks. The *spectral density*, which is the distribution of power of the frequencies composing the signal, is obtained for each frame by using the Discrete Fourier Transform technique.

4. Mel Filter-bank: At this stage, a perceptual scale of pitches inspired in human hearing called Mel scale is used. Because humans are much better at discerning small changes in pitch at low frequencies than they are at high frequencies, the Mel frequency scale is linear up to 1000Hz and logarithmic thereafter according to the following formula:

$$M(f) = 1125 * ln(1 + \frac{f}{700}) \tag{2.1}$$

The spectrum obtained in the third step (3) is passed through a series of traingular filters uniformly spaced on the Mel scale to produce the so called filter-bank energies.

The original values of these energies are replaced by their natural logarithm values, also motivated by human hearing, because loudness is not perceived according to a linear but an exponential scale.

5. Cepstral Coefficients: The Discrete Cosine Transform (DCT) is applied to the logarithm of the filterbank energies to obtain the Cepstral Coefficients. In general, only the lower 12 Cepstral Coefficients are kept. The resulting features are called Mel Frequency Cepstral Coefficients. Additionally, a 13^{th} MFCC computed as the sum of the energy in the frame is included, because it usually improves the performance in phone detection related tasks.

6. Deltas and Double Deltas: Finally, the speech also carries information in the dynamics, i.e, trajectories of the MFCC coefficients over time. Calculating the MFCC trajectories and appending them to the original MFCC vector usually increase the performance of the systems that are based on MFCCs. Both Delta and Delta-Delta features are included, adding to a total of 39 features per frame.

Even though the frame-level features are made up of different features: 13 MFCCs plus 13 deltas and 13 double-deltas, for the sake of simplicity the expression "MFCCs" will be referring to the whole 39 features throughout the current work.

2.1.2 Phone-level Features: Supervectors

Supervectors, which have already been succesfully used in [20, 2] are one of the two features used in this work to train the SVM classifier. Supervectors for a given phone are obtained by extracting the means and weights of different Gaussian Mixture Models derived from the adaptation of a class-independent GMM to each of the instances.

So in order to go into more details of the supervectors features, a brief introduction to GMMs is provided along with an explanation of the adaptation process.

Gaussian Mixture Model

Gaussian Mixture Model is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMMs are often used in biometric systems, specially in speaker recognition systems, due to their ability to represent a large class of sample distributions. One of the powerful attributes of the GMM is to form smooth approximations to arbitrarily shaped densities [21].

The densitiy function of a GMM is defined as:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{2.2}$$

Each Gaussian density $\mathcal{N}(x|\mu_k, \Sigma_k)$ is called a component of the mixture and has its own mean μ_k and variance Σ_k. The parameters π_k are called mixing coefficients and they can be thought of as the prior probability of picking the k^{th} component [22].

In this work, GMMs are used to model the frame-level features mentioned in the previous section: MFCCs (13), plus deltas (13) and double deltas (13). So the features for our GMMs are 39-dimensional vectors corresponding to each frame of each instance of a given phone.

Universal Background Model Adaptation

As in the previous works of the current line of investigation [17, 2], the phone- and class-dependent GMMs are derived by the adaptation of a *Universal Background Model* (UBM) [18], a technique that has been effective on significantly improving the accuracy in the speaker recongition field.

The UBM is a large class-independent GMM intended to represent the class-independent distribution of the features. In a GMM-UBM system, the GMM is derived by adapting the parameters of the UBM to a vector of instances with a *Maximum a Posteriori* (MAP) estimation. This provides a tighter coupling between the generated models that produces better performance than using decoupled models such as class-dependent GMMs. In addition, it can be specially useful when just a small number of training instances is available. In the current work, an individual class-independent UBM (trained with both correctly and mispronounced instances) is generated for each phone.

The UBM approach involves adapting weight, mean and variance of each mixture, though in the current work only weights and means are adapted because of the limited size of the dataset. For each gaussian k composing the GMM-UBM, the weight and mean parameters are updated according to the following equations:

$$\pi_k^{new} = [\alpha_k \pi_k^{'} + (1 - \alpha_k)\pi_k]\gamma \tag{2.3}$$

$$\mu_k^{new} = \alpha_k E_k(x) + (1 - \alpha_k)\mu_k \tag{2.4}$$

$$\alpha_k = \frac{n_k}{n_k + r} \tag{2.5}$$

$$n_k = \sum_{t=1}^{T} Pr(k|x_t) \tag{2.6}$$

where π_k and μ_k are the original values of the weight and mean of the k$^{\text{th}}$ component of the GMM-UBM, $\pi_k^{'}$ and $E_k(x)$ are the estimated values for the weight and mean of the k$^{\text{th}}$ component calculated on the vector of instances provided for the adaptation and the scale factor γ is computed over all the adapted mixture weights to ensure they sum to unity. α_k is the adaptation coefficient that controls the balance between old and new estimates that depends on two things: n_k, which is the sum of the posterior probabilities of the k$^{\text{th}}$ component given the vectors, and the relevance factor r, which is an input parameter. When a mixture component has a low probabilistic count n_k of new data, then $\alpha_k \to 0$ causing the deemphasis of the new (potentially undertrained) parameters and the emphasis of the old (better trained) parameters. Analogously, when mixture components has high probabilistic counts n_k, $\alpha_k \to 1$ causing the use of the new parameters. In addition, a higher value of r produces a decrease in the value of α_k, thus adapting in a more conservative way by assigning more importance to the previous values. By contrary, a lower value of r produces an increment in the value of α_k, thus leading to a more aggresive adaptation strategy by assigning more importance to the new values [18]. In the current work r is set to 0, taking the most aggresive adaptation strategy. This is again motivated by the previous work of the current line of investigation [2].

Supervectors computation

In order to compute the supervectors for a given phone, which are then used to train the SVM classifier, the following steps are performed:

1. At first, a GMM-UBM is trained on the MFCCs of all the available training instances for that phone, regardless of their class label (mispronounced or correctly pronounced).

2. Once the GMM-UBM is trained, for each of the instances of that phone two additional steps are performed:

   (a) An individual MAP adaptation process of the GMM-UBM to the feature vectors within the phone instance is carried out, thus generating an adapted GMM for that

instance.

(b) The supervector for that instance is obtained by stacking the means and the weights of each of the gaussians composing the adapted GMM.

The supervector computation process for a given phone instance is summarized in Fig. 2.2 shown below:

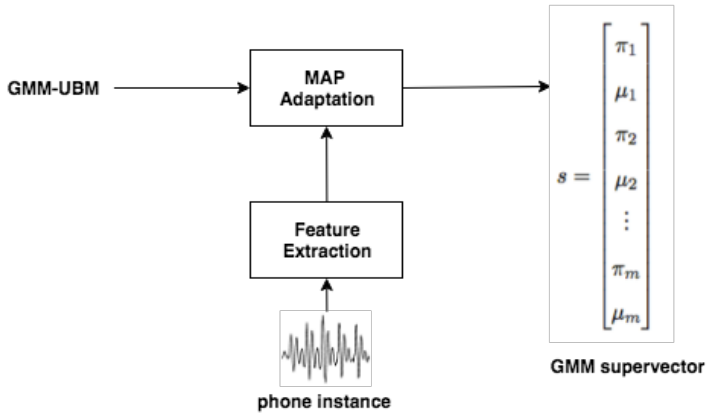


Fig. 2.2: Adapted figure from Campbell, Sturim, Reynolds, Solomonoff 2006 [20]. A MAP Adaptation of a GMM-UBM with m gaussians is performed using the frame-level features of a phone instance. The resulting supervector is made up of the stacked weights π_i and means μ_i $\forall$ $1 \leq i \leq m$, where each weight is a real value and each mean is a 39-dimensional vector.

As each derived GMM has the same number of gaussian components as the initial GMM-UBM, then the dimension of each supervector is: $39*m+m$, where m is the number of gaussians of the GMM-UBM. The first term corresponds to the means while the second one to the weights.

2.1.3 Phone-level features: Dynamic features

Supervectors do not include information about the dynamics of the MFCCs over time. Even though the frame-level features do include deltas and double-deltas, these features only relate the information of adjacent frames.

The evolution of the MFCCs over time can be modeled by using different parameterization techniques, and Dynamic Features are obtained by extracting the coefficients of those parametric functions. Given a phone instance, the values of each of the MFCCs along the whole interval are modeled separately, so a total of 39 functions are computed. The chosen methods for the current work are Legendre Polynomials and Discrete Cosine Transform.

The important thing about Dynamic Features is that they may carry information complementary to supervectors' information, that can be used to improve the performance of the

SVM classifier. A potential gain in the combination of Supervectors with Dynamic Features is therefore an interesting subject to be studied and it is the main topic of the current thesis.

2.1.4 Legendre

Legendre polynomials are frequently encountered in physics and engineering. For example, Legendre is widely used in the determination of wave functions of electrons in the orbits of an atom, or in the determination of potential functions in the spherically symmetric geometry [23]. With regard to speech processing, this technique has been used successfully in the speaker recognition field [24].

Legendre functions are the solutions to *Legendre's differential equation*:

$$(1-x^2)y''(x) - 2xy'(x) + n(n+1)y(x) = 0, \ \ for -1 \leq x \leq 1 \tag{2.7}$$

The solution for each particular $n = 0, 1, 2 \ldots \in \mathbb{N}$ is a polynomial of degree n: $P_n(x)$. These solutions are well known for each $n \in \mathbb{N}$, and they can even be generated recursively. The first five Legendre Polynomials (Fig. 2.3) are:

$$P_0(x) = 1$$

$$P_1(x) = x$$

$$P_2(x) = \tfrac{1}{2}(3x^2 - 1)$$

$$P_3(x) = \tfrac{1}{2}(5x^3 - 3x)$$

$$P_4(x) = \tfrac{1}{8}(35x^4 - 30x^2 + 3)$$

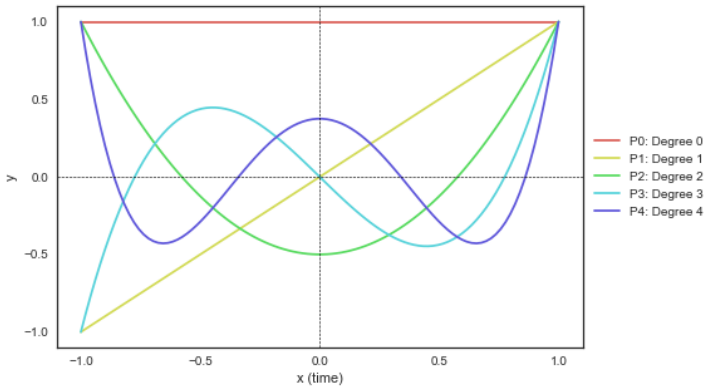


Fig. 2.3: First five Legendre Polynomials in the interval [-1, 1]

Legendre Polynomials are orthogonal with respect to the *L2 norm* on the interval $-1 \leq x \leq 1$:

$$\int_{-1}^{1} P_n(x) * P_m(x)dx = 0,\ m \neq n \tag{2.8}$$

Moreover, in the interval $-1 \leq x \leq 1$ any function f can be represented as sum of Legendre Polynomials, leading to the Fourier-Legendre Series:

$$f(x) = \sum_{0}^{\infty} a_n P_n(x) \tag{2.9}$$

An infinite terms of Legendre Polynomials over the interval $-1 \leq x \leq 1$ can be used to approximate any function. Each coefficient models a particular aspect of the curve: a_0 is the *mean* of the segment, a_1 is the slope, a_2 gives information about the curvature of the segment and the subsequent coefficients model the finer details.

Given the time instants $X = [x_1, x_2 \dots x_t]$ and their respective values of a particular MFCC coefficient at those instants: $Y = [y_1, y_2 \dots y_t]$, the problem of finding the coefficients of *Legendre Polynomial* of degree k (for a previously defined k) that best approximates the values can be formulated in terms of a *Least Squares* minimization problem:

$$\min_{C} \|AC - Y\|^2 \tag{2.10}$$

where the solution of the system C is a $(k+1)$ vector that represents the coefficients of the *Legendre Polynomial* of degree k that best approximates the curve. A is a t x $(k+1)$ matrix, where each column i represents the result of computing *Legendre's* P_i polynomial for each of the values $[x_1, x_2, \dots x_t]$, also known as the *Legendre-Vandermonde* matrix:

$$A = \begin{pmatrix} P_0(x_1) & P_1(x_1) & \cdots & P_k(x_1) \\ P_0(x_2) & P_1(x_2) & \cdots & P_k(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ P_0(x_t) & P_1(x_t) & \cdots & P_k(x_t) \end{pmatrix} \tag{2.11}$$

When modeling a particular event is preferable to pick the simplest hypothesis that best explains the observations, because simpler hypothesis usually generalize better to new observations than more complex hypotheses. As it was said before, the higher the degree of the *Legendre Polynomial*, the more subtle details it models. In order to achieve the right balance between

decreasing the least squares error and keeping the hypothesis simple two different approaches are taken. On the one hand the degree of the polynomial k is limited up to a few coefficients. On the other hand, a regularization technique often used when solving Least Squares problems is explored.

Lasso Regression

A regularization technique is evaluated along with the Legendre Polynomials. Its name is *Lasso Regression* and it modifies the original minimization problem by adding an additional term that penalizes the L1 norm of the solution:

$$\min_{C} \|AC - Y\|^2 + \lambda \|C\|_1 \tag{2.12}$$

Regularization terms leads to solutions where more responsibility is assigned to the coefficients that contributes the most in lowering the error term while shrinking the coefficients with less contribution to lowering the error.

There exists other alternatives to *Lasso* with regard to regularization. A widely-used one is *Tikhonov Regularization*, also known as *Ridge Regression*, where the square of the *L2 norm* ($\|C\|_2^2$) is used as regularization term.

Unlike the square of the *L2 norm*, the *L1 norm* ($\|C\|_1$) leads to an interesting and useful property: In *Ridge Regression*, as the penalty is increased, all parameters are reduced while still remaining non-zero, while in *Lasso* increasing the penalty will cause more and more of the parameters to be driven to zero. Thus *Lasso* automatically selects more relevant features and discards the others whereas *Ridge Regression* never fully discards any features.

2.1.5 Discrete Cosine Transform

An alternative to Legendre Polynomials to capture the general aspects of the utterances in order to summarize the information carried out by the *MFCCs* across time is the *Discrete Cosine Transform*. Variations on the values of MFCCs through time may be better explained using periodic functions instead of polynomial ones for some phones, making *DCT* a more suitable technique for those phones. *DCT* is used in many processes related with science and engineering such as lossy compression of audio (e.g. *MP3*) and images (e.g. *JPEG*), which are among the most popular formats in their respective fields. This technique has also been used

to approximate prosodic features in speaker verification tasks [25].

DCT belongs to the family of the Fourier analysis. As a member of the family, it provides a way to approximate a general function by sums of simpler trigonometric functions. These transformations map a function to a set of coefficients of basis functions, where the basis functions are sinusoidal and are therefore strongly localized in the frecuency spectrum. In particular, *DCT* expresses a finite sequence of data points in terms of a sum of *cosine* functions oscillating at different frequencies. Unlike the *Fourier Transform* that expresses the sequence in terms of both sine and cosine functions, thus needing a complex number to represent the coefficient of each frecuency, it only uses real numbers to output the coefficient of each frecuency.

There exists different variants of the *DCT*, being the type-II the most common and the one that it is used in the current work:

$$X_k = \sum_{n=0}^{N-1} x_n cos[\frac{\pi}{N}(n+\frac{1}{2})k], \ for\ k = [0, 1, \ldots, N-1] \tag{2.13}$$

where N is the number of the extracted samples of the signal to be decomposed. In this way, n coefficients are obtained corresponding to each of the frequencies from 0 to $N-1$. An illustration of the curves with the first two frequencies is shown in Fig. 2.4. Each coefficient is obtained from the sum of the individual contributions of the samples to the k^{th} frequency.

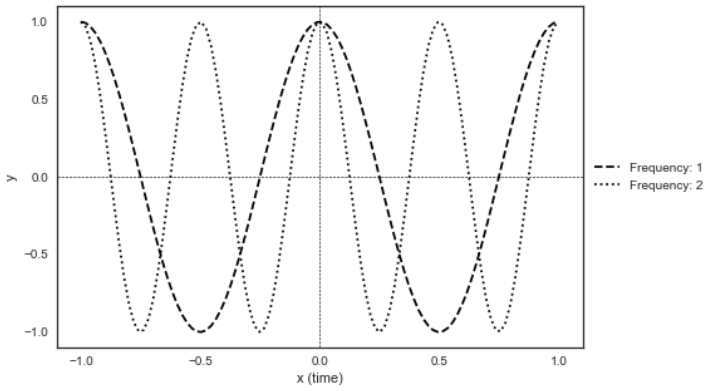
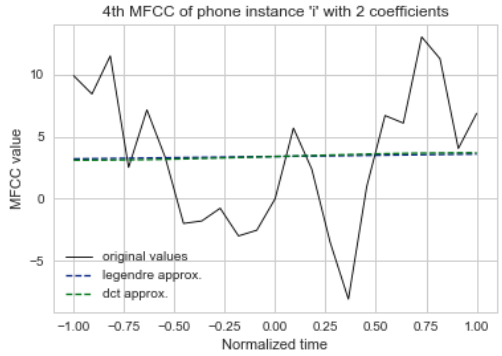


Fig. 2.4: Basis cosine functions of frequencies 1 and 2 over the interval [-1, 1]

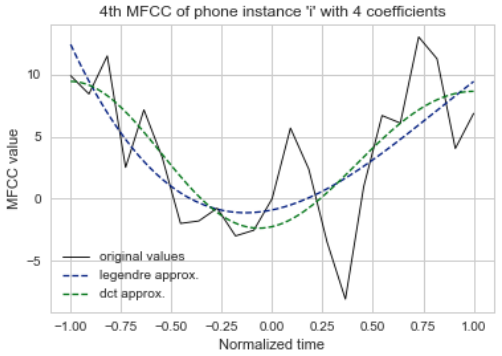
As in Legendre Polynomials, the number of coefficients to be obtained is again limited up to a few coefficients to avoid overfitting to the training set.

2.1.6 Comparison between Legendre and DCT

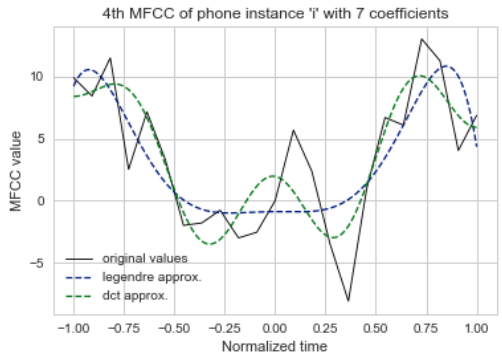
To conclude this section some illustrative visual examples that compare the approximations made by both techniques for a given set of values and different number of coefficients are included below (Fig. 2.5a). The values are extracted from the 4^{th} MFCC of an instance of the phone 'i', taken from the development set.



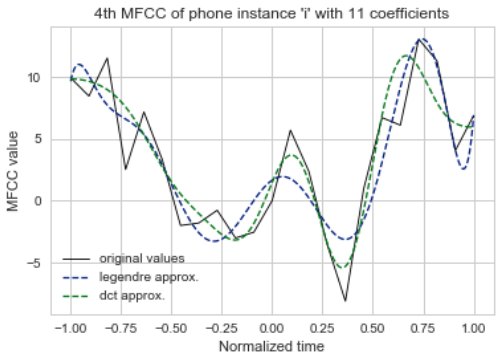
(a) Approximations of Legendre Polynomials with 2 coefficients (degree 1), and DCT with 2 coefficients.



(b) Approximations of Legendre Polynomials with 4 coefficients (degree 3), and DCT with 4 coefficients.



(a) Approximations of Legendre Polynomials with 7 coefficients (degree 6), and DCT with 7 coefficients.



(b) Approximations of Legendre Polynomials with 11 coefficients (degree 10), and DCT with 11 coefficients.

Both techiniques perform pretty well in approximating the curve, and they are also pretty similar when using the same number of coefficients. It can be observed in the example with 7 coefficients in Fig. 2.6a, though, that in some cases DCT is better at modeling the oscilations of the curve while Legendre Polynomials prioritize to capture the overall shape of the curve.

Comparative experiments are carried out in this work in order to find which is the dynamic feature that best complements the supervectors when training the SVM classifier.

2.2 Support Vector Machines

The chosen model for the discriminative analysis is the *Support Vector Machine* classifier, an approach for classification that was developed in the computer science community in the 1990s and has grown in popularity since then. SVM has been successfully used in many speech processing fields, such as speaker recognition and pronunciation assessment.

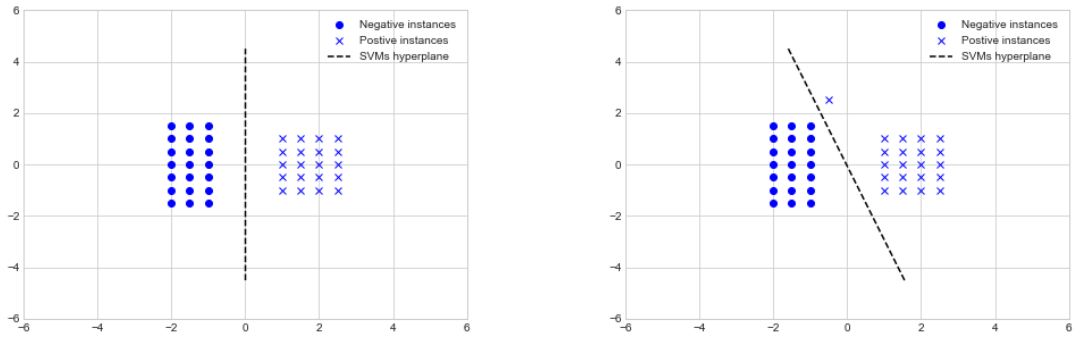
Given a p-dimensional feature space, an SVM (with linear kernel) constructs a hyperplane of dimension p - 1 that separates the space in two halves, which allows to classify the instances according to their relative position to the hyperplane [26]. The classification procedure is then just simply a case of determining which side a test observation falls on.

If the instances are separable by a hyperplane, then there exists potentially infinite separating hyperplanes that can be generated from those same instances, because it is possible to slightly translate or rotate the initial plane without touching any training observation. Among all of them, SVM generates the hyperplane that has the largest distance to the nearest training data points of either of both classes. Such a classifier is known as Maximal Margin Classifier, and in most of the cases generalizes better to unobserved data than any of the other possible separating hyperplanes.

One of the key features of the Max Margin Classifier is that the location of the hyperplane only depends on the support vectors, which are the training observations that lie directly on the margins' boundaries.

2.2.1 Soft Margin

One of the problems with Max Margin Classifiers is that they can be very sensitive to the addition of new training observations: The hyperplane may be adjusted substantially to force every instance to lie in the right side of the margin or hyerplane, thus leading to overfitting while reducing the value of the margin (Figure 2.7). Moreover, many datasets are not perfectly separable via a linear hyperplane at all. Because of these reasons, the requirement that a separating hyperplane will perfectly separate every training observation on the correct side of the line according to its class label can be relaxed by introducing a new concept developed in [27]: the Soft Margin.



(a) An initial training set is optimally divided by the Max Margin Hyperplane of an SVM

(b) The additon of a single instance substantially modifies the position of the hyperplane.

Fig. 2.7: Margin adjustment when inserting a new training instance

A Support Vector Classifier or Soft Margin Classifier allows some observations to be on the incorrect side of the margin or even on the incorrect side of the hyperplane, thus providing a "soft" separation. The soft margin method will choose a hyperplane that splits the example as cleanly as possible while still maximizing the distance to the nearest training instances. The effects of the soft margin can be observed in Fig. 2.8.

Given a labeled dataset D of n tuples of the form: $D = \{\ (x_i, y_i)\ |\ x_i \in \mathbb{R}^p,\ y_i \in \{-1, 1\}\ \}$, the problem can be formulated as:

$$\min_{\omega,\xi} \left\{ \frac{1}{2}\|w\|^2 + C\sum_{i=i}^{n} \xi_i \right\},\ s.t\ \forall i:\ y_i * (\omega^t \cdot x_i + b\ ) \geq 1 - \xi_i,\ \xi_i \geq 0 \tag{2.14}$$

Each x_i is a p-dimensional real number vector. The hyperplane $\omega^t \cdot x + b = 0$ splits the space in two halves. The condition $y_i * (\omega^t \cdot x_i + b) \geq 1$ for any instance x_i is known as the margin condition, and ensures a margin γ on each side of at least $\frac{1}{\|\omega\|}$:

$$\gamma = \frac{|\omega^t \cdot x_i + b|}{\|\omega\|} \geq \frac{1}{\|\omega\|} \tag{2.15}$$

Finally, slack variables ξ_i allow the individual observations to be on the wrong side of the margin or hyperplane, while C is an input parameter that determines the cost of the errors, thus establishing a trade off between a large margin and the error penalty.

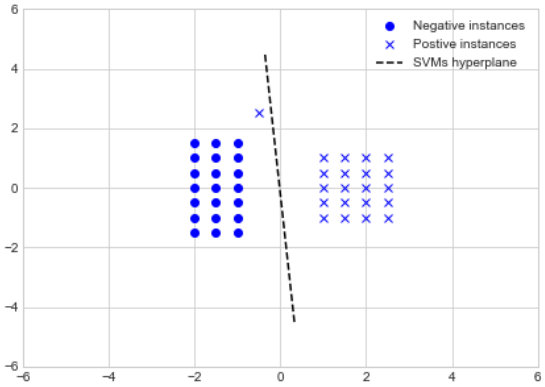


Fig. 2.8: When lowering the cost of misclassifications (C parameter), the SVM still prioritizes maximizing the margin, though it allows an instance to end in the wrong side of the hyperplane.

2.3 Equal Error Rate

In order to measure the performance of the different systems, the *Equal Error Rate* metric is chosen. As its name suggests, the *EER* prioritizes in an equal manner the *False Positive Rate* and the *False Negative Rate*. This metric was used in the previous works of the current line of investigation related to pronunciation assessment at phone level [17, 2], so the same approach was taken in the present work.

The process for evaluating the performance of a classifier usually involves the following steps: At first, the decision function of the classifier, which can be for example predicted probabilities or in our case distances to the hyperplane, is computed for each instance of the test set. Then, the obtained results are used to generate a distribution of the instances count as function of the values in the domain of the decision function. Finally, a threshold is chosen in order to make class predictions and the instances with a result of the decision function above that threshold are classified as positive, while those with results below the threshold are classified as negative. Most classifiers usually set a threshold by default, such as 0 in the case of Support Vector Machines that determines a separation of positives and negative instances according to the sign of their distance to the hyperplane.

The EER can be found by sweeping the threshold until reaching the condition that False Positive Rate (FPR) equals False Negative Rate (FNR). FPR is computed as the proportion of negative instances wrongly categorized as positive: $\frac{FP}{TN+FP}$. Analogously, FNR is computed as the proportion of positve instances wrongly categorized as negative: $\frac{FN}{TP+FN}$. A simple example

of EER threshold is shown in Fig. 2.9.

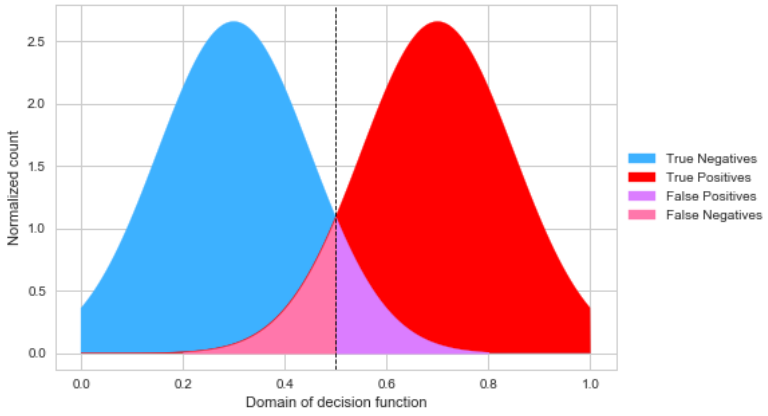


Fig. 2.9: An example of EER threshold with perfectly balanced classes. Both negative and positive distributions have the same size and shape. The threshold separates the instances in such way that FPR equals FNR.

2.4 Statistical Significance Techniques

In order to analyse if there is a statistically significant gain in combining the supervectors and the dynamic features over using supervectors as the only features, two different techiques are used in this work: *Bootstrapping* and *McNemar's Test*.

Bootstrapping is a technique that models the underlying distribution by using a resampling method on the available samples while McNemar's Test is an statistical test used on paired data.

An advantage for McNemar's over Bootstrapping is that it bases its comparisons on cross information using exactly the same sample. McNemar's Test however assumes that the samples are independent, which is not true when considering all the instances of a given phone. Even though Bootstrapping also assumes that the samples are independent, in the current work resampling is performed at speaker level, which ensures the independence of the instances being resampled. For this reason, we use both tests, since both have advantages and disadvantages with respect to the other one.

2.4.1 Bootstrapping

Bootstrapping [28] is a general intuitive method applicable to almost any kind of sample statistic and can be understood without much knowledge of sampling distributions. It was introduced in 1979 by B. Efron and it has been widely used in numerous areas since then.

Supposing that it were possible to draw repeated samples of the same size from the population of interest a large number of times, then a fairly good idea about the sampling distribution of a particular statistic could be obtained from the collection of values arising from these repeated samples. This method, however, does not make sense in practice since it assumes one has access to the underlying distribution. The idea behind *Bootstrapping* is to use the data of a sample study at hand as a "surrogate population" for the purpose of approximating the sampling distribution of a statistic. This is achieved by resampling with replacement from the sample data at hand, creating a large number of alternative or duplicated sample sets known as bootstrap samples. The process is usually repeated between 1 and 10 thousand times, and the performance metric, EER in our case, is computed on each of the bootstrap samples. A histogram of these computed values is referred to as the bootstrap distribution of the statistic.

In the current work, resampling is performed at speaker level. In other words, given a set S of n speakers in the test set, in each iteration of bootstrapping a new set of n speakers S' is obtained by drawing speakers with replacement from the original set S. The sample set is then obtained by gathering the instances of the speakers in S', which will include repeated samples for the speakers that appears more than once in S'. *Bootstrapping* technique is used along with Confidence Intervals in order to determine if the SVM trained with the combined features performs better than the one trained only with features derived from the GMM adaptation. The first step is to generate a distribution of EER for both types of classifiers. After that, a 95% confidence interval is computed for each distribution by finding the interval $[\hat{\theta}_{B1}, \hat{\theta}_{B2}]$ that enclose the 95% of the samples composing each distribution. If it is true that the SVM trained with the combined features performs better than the SVM trained with a single feature source, then its confidence interval should be shifted with respect to the confidence interval of the latter system. The bigger the overlap between both intervals, the lesser the evidence of the results coming from different distributions, thus resulting in less significant results.

2.4.2 McNemar's Test

McNemar's Test [29] is used as an alternative to determine whether the SVM based on the combined features and the SVM based only on the supervectors are significantly different.

Given a set of n test samples labeled by two systems, A_1 and A_2 the results can be summarized as:

	Correctly Predicted by A_2	Incorrectly Predicted by A_2
Correctly Predicted by A_1	$N_{0,0}$	$N_{0,1}$
Incorrectly Predicted by A_1	$N_{1,0}$	$N_{1,1}$

- $N_{0,0}$ is the number of samples which A_1 classifies correctly and A_2 classifies correctly
- $N_{0,1}$ is the number of samples which A_1 classifies correctly and A_2 classifies incorrectly
- $N_{1,0}$ is the number of samples which A_1 classifies incorrectly and A_2 classifies correctly
- $N_{1,1}$ is the number of samples which A_1 classifies incorrectly and A_2 classifies incorrectly

Let p_1 be the error rate of the system A_1 and p_2 the error rate of the system A_2. Let $q_{0,0}$ be the probability of (A_1 correct $\wedge$ A_2 correct), $q_{0,1}$ the probability of (A_1 correct $\wedge$ A_2 incorrect), and so on. Hence:

$$p_1 = q_{1,0} + q_{1,1} \qquad (2.16) \qquad\qquad p_2 = q_{0,1} + q_{1,1} \qquad (2.17)$$

The *null hypothesis* $p1 = p2$ is equivalent to $q_{1,0} = q_{0,1}$. Defining $q = q_{1,0}/(q_{1,0} + q_{0,1})$ the *null hypothesis* becomes $q = \frac{1}{2}$. The probability q is the probability that given that only one of the systems made an error, it was system A_1 that did. Under the *null hypothesis*, $N_{1,0}$ is distributed as a Binomial Distribution $B(n, 1/2)$, where n is the observed number of samples for which only one system made an error: $N_{1,0} + N_{0,1}$. At this point, a two-tail test is applied to obtain a *p-value* that represents the statistical significance of the results.

As it was said before, *McNemar's Test* is based on cross information by comparing the results of evaluating both systems on exactly the same sample. *Bootstrapping* differs on this point because each sample is evaluated independently for each system. A complementary analysis using both different techniques is carried out in this work in order to determine the statistic significance of the results.

3. EXPERIMENTAL SETUP

In this section, we will describe the database that is used in the current work. In addition, we will review the way the dataset is partitioned in order to carry out the experiments.

3.1 Database Description

The dataset for the current work is obtained from a Latin-American Spanish speech database [30]. The same database has been used in the previous work of the current line of investigation [2].

Even though the database includes recordings by natives and nonnative speakers, only the nonnative recordings were used in the current work. The nonnative data consists in 2550 utterances of read speech of different sentences taken from Spanish newspaper data, adding up to a total of 130,000 phone instances. The utterances were pronounced by 206 native American English speakers who had studied some Spanish locally or abroad. Their levels of proficiency were varied, and an attempt was made to balance the number of speakers by level of proficiency as well as by gender.

The set of sentences was chosen to maximize the number of occurrences of potential pronunciation problems gathered by a linguist and a Spanish language instructor. It was also intended to include phones in different contexts that are known to be difficult for native American English speakers to pronounce. Examples are the dipthtong [eu], /r/ after [l] [n] and [s] (which should be thrilled), and [p], [t] and [k] in any context (which nonnatives may aspirate).

Four native Spanish-speaking phoneticians provided the detailed phonetic transcriptions for the nonnative utterances. Labels for each phone instance were then generated by comparing the phonetician's transcriptions with the canonical transcriptions of the sentence: transcribed phone instances that matched the cannonical transcription were labeled as correct, while phone instances that did not match the cannonical transcription were labeled as mispronounced. When performing assessment tasks that involve several raters, usually there is a certain level of disagreement among the transcribers. In fact, not all the phones were transcribed with the same level of reliability. *Cohen's Kappa* coefficient K [31] is used to describe how reliably the transcribers agree on the transcriptions for each of the native phones. For eight of the phones (/β/, /δ/, /γ/, /b/, /w/, /m/, /i/, /s/), all four transcribers showed at least a moderate level of

agreement (using $K > 0.4$ to mean "moderate" agreement).

3.2 Development Set and Hold-Out Set Definitions

The previously described dataset is partitioned into different subsets to apply some statistical methods that usually improves the robustness of the experiments when performing machine learning tasks. At a higher level, the data is divided into two groups: the *development* set and the *hold-out* or *validation* set. The former is used to fit the different models while the latter is used to evaluate the performance of the final model on unseen data. At the same time, the *development* set is divided in four subsets in order to apply *k-fold cross validation* ($k = 4$), a technique that leads to more accurate estimates of the errors when exploring different models in the training phase.

3.2.1 Hold-Out Set

The *hold-out* set or *validation* set is a subset of the data that is kept separately of the development set and remains untouched until the final model is fitted. It is then used to test the final model and verify if it generalizes to unseen data. Additionally, it allows an unbiased estimate for the test error because none of the instances was used in the fitting process. It is worth noting that in order to keep this condition true, the samples of the hold-out set must come from speakers that are not used in the development set.

The partitioning is carried out at speaker level. Speakers are randomly split into two groups that contain 20% and 80% of the total number of speakers. The hold-out set is obtained from the group with 20% of the speakers by gathering all the utterances pronounced by those speakers. The development set is obtained from the group with 80% of the speakers in the same way.

3.3 K-Fold Cross Validation

K-Fold Cross Validation [26] is a method which aims to improve the accuracy in the estimate of the errors for a given model by fitting and testing the model using different subsets of the available development data. The error estimate is more robust than that computed when fitting the model only once using all the available training data.

In order to keep the error estimates unbiased the set of speakers in each of the folds must be disjoint (the same as when splitting the dataset into development and hold-out). The par-

titioning of the development set is then carried out at speaker level, and each fold is obtained by gathering all the utterances pronounced by the speakers in that fold.

To apply K-Fold Cross Valdation, the development set is divided in k groups or folds of approximately equal size. Then, for each fold i ($1 \leq i \leq k$) the following procedure is repeated:

1. A new model is fitted using the instances of all the folds but the i^{th} fold.

2. The fitted model is tested against the instances of the i^{th} fold.

The results of all the iterations are then combined to obtain a final estimate of the error.

K-Fold Cross Validation maximizes the number of both training and testing instances. Every instance is used exactly once for testing, and $k - 1$ times for training the different models. It also allows an unbiased estimation of the results, because in none of the iterations the same instance is used for both training and testing the model.

In the current work, Cross Validation is performed in the development set using 4 folds. On each iteration, the different models and features are computed using the instances of three of the folds and tested on the other one. EER is computed by gathering the scores from all the folds. This approach for EER computation leads to a more realistic statistic than computing an EER per fold and taking the average, because the latter would lead to optimistic results.

4. EXPERIMENTS AND RESULTS

In this section we introduce the experiments that were carried throughout this work along with the obtained results for each of them. The outline of the experiments can be summarized as follows:

1. At first, initial or baseline experiments are performed in order to replicate the results obtained in the previous work [2] for both GMM and SVM classifiers.

2. In a second stage, different parameters for the dynamic features are explored. Those parameters include the optimal weight to be used when combining the supervectors with the dynamic features.

3. At last, the final model is selected and it is tested against the hold-out data.

4.1 Baseline Experiments

The initial experiments are carried out in order to replicate some of the results obtained in [2]. By succesfully achieving this, it could be assumed that the models are being properly trained and use them as baseline to build the new system.

The results to be replicated come from the comparison between GMM and SVM models. The GMMs are obtained by adaptation of UBM-GMMs, as it was previously described in the method section 2.1.2, and they are based on a likelihood-ratio detection method. The SVM model approach is based on supervectors, as it was also described in 2.1.2.

Some of the models' parameters are tuned during the baseline experiments and kept throughout the subsequent experiments. For each phone's GMM, the number of gaussian mixtures to be trained is kept proportional to the number of available training instances. A mixture component is trained every 15 phone instances in order to keep a similar proportion to what was used in the previous work [2]. With regard to the SVM classifier, different values of the input parameter C, which is described in section 2.2.1, were tested. The best results were obtained when using $C = \frac{1}{avg(x*x)}$, which is the average of the squared norm of the training feature vectors. This value was taken from the documentation of SVM-Light, a library that implements SVMs in C language. Finally, *sklearn* provides a *class-weight* parameter, that allows to adjust the weight of

the C parameter for each class. If not given, all samples are assumed to have the same weight. This parameter is set to "balanced", which adjusts the weights inversely proportional to the class frequencies in the input data. This is particularly useful because the number of instances for each class is unbalanced for most of the phones.

Only SVMs with linear kernels are used throughout the experiments, i.e, solutions involving other types of kernels such as polynomial or radial are not explored. This is because of two reasons: In the first place, the dimension of the features space is relatively high compared with the number of instances for each phone, and SVMs with linear kernels usually perform better on this scenarios. Secondly, as it was mentioned before, the current work is based on a previous work [2], which uses a linear kernel. The strategy used in the current thesis is to keep the initial configuration for the SVM classifier while focusing on exploring the new set of dynamic features.

Additionally, features are standardized by removing the mean and scaling to unit variance before training the models to avoid the dominance of certain features over other ones.

Below is shown the comparison of the results of the GMM models and SVM models obtained in the development set during the experiment replication (Fig. 4.1). The phones are sorted in descending order according to their Kappa coefficients. Results show that the SVMs system produced an overall weighted EER relative reduction of 2.5% with respect to the Adapted GMMs, which is in line with the results obtained in [2].

Additionally, there seems to be some correlation between the phones with high Kappa and those who achieved the best results. This seems reasonable, because it may exist clearer differences between correctly and mispronounced utterances of the phones that produced better agreement over the transcribers. These differences may lead the classifiers to perform better. Among the eight phones with higher Kappa values ($K > 0.4$, which implies moderate agreement): /β/, /δ/, /γ/, /b/, /w/, /m/, /i/, /s/, only /s/ exceeds an EER of 0.3.

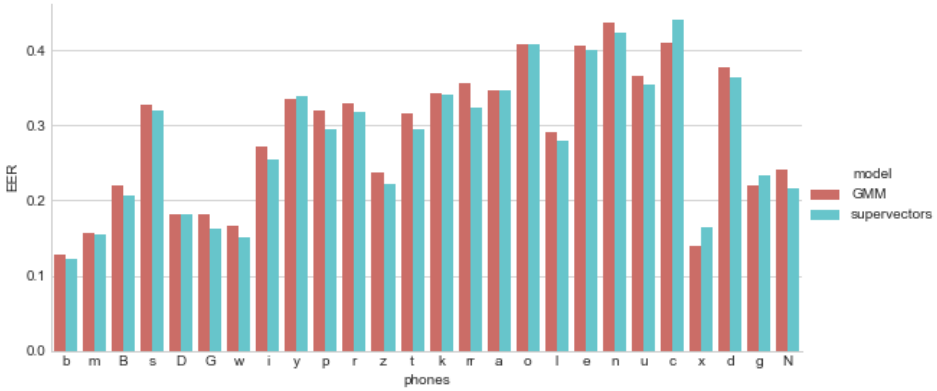


Fig. 4.1: Comparison of EER between Adapted GMMs and SVM trained on supervectors for all phones obtained in the development set, sorted in descending order according to their Kappa values.

At the end of the experimental phase, the same experiment was carried out in the heldout set leading to the results shown below (Fig. 4.2). The models are trained using all the instances in the development set, namely, the instances of the 4 folds. They both have a very similar performance for each phone compared with the results obtained in the development set (Fig. 4.1), from which one can deduce that both classifiers generalize well to unobserved data. This time, however, it was the Adapted GMM model which produced an overall weighted EER relative reduction of 0.8% with respect to the SVMs trained on supervectors. Even though the expected result would have been that the SVM performed slightly better than the GMM, as it happened in the development set, the degradation is less than 1% so it is still a reasonable result.

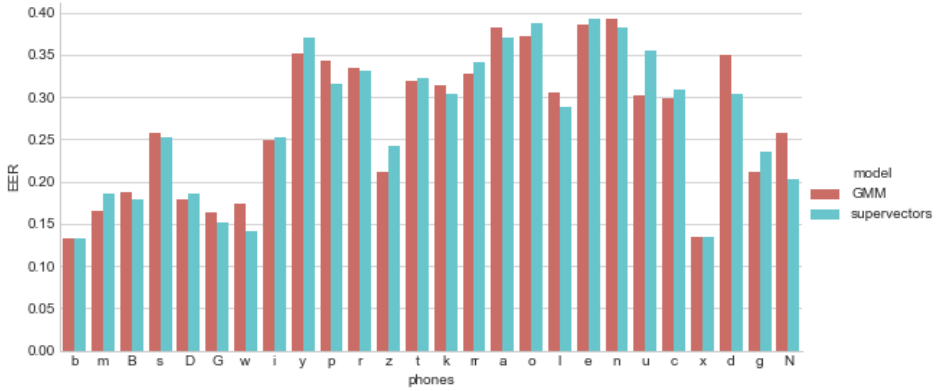


Fig. 4.2: Comparison of EER between Adapted GMMs and SVM trained on supervectors for all phones obtained in the hold-out set, sorted in descending order according to their Kappa values.

After the successful replication of the initial experiments, a validated SVM model based on supervectors is obtained. This classifier is used as baseline and starting point for the next experiments.

4.2 Tunning Experiments

Tunning experiments are carried out in order to compare both alternative dynamic features. Features are first analysed in isolation to determine their best possible configurations.

After that, the features generated with those configurations are combined with the supervectors and the results of those combinations are compared to each other.

Tunning is performed in the development set using 4-Fold Cross Validation. Results are calculated only for the phones with high Kappa coefficients by averaging them. Only these phones are considered in the process because they are the most reliable ones.

The most important parameter to be fitted is the number of coefficients to be used for both methods. We assume this parameter to be phone-independent, so it is shared among all the phones. This helps to keep the models simple by reducing the number of phone-dependent configuration parameters.

4.2.1 Legendre Best System

The objective of the Legendre tunning experiments is to find the optimum degree of the Legendre Polynomials along with the best configuration. Tunning the best Legendre features involves the analysis of the following factors:

- Legendre Polynomial degree
- Time axis normalization between -1 and 1
- Inclusion of duration
- Lasso-Regression regularization

Results are depicted in Fig. 4.3, which shows the variation of the EER average taken over the phones with high Kappa as function of the degree of the Legendre Polynomials for different configurations. The analysis is restricted to polynomials up to degree 6 because no further gains are obtained when using polynomials of higher degrees.

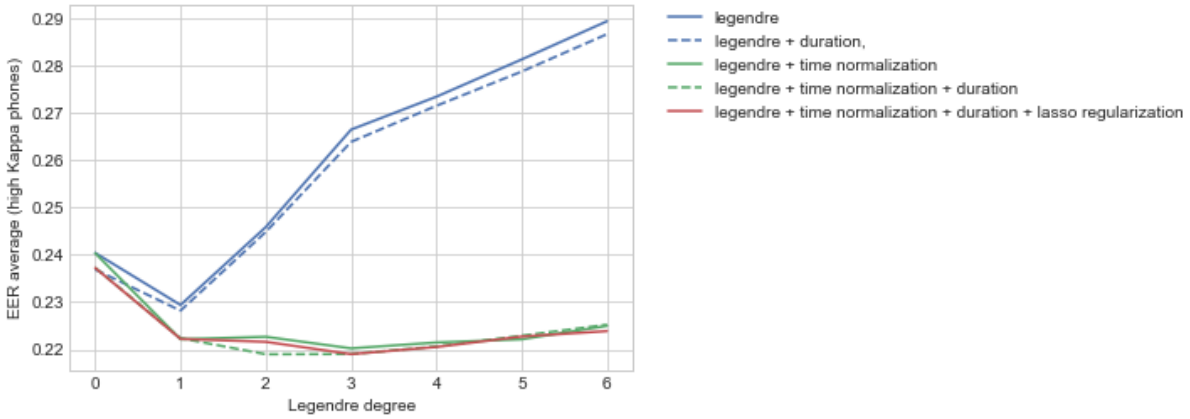


Fig. 4.3: EER average over phones with high Kappa for different degrees of Legendre Polynomials obtained in the development set. Additional configurations are also studied along with the degree and represented by different curves: time normalization, appending the duration and applying Lasso Regression.

Time normalization before approximating the MFCCs turns out to be an essential property for the Legendre Polynomials, evidenced by a considerable reduction of the EER average (gap between the blue and green curves). At the same time, appending the duration to the features produces a slight improvement in the performance. Lasso Regularization, however, does not contribute to reduce the average EER. Results show that the optimal configuration is to model the dynamics of the temporal dependencies using Legendre Polynomials of degree 2, along with normalizing the time axis and appending the duration of the phone utterances to the features.

4.2.2 DCT Best System and Comparison Between Features

Analogous to Legendre, the objective of the DCT tunning experiments is to find the optimum number of coefficients of DCT to approximate the dynamics of the temporal dependencies. In the DCT case, the only additional parameter to be determined is whether or not appending the duration to the DCT coefficients.

As in the Legendre optimal configuration, DCT tunning experiments also shows that appending the duration to the coefficients produces a slight reduction in the averaged EER (around 1% relative), so the duration is appended to the DCT coefficients.

A comparative plot between the variation of the EER as function of the number of coefficients for both dynamic features is shown below in Fig. 4.4. The additional parameters are fixed according to their optimal configuration for both techinques: Normalizing the time axis and appending the duration for Legendre (without Lasso Regularization), and appending the duration for DCT. The degree of the Legendre Polynomials is mapped to the number of coeffi-

cients in the *x-axis* to allow the comparison: the 0 degree Legendre Polynomial is mapped to 1 coefficient, the 1 degree Legendre Polynomial is mapped to 2 coefficients, and so on.

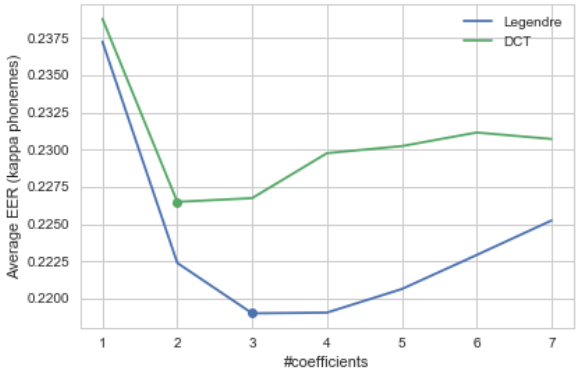


Fig. 4.4: EER average over phones with high Kappa as function of the number of coefficients for both Legendre Polynomials and DCT obtained in the development set.

A dot is placed on each curve showing the optimal number of coefficients for both techniques. As it is known from Fig. 4.3, the optimal number of coefficients for the Legendre Polynomials is 3 (Polynomials of degree 2). Polynomials with 4 coefficients performs almost as well as polynomials with 3 coefficients, but it is always preferable to keep the model as simple as possible. On the other hand, the optimal number of coefficients for DCT is 2.

The results show that when the analysis is done in isolation (in absence of the supervectors features), Legendre Polynomials lead to a better performance, generating a relative reduction of the average EER of 3.3% compared with DCT.

4.2.3 Dynamic Features and Supervectors Combination Comparison

Even though the SVM based solely on Legendre performs better than the SVM based solely on DCT, a new experiment is carried out in order to determine which dynamic feature integrates better with the supervectors producing the best results. Two types of combination are studied in the current thesis: a Features Combination and a Score Combination.

In order to describe how the Features Combination is computed we recall some concepts previously mentioned in the Method section. As it is described in equation 2.14, a labeled set $D = \{ (x_i, y_i) \mid x_i \in \mathbb{R}^p,\ y_i \in \{-1, 1\} \}$ is used when training the SVM for a given phone, where x_i represents the feature vector for the i^{th} instance of that phone and y_i represents the label.

Let us denote:

$$x_i = \begin{cases} s_i : [\ s_{i1}, s_{i2}, \dots s_{in}\ ], & \text{when using supervectors features} \\ d_i : [\ d_{i1}, d_{i2}, \dots d_{im}\ ], & \text{when using dynamic features} \end{cases} \tag{4.1}$$

Supervectors features are n-dimensional vectors while dynamic features are m-dimensional vectors. When combining the features, dynamic features are appended to supervectors. A factor $c = \frac{n}{m}$ multiplies the dynamic features in order to compensate the difference in the number of features between supervectors and dynamic features. Additionally, a weight factor $w \in \mathbb{R}$ that also multiplies the dynamic features is chosen to determine the priority, or importance *a priori* of the dynamic features with respect to supervectors within the SVM (the weight for supervectors is implicitly set to 1). The higher the value of w, the less regularized are the dynamic features at the beginning. The features combination vector for the i^{th} instance is then computed as:

$$x'_i = s_i \circ (w * c * d_i) = [\ s_{i1},\ s_{i2},\ \dots s_{in},\ w * c * d_{i1},\ w * c * d_{i2},\ \dots w * c * d_{im}\ ] \tag{4.2}$$

where symbol "$\circ$" represents concatenation.

Score Combination, on the other hand, is computed by combining the scores of the SVM classifiers trained independently on both supervectors and dynamic features. Again, a weight $w \in \mathbb{R}$ is chosen to determine the amount of the score of the dynamic-feature-based system to be considered in the final score. Let us denote the score of the SVM trained on supervectors when classifying the i^{th} instance as sup_i and the score of the SVM trained on dynamic features when classifying the i^{th} instance as dyn_i.

The final score for the i^{th} instance is computed as:

$$\text{score}_i = \text{sup}_i + w * \text{dyn}_i \tag{4.3}$$

The range of values to be explored for the parameter w was adjusted empirically for both Features Combination and Score Combination. The obtained interval to be explored for w was [0.0 - 1.0] with steps of 0.1. It is worth noting that choosing a value for w of 0.0 leads to not considering the dynamic features at all.

The experiment is again scoped to the phones with high Kappa values. Weights to be used

when combining the features and the scores are kept phone-dependent. The objective is thus to compute two optimal weights for each phone: one for the features combinations and the other one for the score combination.

In the Features Combination experiment, the optimal weight for the majority of the phones for both Legendre and DCT approaches turns out to be 0.1. There are also some phones such as /γ/ and /b/ for which the best weight is 0.0. This can be interpreted as that no weight produces an improvement when combining the dynamic features with the supervectors.

On the other hand, in the Score Combination experiment, the optimal weight for the results of the different phones are more scattered along the whole interval [0.0 - 1.0] for both Legendre and DCT. Again, there are also some phones for which the optimal weight is 0.0, but they are considerable fewer than in the features combination case.

For both combination types, the DCT features yield better results than the Legendre features, though the results for both features are very close to each other (around 1% of relative difference), as it is shown in Table 4.1.

	Legendre + Supervectors	DCT + Supervectors
EER Features Combination (Avg)	0.188	0.187
EER Score Combination (Avg)	0.189	0.187

Tab. 4.1: EER average over phones with high Kappa using the best weight for both Dynamic Features and combination approaches, obtained in the development set.

At this point, the DCT features are chosen as the Dynamic Features to be combined with the supervectors when training the final models, and for each of the remaining phones with low Kappa values the phone-dependent weights are computed.

Even though training a system from the combination of the three features (Supervectors, Legendre and DCT) was possible, it would have required the tunning of an additional parameter for each phone, which is costly to mantain. Moreover, this could have lead to overfitting the model. Because of these reasons, and the fact that we weren't expecting significant gains in combining both types of dynamic features because they model the same temporal characteristics, we didn't explore a combination of the three features as a solution.

4.3 Final Models Validation

The final experiments are carried out in order to determine if there is a real gain when combining supervectors with DCT features compared with training an SVM based only on supervectors. Additionally, we are interested to see if any of both fusion systems is better than the other one. The three systems to be compared are then:

- SVM trained only on supervectors, which is used as the baseline system.
- SVM trained on the features combination of DCT and supervectors (using for each phone the best features combination weights obtained in the tunning phase).
- System based on the score combination of an SVM trained only on supervectors and an SVM trained only on DCT features (using for each phone the best score combination weights obtained in the tunning phase).

An initial experiment is performed in the development set followed by a second experiment, where the final models are tested against the heldout data. The relative gains of the final models are calculated using the SVM trained only on supervectors as baseline system.

Unlike the tunning phase, where the experimentation is driven by the results obtained over the phones with higher kappa values, a new criteria is used for the model validation. A McNemar's test is performed as previous step to scope the analysis to a subset of phones with statistically significant gains in the development data ($p\text{-}value < 0.05$). McNemar's, which is described in Method Section 2.4.2, is used to estimate the confidence that a final model is different from the baseline model by using the paired nominal results of both classifiers.

4.3.1 Development Results

The three systems are trained applying 4-Fold Cross Validation in the development set and then compared to each other. The EER for each of the systems is computed by gathering the scores from all the folds. In addition, McNemar's test is also computed over the scores from all the folds to determine if the results yield statistically significant gains.

The phones for which the McNemar's test gives significant results in the development set are:

- /s/, /y/, /e/, /o/, /l/, /n/, /m/ for Features Combination

- /s/, /y/, /e/, /o/, /l/, /z/, /n/, /m/, /rr/, /t/ for Score Combination

The relative gains for those phones are summarized in Fig. 4.5 (results for all the phones are available in tables 6.1 and plots 6.2 of the Appendix). It is worth noting that phones that give statistically significant gains in the Score Combination experiment are a superset of the phones that give statistically significant gains in the Features Combination experiment. Three of the phones (/z/, /rr/, /t/) give statistically significant gains when combining the scores but not when combining the features. For that reason, only the bars associated with the Score Combination results are included for those phones.

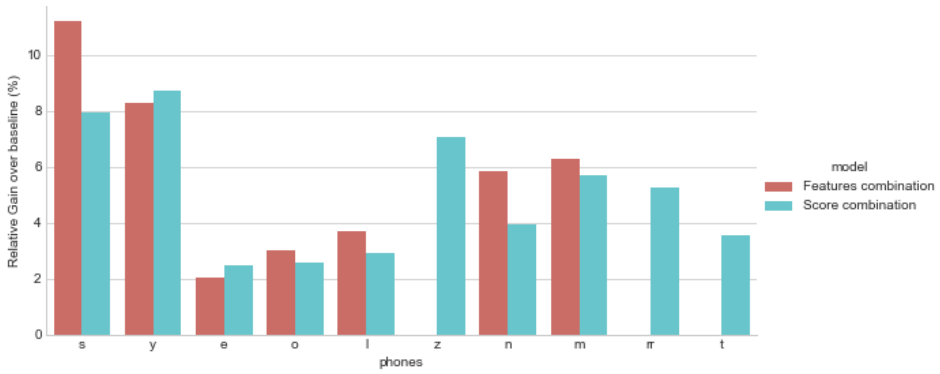


Fig. 4.5: Comparison of relative gains over the Baseline systems between the Features Combination systems and the Score Combination systems computed on the development set. Results are sorted in descending order according to their McNemar's p-values.

A Bootstrapping test is also performed in the development set to provide a complementary analysis. As it is described in Method Section 2.4.1, the main goal is to compute the confidence intervals that enclose the 95% of the samples of the target distributions (in our case, EER distributions) to find out if they are shifted with respect to each other. A thousand bootstrapping iterations are applied by resampling the speakers and computing the EER over those speakers' instances. Results are summarized in Fig. 4.6. Plots are split in two in order to adjust the y-axis and display the intervals with greater detail.

The plots show that most of the fusion system intervals are shifted downwards with respect to the supervectors-based baseline system, which is a sign that fusion systems perform better than the baseline systems. The only exception seems to be /z/, which despite having almost identical confidence intervals for both systems, it yields a significant difference between the results obtained for the baseline system and the Score Combination system.

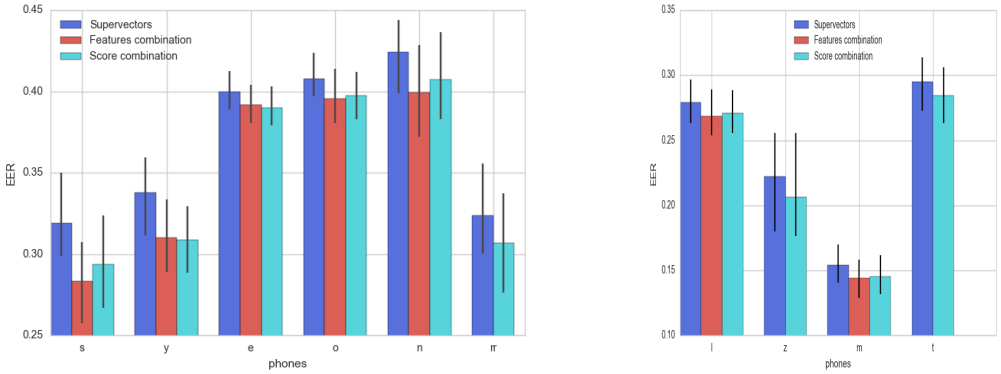


Fig. 4.6: Comparison of 95% confidence intervals for the baseline and fusion systems. A thousand iterations of Bootstrapping are applied in the development set for the phones that gave statistically significant gains in the McNemar's test.

4.3.2 Hold-out results

In this experiment, the three aforementioned systems are trained using all the instances of the development set and tested against the hold-out set.

The analysis is again scoped to our phones of interest, which are those with statistically significant results in the development set. The results for all the phones, however, are included in the tables (6.1) and plots (6.2) of the Appendix. Even though a new McNemar's test is computed on the results obtained in the hold-out data, in many cases it does not yield statistically significant gains for the phones of interest. This can be explained by the fact that hold-out data has around 1/4 of the instances of the development set, which results in very little testing data affecting the efectiveness of the statistical test.

The relative gains for the phones of interest are summarized in Fig. 4.7. Again, the bars for the three phones (/z/, /rr/, /t/) that gave statistically significant gains when combining the scores but not when combininig the features in the development set are only included for the Score Combination experiment.

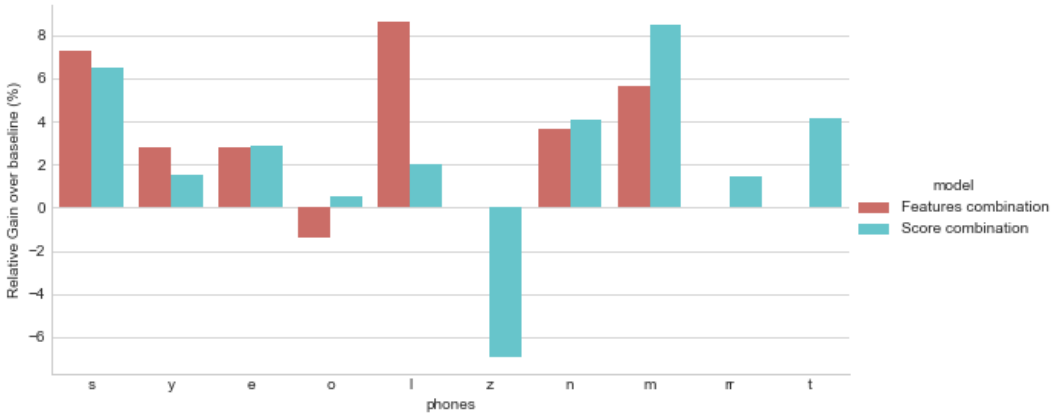


Fig. 4.7: Comparison of relative gains over the Baseline systems between Features Combination systems and Score Combination systems computed on the hold-out set for the phones of interest. Results are sorted in descending order according to their McNemar's p-values computed on the development set.

Results show that, for the phones that gave statistically significant gains in the Score Combination experiment carried out in the development set, only 'z' degrades its performance compared with the baseline system (around a 6% relative). Among the remaining phones, some reduce its gain compared with the development set (such as /y/ from 8% to less than 2% and /rr/ from almost 6% to less than 2%). Others such as /m/ improves its gain from 6% to more than 8%.

On the other hand, results show that for the phones that gave statistically significant gains in the Features Combination experiment carried out in the development set, only /o/ degrades its performance and only slightly. Among the remaining phones, for the Features Combination experiment there are cases where both improvements and degradations of the performance are observed.

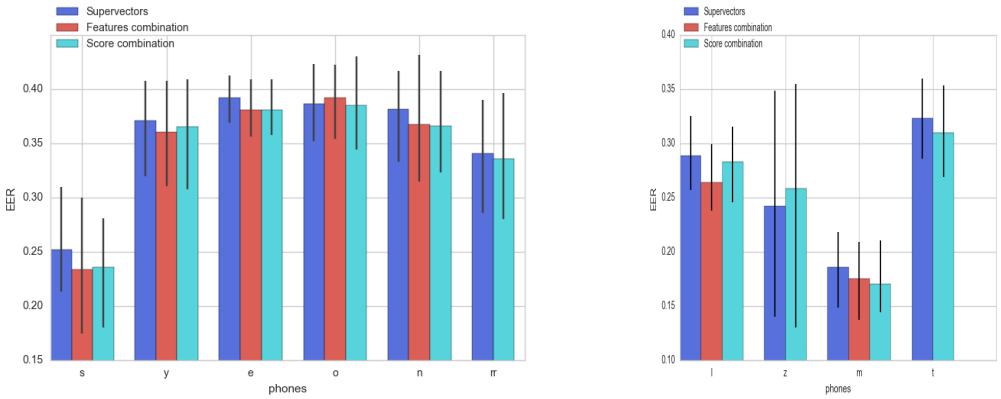
At the very beginning, when restricting our analysis to this particular subset of the phones, we picked out some of them such as 'rr' and 'm' and inspected their instances in order to figure out why those instances would benefit from the dynamic features. However, what we found for example when analysing the utterances of the 'rr', is that many of the instances labeled as correct did not sound thrilled, and the other way around too. Many of the instances that sounded thrilled for us, were labeled as incorrect. In other words, it seems to be a mismatch between our criteria and annotators' criteria. A similar thing happend when analysing the utterances of the 'm'. After these findings, we abandon the initiative of trying to explain the reasons why some of the phones benefits from the dynamic features. Also, there exists the possibility that for some of the phones the boundaries of performance imposed by the annotation errors have

already been reached.

A Bootstrapping test is again performed in the hold-out set to provide a complementary analysis, and the results are summarized in Fig. 4.8. The plots are again split in two in order to adjust the y-axis and display the intervals with greater detail.

The most relevant thing to mention is that, unlike in the development set, the overlap between the confidence intervals of the three systems is very big, to a point such that for some phones it is difficult to determine if the intervals come from different distributions. There are, though, some phones for which the intervals for the fusion systems seem to be shifted downwards with respect to the baseline system. The most representative example is /s/, followed in lesser extent by other phones such as /l/, /m/, /t/, /e/ and /y/.

A positive aspect to be considered is that none of the phones degrade their results in a significant manner. In addition, the confidence intervals for the two systems for the phone /z/ are very wide and they almost completely overlap, which is an evidence that the degradation is not statistically significant.



(a) *(b)*

Fig. 4.8: Comparison of 95% confidence intervals for the baseline and fusion systems. A thousand iterations of Bootstrapping are applied in the hold-out set for the phones of interest.

5. CONCLUSIONS

In the current thesis, we explored different Dynamic Features to be used as complementary features to Supervectors when classifying phone instances using an SVM classifier. As in the previous work [2], Support Vector Machines stands as a suitable model for discriminating mispronounced and correctly pronounced phone instances.

When tested in isolation, Dynamic Features turned out to be effective features in regard to phone assessment, producing better results than a naive system. The results for both Legendre and DCT methods were very similar, yet DCT produced slightly better results in combination with supervectors.

Experiments showed that among all the phones that were analysed in the current work, only a subset of them benefits from the combination of supervectors and dynamic features. A proper strategy when following this technique is then to combine the features using a phone-dependent approach.

The obtained results suggest that in fact dynamic features carry information complementary to supervectors' information, which leads to a reduction of the errors made by the classifiers. For most of the phones for which the combination experiments gave statistically significant gains in the development set, the experiments in the hold-out set also yielded gains, indicating that the approach generalizes well to unseen speakers.

The number of instances for each phone was insufficient in many cases. The classes for many phones are highly unbalanced, which led to training models with high variance an hindered the computation of reliable statistics for the errors. Some possible consequences of this, that can be observed in the statistical analysis carried out in the hold-out set, could be the fact that bootstrapping confidence intervals were very wide for some of the phones, or that McNemar's test did not yield statistically significant results for many of the phones for which the same test had given statistically significant results in the development set.

Finally, it is not possible to determine if any of the combination approaches is better than the other based on the obtained results. Both Features Combination and Score Combination approaches yielded similar results with no clearer differences. Again, using a bigger dataset would have helped to pick out one method over the other.

5.0.1 Future Work

As it was mentioned in section 1.1, a new dataset of argentinian children reading english phrases is currently being collected and annotated. In the future, we plan to verify our findings on this dataset, which will be significantly bigger than the one used in this work.

Due to the increasing number of highly effective DNN-based methods that can be found in many fields (including pronunciation assessment, as it was described in the Previous Work section 1.2), it would be worthwile to keep exploring DNN-based solutions to phone-based pronunciation assessment, using some training algorithms and features that we have predefined as starting point.

Finally, a different approach that is worth studying, even though it is not standard in the literature on pronunciation assessment, is the usage of *Detection Cost Function* (DCF) to explore an alternative performance metric instead of Equal Error Rate. DCF combines both *False Positive Rate* and *False Negative Rate* according to a given cost, which is an input parameter. As consequence, the system can focus on different operating points by varying the value of the cost parameter and thus prioritizing, in different degrees, one measure over the other one. In the context of pronunciation assessment in CALL systems, labeling a correctly pronounced utterance as incorrect should be penalized stronger than labeling a mispronounced utterance as correct, in order to avoid discouraging the students. Because of those reasons, we think that DCF would be a more suitable metric for this scenario.

6. APPENDIX

This appendix provides complementary information obtained in the model validation phase of the experiments.

6.1 Model Validation Results Tables for All Phones

In this section we include two tables that summarizes the results obtained when validating the final models for both fusion types: Features Combination and Score Combination.

The tables include information of the results for all phones, and they are split in two: the left side contains the results obtained in the development set, while the right side contains the results obtained in the hold-out set. Along with the EER values obtained in each dataset and the p-values obtained in the McNemar's test, the number of instances for each class and each phone are included. The table is sort by decreasing Kappa values of the phones.

Rows with grey background corresponds to the phones for which the McNemar's test gives significant results in the development set, which are the phones that are analized in the Experiments and Results section 4.3.

6.1.1 Features combination

	Development						Hold-Out						
phones	base EER	F.C EER	delta	pvalue	#pos	#neg	base EER	F.C EER	delta	pvalue	#pos	#neg	kappa
b	0.122	0.122	%0.0	1	528	395	0.133	0.133	%0.0	1	140	95	0.9
m	0.154	0.144	%6.5	0.047	3234	686	0.186	0.176	%5.4	0.268	801	188	0.76
B	0.206	0.2	%2.9	0.374	428	1169	0.179	0.189	-%5.6	0.523	126	302	0.7
s	0.319	0.283	%11.3	8.32E-19	7555	480	0.252	0.234	%7.1	0.454	1963	107	0.57
D	0.182	0.182	%0.0	1	920	2009	0.185	0.185	%0.0	1	268	486	0.55
G	0.162	0.162	%0.0	1	222	643	0.152	0.152	%0.0	1	61	145	0.51
w	0.151	0.151	%0.0	1	743	500	0.141	0.141	%0.0	1	179	128	0.43
i	0.254	0.252	%0.8	0.481	4929	1238	0.252	0.256	-%1.6	0.714	1224	301	0.41
y	0.338	0.31	%8.3	2.02E-05	2453	574	0.371	0.361	%2.7	0.844	596	156	0.39
p	0.295	0.295	%0.0	0.715	1657	1055	0.315	0.316	-%0.3	1	441	254	0.36
r	0.317	0.315	%0.6	0.561	3650	2617	0.331	0.315	%4.8	0.046	910	641	0.36
z	0.222	0.222	%0.0	1	189	997	0.242	0.242	%0.0	1	49	247	0.35
t	0.295	0.288	%2.4	0.183	2938	1542	0.323	0.314	%2.8	0.207	733	360	0.34
k	0.341	0.335	%1.8	0.339	1708	1472	0.304	0.305	-%0.3	0.913	434	388	0.32
rr	0.324	0.314	%3.1	0.299	491	1739	0.341	0.357	-%4.7	0.359	122	453	0.29
a	0.346	0.345	%0.3	0.712	10144	2069	0.371	0.354	%4.6	0.041	2509	548	0.26
o	0.408	0.396	%2.9	0.002	8040	2077	0.387	0.392	-%1.3	0.598	2030	548	0.23
l	0.279	0.269	%3.6	0.019	3505	1373	0.289	0.264	%8.7	0.009	851	356	0.22
e	0.4	0.392	%2.0	0.022	10597	3484	0.392	0.381	%2.8	0.095	2658	899	0.18
n	0.424	0.399	%5.9	1.27E-04	7152	476	0.382	0.368	%3.7	0.772	1792	125	0.15
u	0.354	0.344	%2.8	0.296	1948	482	0.355	0.336	%5.4	0.308	471	110	0.14
x	0.164	0.164	%0.0	1	590	153	0.135	0.162	-%20.0	1	161	37	-
d	0.364	0.364	%0.0	1	773	89	0.304	0.304	%0.0	1	191	18	-
g	0.234	0.232	%0.9	0.222	887	114	0.236	0.276	-%16.9	2.16E-07	237	29	-
N	0.217	0.198	%8.8	0.088	911	443	0.203	0.217	-%6.9	0.296	246	116	-

Tab. 6.1: Results of SVM system trained on the Features Combination of supervectors and DCT, for both development and hold-out sets, compared to the SVM trained only on supervectors (baseline system). F.C. stands for Features Combination.

6.1.2 Score combination

	Development						Hold-Out						
phones	base EER	S.C. EER	delta	pvalue	#pos	#neg	base EER	S.C EER	delta	pvalue	#pos	#neg	kappa
b	0.122	0.122	%0.0	1	528	395	0.133	0.133	%0.0	1	140	95	0.9
m	0.154	0.145	%5.8	0.01	3234	686	0.186	0.17	%8.6	0.233	801	188	0.76
B	0.206	0.199	%3.4	0.096	428	1169	0.179	0.185	-%3.4	0.289	126	302	0.7
s	0.319	0.294	%7.8	3.67E-07	7555	480	0.252	0.236	%6.3	0.004	1963	107	0.57
D	0.182	0.179	%1.6	0.327	920	2009	0.185	0.193	-%4.3	0.146	268	486	0.55
G	0.162	0.162	%0.0	1	222	643	0.152	0.159	-%4.6	1	61	145	0.51
w	0.151	0.151	%0.0	1	743	500	0.141	0.141	%0.0	1	179	128	0.43
i	0.254	0.248	%2.4	0.074	4929	1238	0.252	0.248	%1.6	0.039	1224	301	0.41
y	0.338	0.309	%8.6	7.40E-06	2453	574	0.371	0.365	%1.6	0.842	596	156	0.39
p	0.295	0.295	%0.0	1	1657	1055	0.315	0.315	%0.0	1	441	254	0.36
r	0.317	0.316	%0.3	0.285	3650	2617	0.331	0.328	%0.9	0.832	910	641	0.36
z	0.222	0.206	%7.2	0.002	189	997	0.242	0.259	-%7.0	0.774	49	247	0.35
t	0.295	0.285	%3.4	0.024	2938	1542	0.323	0.31	%4.0	0.056	733	360	0.34
k	0.341	0.333	%2.3	0.102	1708	1472	0.304	0.283	%6.9	8.78E-04	434	388	0.32
rr	0.324	0.307	%5.2	0.022	491	1739	0.341	0.336	%1.5	0.913	122	453	0.29
a	0.346	0.344	%0.6	0.883	10144	2069	0.371	0.362	%2.4	0.139	2509	548	0.26
o	0.408	0.398	%2.5	2.13E-04	8040	2077	0.387	0.385	%0.5	1	2030	548	0.23
l	0.279	0.271	%2.9	4.31E-04	3505	1373	0.289	0.283	%2.1	0.152	851	356	0.22
e	0.4	0.39	%2.5	1.78E-04	10597	3484	0.392	0.381	%2.8	0.022	2658	899	0.18
n	0.424	0.407	%4.0	0.006	7152	476	0.382	0.366	%4.2	0.103	1792	125	0.15
u	0.354	0.343	%3.1	0.134	1948	482	0.355	0.327	%7.9	0.02	471	110	0.14
c	0.44	0.404	%8.2	0.01	405	105	0.308	0.353	-%14.6	0.442	104	24	-
x	0.164	0.159	%3.0	1	590	153	0.135	0.112	%17.0	0.227	161	37	-
d	0.364	0.361	%0.8	0.28	773	89	0.304	0.319	-%4.9	0.031	191	18	-
g	0.234	0.228	%2.6	0.774	887	114	0.236	0.238	-%0.8	0.625	237	29	-
N	0.217	0.203	%6.5	0.303	911	443	0.203	0.215	-%5.9	0.015	246	116	-

Tab. 6.2: Results of the Score Combination between the SVM system trained on supervectors and the SVM system trained on DCT coefficients, for both development and hold-out sets, compared to the SVM trained on supervectors only (baseline system). S.C stands for Score Combination.

6.2 Model Validation Plots for All Phones

In this section we include two plots that summarizes the results obtained when validating the final models in both development and hold-out set.

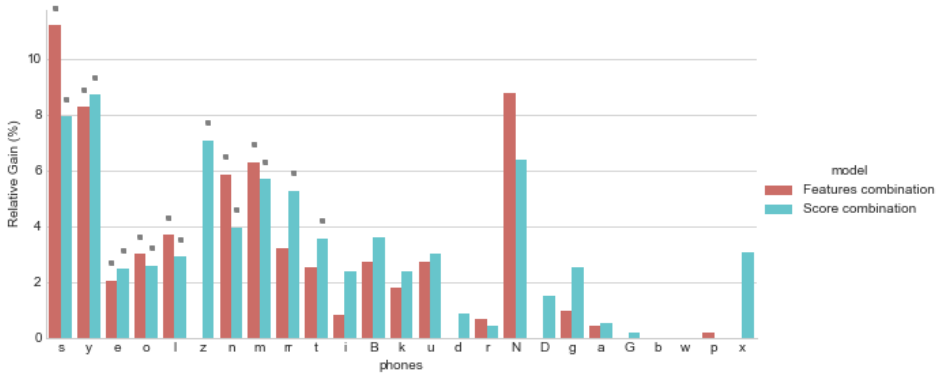


Fig. 6.1: Comparison of relative gains over the Baseline Systems between Features Combination systems and Score Combination systems computed on the development set for all the phones. Results are sorted in descending order according to their McNemar p-values computed on the development set. Only the results marked with (*) gave statistically significant gains in the development set.

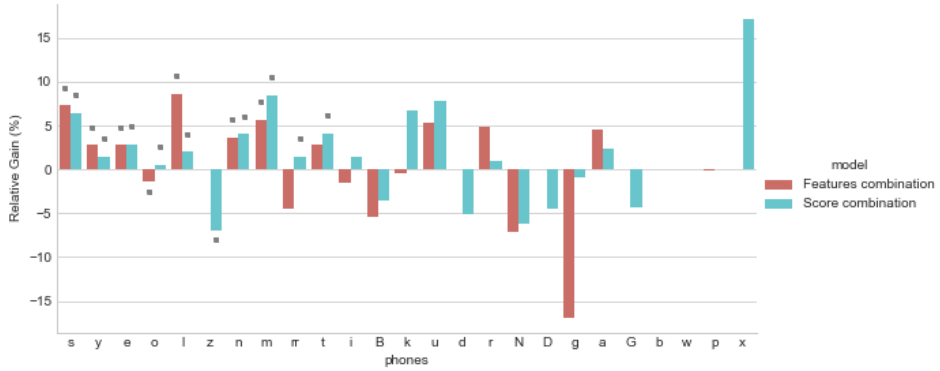


Fig. 6.2: Comparison of relative gains over the Baseline Systems between Features Combination systems and Score Combination systems computed on the hold-out set for all the phones. Results are sorted in descending order according to their McNemar p-values computed on the development set. Only the results marked with (*) gave statistically significant gains in the development set.

BIBLIOGRAPHY

[1] Y. Kim, H. Franco, and L. Neumayer. "Automatic pronunciation scoring of specific phone segments for language instruction". In: *Proc. EUROSPEECH 97* (1997).

[2] H. Franco, L. Ferrer, and H. Bratt. "Adaptive and discriminative modeling for improved mispronunciation detection". In: *ICASSP 14* (2014).

[3] H. Franco, L. Neumayer, Y. Kim, and O. Ronen. "Automatic pronunciation scoring for language instruction". In: *Proc. ICASSP 97* (1997).

[4] S. Witt and S. Young. "Language learning based on non-native speech recognition". In: *Proc. EUROSPEECH 97* (1997).

[5] S. Witt and S. Young. "Phone-level pronunciation scoring and assessment for interactive language learning". In: *Speech Comunication* (2000).

[6] S. Kanters, C. Cucchiarini, and H. Strik. "The Goodness of Pronunciation Algorithm: a Detailed Performance Study". In: (2009).

[7] H. Wang, X. Qian, and H. Meng. "Predicting gradation of L2 english mispronunciations using crowdsourced ratings and phonological rules". In: *Proc. of Speech and Language Technology in Education (SLaTE)* (2013).

[8] A. Harrison, W Lo, X. Qian, and H. Meng. "Implementation of an extended recognition network for mispronunciation detection and diagnosis in computer-assisted pronunciation training". In: *Proc. of the 2nd ISCA Workshop on Speech and Language Technology in Education* (2009).

[9] S. Yoon, M. Hasegawa-Johnson, and R. Sproat. "Automated pronunciation scoring using confidence scoring and Landmark-based SVM". In: (2009).

[10] J. Van Doremalen, C. Cucchiarini, and H. Strik. "Automatic Detection of Vowel Pronunciation Errors using multiple information sources". In: (2009).

[11] S. Wei, H. Guoping, Y. Hu, and R. Wang. "A new method for mispronunciation detection using Support Vector Machine based on Pronunciation Space Models". In: (2009).

[12] S. Yoon, M. Hasegawa-Johnson, and R. Sproat. "Landmark-based Automated Pronunciation Error Detection". In: (2010).

[13] H. Strik, K. Truong, F. De Wet, and C. Cucchiarini. "Comparing classifiers for pronunciation error detection". In: (2009).

[14] "Using Deep Neural Networks to Improve Proficiency Assessment for Children English Language Learners". In: *Interspeech 2014* (2014).

[15] W. Hu, Y. Qian, and F. Soong. "A New DNN-based High Quality Pronunciation Evaluation for Computer-Aided Language Learning (CALL)". In: *Interspeech 2013* (2013).

[16] W. Hu, Y. Qian, F. Soong, and Y. Wang. "Improved mispronunciation detection with deep neural network trained acoustic models and transfer learning based logistic regression classifiers". In: *Speech Communication 2015* (2014).

[17] H. Franco, L. Neumayer, M. Ramos, and H. Bratt. "Automatic detection of phone-level mispronunciations for language learning". In: *Proc. Eurospeech 99* (1999).

[18] D. Reynolds, T. Quatieri, and R. Dunn. "Speaker Verification Using Adapted Gaussian Mixture Models". In: *Digital Signal Processing 10* (2000).

[19] S. Davis and P. Mermelstein. "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1980).

[20] W. Campbell, D. Sturim, D. Reynolds, and A. Solomonoff. "SVM based speaker verification using a GMM supervector kernel and NAP variability compensation". In: *ICASP* (2006).

[21] D.A Reynolds. "Gaussian Mixture Models". In: *MIT Lincoln Laboratory* ().

[22] C.M Bishop. "Pattern Recognition and Machine Learning". In: *Springer* (2006).

[23] Anli Fikret and Gungor Suleyman. "Some useful properties of Legendre polynomials and its applications to neutron transport equation in slab geometry". In: *Applied Mathematical Modelling* (2007).

[24] N. Dehak, P. Dumouchel, and P. Kenny. "Modeling Prosodic Features with Joint Factor Analysis for Speaker Verification". In: *IEEE Transactions on Audio, Speech and Language Processing* (2007).

[25] M. Kockmann, L. Ferrer, L. Burget, and J. Cernocky. "iVector Fusion of Prosodic and Cepstral Features for Speaker Verification". In: *Interspeech* (2011).

[26] James, Witten, Hasti, and Tibshirani. "An Introduction to Statistical Learning". In: *Springer* (2013).

[27] C. Corinna and V. Vapnik. "Support-Vector Networks". In: *Machine Learning, volume 20* (1995).

[28] K. Singh and M. Xie. "Bootstrap: A Statistical Method". In: (2008).

[29] Luciana Ferrer. "Statistical Modeling of Heterogeneous Features for Speech Processing Tasks". In: *Ph.D. Dissertation, Standford University* (2008).

[30] H. Bratt, L. Neumeyer, E. Shriberg, and H. Franco. "Collection and detailed transcription of a speech database for depelomnet in language learning technologies". In: *ICSLP* (1998).

[31] S. Siegel and J. Castellan. "Nonparametric Statistics for the Behavioral Sciences, Second Edition". In: *McGraw-Hill* (1988).