Language Identification Based on Detection of Phonetic Characteristics

Vegar Enersen Vindfallet

Master of Science in Electronics
Submission date: June 2012
Supervisor: Torbjørn Svendsen, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

The starting point of this thesis is to use an excisting recognizer for phonetic characteristics as a front-end of a language recognition system. Instead of a purely phonotactic approach, an alternative approach of vector space modeling (VSM) is to be the basis for the recognizer back-end.
The assignment is to implement a VSM-based language recognizer based on a phonetic attribute tokenizer and to investigate alternatives to performing the final classification such as Support Vector Machines (SVM) and Gaussian Mixture Models (GMM).

Abstract

This thesis has taken a closer look at the implementation of the back-end of a language recognition system. The front-end of the system is a Universal Attribute Recognizer (UAR), which is used to detect phonetic characteristics in an utterance. When a speech signal is sent through the UAR, it is decoded into a sequence of attributes which is used to generate a vector of term-count. Vector Space Modeling (VSM) have been used for training the language classifiers in the back-end. The main principle of VSM is that term-count vectors from the same language will position themselves close to eachother when they are mapped into a vector space, and this property can be exploited for recognizing languages.

The implemented back-end has trained vectors space classifiers for 12 different languages, and a NIST recognition task has been performed for evaluating the recognition rate of the system. The NIST task was a verification task and the system achived a equal error rate (EER) of 6.73%. Tools like Support Vector Machines (SVM) and Gaussian Mixture Models (GMM) have been used in the implementation of the back-end. Thus, are quite a few parameters which can be varied and tweaked, and different experiments were conducted to investigate how these parameters would affect EER of the language recognizer. As a part test the robustness of the system, the language recognizer were exposed to a so-called out-of-set language, which is a language that the system has not been trained to handle. The system showed a poor performance at rejecting these speech segments correctly.

Sammendrag

Denne oppgaven har tatt en nærmere titt på implementeringen av back-end for et språkgjenkjennelses-system. Front-end av systemet er en Universal Attributt Gjenkjenner (UAG), som brukes til å oppdage fonetiske egenskaper i en ytring. Når et talesignal sendes gjennom UAG-en, blir det dekodet til en sekvens av attributter som igjen brukes til å generere en vektor med tellere. VektorRoms-Modellering (VRM) har blitt brukt til trening av språk-klassifisererne i back-end. Hovedprinsippet for VRM er at teller-vektorene fra det samme språket vil posisjonere seg nær hverandre når de blir tegnet i et vektorrom, og denne egenskapen kan utnyttes for å gjenkjenne språk.

Den implementerte back-end har trent vektorroms-klassifiserere for 12 forskjellige språk, og en gjenkjennelses-oppgave fra NIST har blitt utført for å evaluere gjenkjennelses-raten til systemet. NIST-oppgaven var en verifikasjon-oppgave og systemet oppnådde en Equal Error Rate (EER) på $6,73\%$. Verktøy som Support Vector Machines (SVM) og Gaussiske Blandings-Modeller (GBM) har blitt brukt i implementasjonen av back-end. Dermed er ganske mange parametere som kan varieres og skiftes, og forskjellige eksperimenter ble utført for å undersøke hvordan disse parametrene vil påvirke EER-en til språk-gjenkjenneren. For å teste robustheten i systemet, ble språk-gjenkjenneren utsatt for et såkalt out-of-set-språk, som er et språk som systemet ikke har blitt opplært til å håndtere. Systemet var ikke spesielt flink til å klassifisere disse talesegmentene riktig.

Acknowledgements

I would like to thank Professor Torbjørn Svendsen for supervising this thesis. Without his expertise and experience, this report would never have seen the light of day. Thanks to fellow student Åsmund Tokheim for explanatory discussions and for giving me tips and tricks on both programming and on the technical work. I would also like to thank all of my other fellow Master students who have contributed with an impeccable and motivational spirit at the study office.

Contents

Chapter 1

Introduction

Language recognition (LRE) is a task which is getting more and more important in the field of speech recognition. As a result of globalization, it is getting more common to encounter speech and conversations on different languages, and LRE could be an important tool to handle this information in a correct way. Language recognition have several fields of applications, for instance in multilingual call-centers where the language spoken by the caller needs to be detected, or for indexing the recordings in a speech data base. LRE could also be used as a pre-processing step for a multi-lingual speech recognizer. In addition, the use of loan words from other languages are becoming quite usual, and detection of these words could possibly help to improve the speech recognition.

There are several characteristics in speech that can be used to differentiate between languages, but for this thesis, languages will be identified by looking at the phonotactic content of a speech signal. The main idea is that an utterance can be regarded as a sequence of features, and languages can be distinguished by looking at how frequently these features occur. In addition, it's possible to look at combinations (n-grams) of subsequent features and make a model which describes how often they appear for each language. When people are exposed to a language they don't understand, they tend to rely on the phonotactics in order to make a qualified guess. To identify the spoken language they listen for certain cues, like the nasal vowels in French, consonant cluster in Eastern Europe languages or the characheristic vowel endings of Italian.

Most Language Identification (LID) systems uses some form of tokenization when classifying a speech signal. This tokenization is, in most cases, performed by sending the acoustic signal through a phoneme recognizer, and the decoded phoneme string is used for further processing. Phoneme recognizers are actually language dependant, which can cause trouble, for instance, if we want to design language identifiers for languages where there are little training data. An alternative approach to the process of tokenization is to do a detection of the phonetic characteristics (parameters which describes how sounds are physically realized). The phonetic characteristics are language independant, which means that a universal recognizer can be built for all languages.

There are some diserable advantages of using phonetic characteristics (attributes) compared to the use of phonemes. First of all, the attributes are universal for all languages, which means that the set of attributes doesn't need to be extended if the set of target languages is extended. This is not the case when using phonemes since a language that is added to the set of target languages may include phonemes that haven't been observed earlier. Secondly, if more training data becomes available, the attribute recognizer can be re-trained, regardless of which language new material origins from. The starting point of this thesis is to use an existing universal attribute recognizer (UAR) as a front-end of a LRE system. The task is to implement a back-end with classfiers based on the vector space model (VSM) approach. The main principle of the VSM is that the content in the decoded sequence of attributes from the front-end is used to generate a document vector, and this vector can be mapped as a point in a multi-dimensional vector space. Different utterances from the same language will form a set of document vectors which will be positioned close to each other in this vector space. Hopefully, these clusters of document vectors for each language will be separated in such a way that they can be used to uniquely describe a language. In a recognition task a new document vector, $\tilde{d}$, is generated based on the test utterances and it is mapped in the same vector space. By looking at which cluster of document vectors that is positioned closest to the document vector $\tilde{d}$, the language can be identified.

The report will start off by explaining some more background information on language recognition in the next chapter. In chapter 3.1, the UARs will be presented along with the necessary theory for the VSM-backend. In chapter 4, the conducted experiments are explained and the results are presented. There will also be a discussion in the same chapter. Finally, a conclusion will be drawn in chapter 5.

Chapter 2

Language Recognition

2.1 Language Characterization

There are multiple features which characterize a spoken language, and knowledge of these features is useful and necessary when developing an LRE system. The main principle of LRE is to build statistical model for each target language, by using one or more of these features, in order to separate the languages from one another. The following list mentions some examples of language characteristics that may be used for recognition purposes [2]:

- **Spectral characteristics:** Purely acoustic features with no linguistic content.
- **Prosody:** Patterns of duration, pitch and stress may differ from one language to another.
- **Phonological information:** Information about a language's phonemic inventory, phonotactics and articulatory features.
- **Lexical information:** Different languages have different vocabularies.

2.2 Approaches

There are several ways of performing language recognition, but roughly speaking, the approaches can be divided into two categories:

2.2.1 Spectrally Based LRE

This approach uses only the acoustic information in a speech signal to generate a set of feature vectors which describe the language. These features can be extracted by applying short time analysis to the speech signal, while linguistic data, such as words or phones, are not used at all. Mel-Frequency Cepstral Coefficients (MFCC) are a common choice of features, but Shifted-Delta Cepstral (SDC) features are often preferred as it provides information on temporal evolution. A decision is made by evaluating the features in a bank of classifiers to find the language that have the highest probability of producing the known observations.

2.2.2 Token Based LRE

In token-based LRE, speech is decoded into a sequence of labels using a tokenizer. The most common tokens used in LRE are phones, which can be used to build n-gram language models. This is the case for PRLM (Phone Recognition followed by Language Modeling), which has proven to be an effective system for recognizing languages [12]. However, the use of phone recognizers has some major drawbacks since the recognizers are language dependant, and labeled data is needed for each recognizers. By using an universal tokenizer these problems will be eliminated.

2.3 Types of Recognition Tasks

There are two different types of recognition tasks that will be evaluated in this thesis. The first tasks is language identification, which is the process of identifying a language of a spoken utterance from a closed set of target languages. The other task is language verification, where a speech signal is presented along with a claim that the utterance belong to a certain language. In this case, the task is to determine whether this initial hypothesis is true or false. When performing language verification, the utterance may belong to a language which the system haven't encountered before, and this complicates the task. However, the verification task is in general an easier task than the identification task.

Chapter 3

System Overview

A language recognition system is usually divided into a front-end and a back-end. The front-end will perform some kind of feature extraction and represent an utterance as a sequence of features. The back-end of the system will use these features to classify which language the utterance belongs to. A block diagram of the LRE system can be seen in figure 3.1, and each block will be described step-by-step in this chapter.

3.1 Universal Attribute Recognizers

The UARs that will be used in the following LRE system was developed by *Sabato Marco Siniscalchi* at *Kore University of Enna*, for researching purposes in connection with the work in [10]. The architecture of the UARs can be seen in figure 3.2. The recognizers perform feature extraction by looking at the temporal context of a windowed segment of an utterance. It splits the segment into a left context (LC) and a right context (RC) and evaluates the temporal changes between them. The classifiers are based on a hybrid of the hidden Markov model and an artificial neural network (HMM/ANN), and the Viterbi algorithm is used to determine the label with the highest state probability at a point in an utterance. Both of the UARs for manner and place attributes have a set of 12 different labels (including silence), which can be seen in table 3.1.

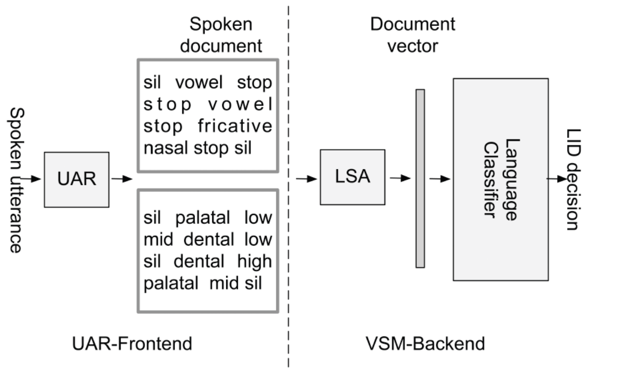

Figure 3.1: Block diagram of the LRE system, separated into a UAR-Frontend and a VSM-backend [10].

The output from the UARs will be two sets of transcriptions (manner-based and place-based) for each utterance.

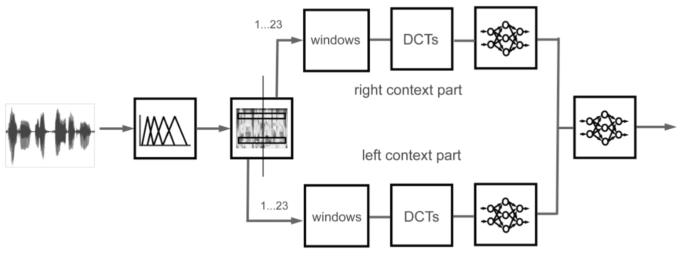

Figure 3.2: The structure of the UAR, which is based on the split temporal context idea.

A part of the OGI-TS corpus, which has phonetic transcriptions for six different languages (English, German, Hindi, Japanese, Mandarin and Spanish), was used to train the two UARs. On average, less than 1 hour of transcribed data from each language was used, which is a fairly small amount compared to the amount that is normally used when training phonotactic tokenizers. The performance of the UARs on the OGI-TS test sentences, for different number

Table 3.1: The sets of attribute labels for the UAR

manner	affricate, diphtong, flap, fricative, glide, liquid, nasal, sibilant, unvoiced-stop, voiced-stop, vowel, silence
place	alveolar, alveo-palatal, bilabial, dental, glottal, high, labio-dental, low, mid, palatal, velar, silence

of attributes, have been investigated in [10]. The LRE system in this thesis will use UARs with the maximum number of attributes (12, including silence), and their respectively error rates can be seen in table 3.2.

Table 3.2: Error rate for the two UARs on the OGI-TS test files when all 11 attribues (excluding silence) are being used.

	# attributes	**Error rate**
Manner	11	30.40 %
Place	11	34.50 %

3.2 Modeling n-grams

The first step in the back-end is to generate a term-count vector by analysing the transcriptions from the UAR. An utterance will be decoded by the UAR and tokenized into a set of speech features. This set, called a spoken document, contains speech units that are drawn from a fixed inventory, $U = u_1, u_2, ..., u_J$, which consists of J different attributes. The attributes are counted with respect to how many times they appear in the spoken document, and this result in a term-count vector of length $M = J$. The UAR tokenizers for both place and manner transcriptions chooses attributes from a set of $J = 12$ symbols. This vector is too small for doing any descent language recognition, so in order to improve the statistical resolution, terms are modeled as n-grams. This means that in addition to counting occurences of single attributes (unigrams), occurences of order-dependent groups of attributes are counted, i.e. bi-grams ($u_i u_j$, a pair of attributes), tri-gram ($u_i u_j u_k$, group of three subsequent attributes), and so on. The number of acoustic labels and n-grams must be chosen carefully when implementing an LRE system, and there is an important trade-off involved in this decision. In order to cover all of the acoustic variations in a language, M need to be large, but if it's too large it will result in a system with high computational complexity. The system implemented in this thesis will use n-grams up to the order of 4 (quad-grams), and since the manner tokenizer has a set of $J = 12$ different attributes, resulting in a term-count vector with $M = J + J^2 + J^3 + J^4 = 22620$ terms. The place transcriptions also have 12 different attributes as well, which results in a term-count vector with a total of 45240 terms.

3.3 Latent Semantic Analysis (LSA)

Generating the term-count vector, described in chapter 3.2, can be regarded as the first step in the latent semantic analysis. The term-count vectors, d_m and d_p, that are generated using respectively manner and place transcriptions, can be evaluated seperately in order to investigate the effect of using each set of attributes. By merging the count-vectors from both place and manner transcriptions, the seperation in the LSA procedure will improve. This is simply done by concatenating the two term-count vectors from the same utterance, $d = [d_m^T \; d_p^T]^T$. In training, a document matrix is generated by using all the available training data from all languages, and each column in this matrix corresponds to a term-count vector. The next step is to normalize the entropies of the data by analyzing the set of vectors and the terms they contain. This is because some terms appear frequently in all languages and these terms are therefore less informative in terms of distinguishing between languages. Terms that occur regularly for a language, but rarely for other languages are more describing and needs to be weighted more heavily [1]. The result after performing the weighting of the count values is a term-document matrix, $W = \{w_{i,j}\}$, where

$$w_{i,j} = (1 - \epsilon_i)\,\frac{n_{i,j}}{n_j} \tag{3.1}$$

$$\epsilon_i = -\frac{1}{\log N}\sum_{j=0}^{N}\frac{n_{i,j}}{n_i}\log\frac{n_{i,j}}{n_i} \tag{3.2}$$

$n_{i,j}$ is the number of times term i appear in document j, n_j is the number of terms in document j and n_i is the number of times term i appears in total in the N training documents. The value $(1 - \epsilon_i)$ will be close to zero if a term appears uniformely across all languages, and close to 1 if a term only appears in a few document vectors. The resulting term-document matrix, W, will have a dimension of $M \times N$.

The term-count vectors have very high dimentionality, and in addition, they are quite sparse since only a minority of the terms occur in an utterance. Singular Value Decomposition (SVD) are used inn order to reduce the computational complexity of these long vectors. This is the final step in the LSA procedure. During training, the SVD procedure performs a factorization of the term-document matrix into the following form:

$$W \approx \widehat{W} = USV^T. \tag{3.3}$$

U have a dimentionality of $M \times Q$, S is $Q \times Q$ and V is $N \times Q$. The value of Q, is the rank of approximation matrix $\widehat{W}$, and it has to be chosen. This value determines how many singular values will be retained and used for the final classification. If $Q = M$ the matrices W and $\widehat{W}$ will be equal, but the value should be chosen as $Q << M$. The main purpose of the SVD is to convert the term-count space into a new concept space with much lower dimentionality, by retaining only the largest singular values. When performing language recognition, an utterance is transcribed by the UAR and a term-count vector is generated. The next step is to make a pseudo vector by using the U matrix which was calculated during training. The reduced Q-rank pseudo vector is calculated as

$$\tilde{d_Q} = d^T\, U \tag{3.4}$$

This results in a vector whose dimentionality has been reduced from M = 45280 to rank Q. Although the dimensionality is severely reduced, the most describing data is hopefully retained, which means that two document vectors with similar content (i.e. two documents from the same language) will have a small distance between themselves in the Q-rank concept space.

3.4 Classifier Using Support Vector Machines (SVM)

At this point, the term-count vectors are represented as pseudo document vectors, and vector-based classifiers must be built to perform language recognition. There excist several techniques for doing so, but the system described in this paper uses a classifier based on Support Vector Machines (SVM). For each language, a 1-versus-all classifier is trained by using the pseudo document vectors from the LSA, and this will result in a bank of binary SVMs. The training documents are labeled x_+ if the data belongs to the target language of the specific SVM, and they are labeled x_- if they belong to another language. The system trains a classifier of the form $f(d) = a^T \phi(d) + b$, where a is the vector weight, b is the offset, and $\phi(.)$ denotes the kernel function. In training, the goal is to seperate the data from the two classes with a hyperplane while maximizing the margin (the distance between the hyperplane and the nearest training vectors). An example of different seperation possibilities using a linear kernel, and how it affects the margin, can be seen in figure 3.3. It is possible to alter the posistion of the hyperplane and deliberately misclassify some of the training data to achieve a wider margin.

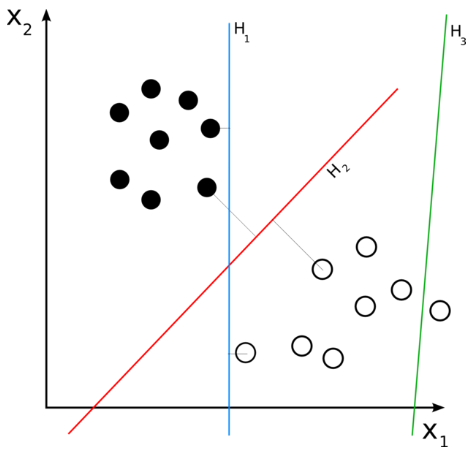

Figure 3.3: H3 (green line) doesn't seperate the two classes. H1 (blue) does, but the margin is very small. H2 seperates the classes with a maximum possible margin.

However, the class data aren't always linearly separable. In that case, a non-linear kernel could be used in order to improve the accuracy of the classifier (see figure 3.4). These kernels perform a transformation and map the training data into a Euclidian space where they are linearly separable. In the LRE system, both classifiers with linear and non-linear kernels will be trained and their performances will be evaluated and compared.

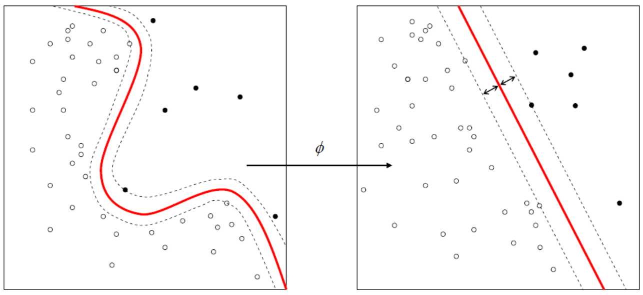

Figure 3.4: Non-linear kernel function used for better fitting of the training data.

3.5 GMM and Final Decision

For each language, a pair of GMMs are modeled. The training documents for a target language are evaluated by the bank of SVMs and uses the SVM distances (the distances from the seperating hyperplane and the training vectors) to model the first GMM. The other GMM, which will describe a so-called anti-target model, is modeled by using the SVM distances from all other languages. When performing a verification task, the pair of GMMs for a claimed language are evaluated, and the final decision is made based on the log-likelihood ratio which is compared to a threshold. Two possible errors can be made when doing verification, namely a *false rejection* (rejection of a hypothesis which is true) or *false alarm* (accept a hypothesis which is wrong). There's a bit of a trade-off between these two errors, and the threshold in the final decision will affect these two error rates. By lowering this threshold less false rejections will be made, but in return, the probability of false alarms will increase. Results from a system evaluation are usually expressed in terms of equal error rate (EER), which is the the point when the probability of a false alarm is equal to the probability of a false rejection. Figure 3.5 shows a more detailed sketch of the classifier. When performning a identification task, the languages' model GMM is evaluated by using the SVM distances, and the language with the highest probability is selected.

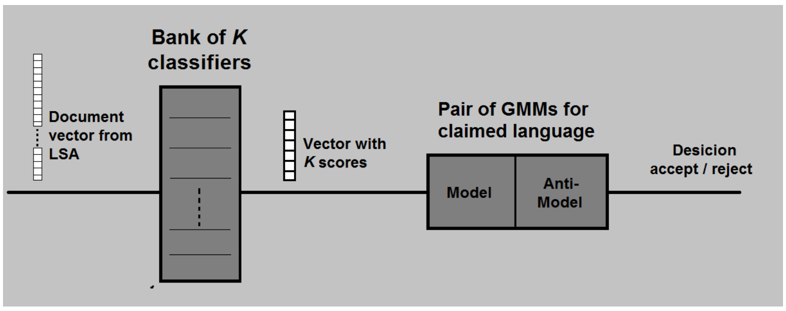

Figure 3.5: A more detailed block diagram of the classifier in the back-end. K denotes number of trained languages in the system.

Chapter 4

Experiments, Results and Discussion

In this chapter, the executed experiments will be explained and the system's performance will be evaluated. The recognition rate of the system will primarily be investigated by performing a verification task on the NIST 2003 evaluation set [7]. The goal is to decide whether a hypotesized language is present or not in a given utterance, and the performance will be expressed in terms of EER. This evaluation set consists of test utterances with a duration of 30 seconds. The languages included in this set are the same twelve languages found in the CallFriend corpus and there are 80 segments of each language. In addition, there are 80 test segments with Russian speech to evaluate the system's robustness with respect to an out-of-set language, and English and Japanese have an additional number of test segments of 160 and 80, respectively, from other corpora. This result in a total of 1280 test segments. There are several factors and variables in the implementation that can be changed, and the effect of varying these will also be evaluated. Finally, a language identification test (identifying the languange of an utterance from a closed set) will be executed. Details concerning the results will be discussed as they are presented, and a more general discussion will be presented in the final section.

4.1 CallFriend Corpus

Speech from the CallFriend database will be used for training and evaluation of the language identifier. The database contains several hours of telephone conversations for twelve different languages (see 4.1). Three of the languages (English, Mandarin and Spanish) have available data for two different dialects, but only one of these dialects are chosen when training the VSM backend.

Table 4.1: Languages included in CallFriend

Arabic	English	Farsi	French
German	Hindi	Japanese	Korean
Mandarin	Spanish	Tamil	Vietnamese

The length of the conversations in the database varies somehow, but most of the recordings have a length of 30 minutes. However, the files are continous recordings, and as a result, they contain a considerable amount of silence. Thus, the amount of speech in each recording are somewhat less than 30 minutes. In total, for each language, there are approximately:

- 10 hours of training data - used for building the initial VSM.
- 10 hours of development test data (*devtest*) - can be used for testing during development in order to tweak on variables that creates the most optimal VSM.
- 10 hours of evaluation test data (*evltest*) - used for evaluting the system. These data must be unknown for the system until the final evaluation tests.

4.2 Results

4.2.1 Singular Value Resolution

This first experiment will investigate the number of singular values needed to achieve a good seperation between languages. This step is done in the LSA procedure, where the term-document vector is mapped into a Q-rank space in order to reduce the dimentionality and the sparsity problem. The results from the NIST 2003 task evaluation is shown in figure 4.1. Terms in the document vector are modeled up to quad-grams (4-grams). The graph to the left shows the results when the term-count vectors from the UMR and UPR transcriptions are modeled and evaluated seperately. The error rate decreases when more singular values are retained, but as the curves approach 200 both of them flatten out. Thus, the number of retained singular values in all of the subsequent experiments are chosen to be 200. The right panel of figure 4.1 shows the error rate when the manner and place documents are combined, and this result verifies that the selection of 200 singular values was a good choice. The EER achived in this experiment was, respectively, 13.2% and 12.1% for manner and place transcriptions. When these transcription were merged, the system attained an EER of 7.1%.

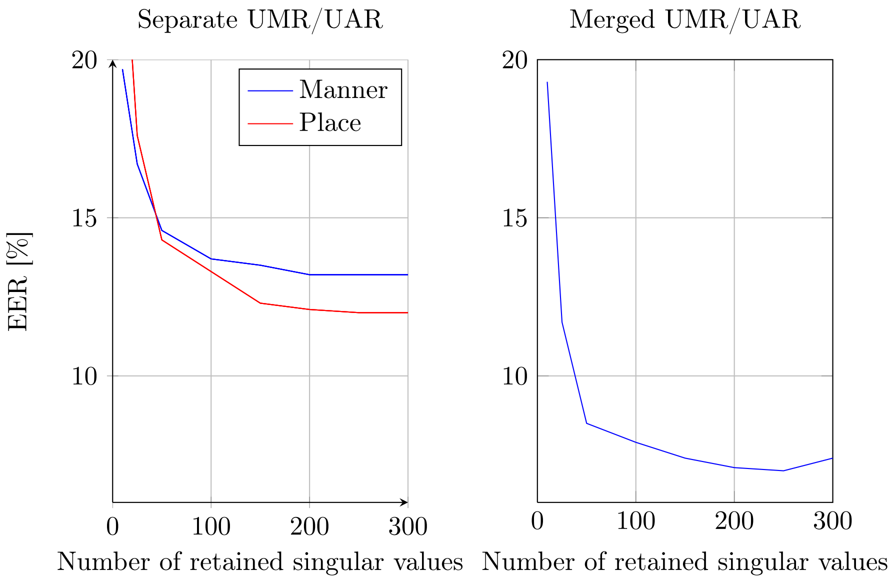

Figure 4.1: EER for NIST 2003 task with respect to number of retained singular values. Result for seperated manner and place transcriptions to the left, and merged document vectors to the right. These results were achieved using classifiers modeled with LIBLINEAR [5].

4.2.2 Different Kernel Functions

The experiments described in the previous section were all executed using classifiers with linear kernels. For this part, the use of unlinear kernels will be investigated. The use of a *radial basis function*, which has been proven effective in previous work [11], and a polynomial kernel will be evaluated. The classifiers were implemented using LIBSVM [3]. Cross validation was used by dividing the training data into five folds. Four folds are used for training and the last one is used for validation. By doing so, one can find the optimal parameters which assures a well fitted decision boundary for the classifier.

Table 4.2: The system's EER on the NIST 2003 task for different types of kernels.

Kernel Type	**EER**
Linear	7.03 %
Polynomial (second order)	7.11 %
Radial basis function	6.73 %

4.2.3 Number of Mixture Components in GMM

The final step in the back-end is to evaluate a pair of GMMs with respect to the outputted SVM distances from the classifiers. These two GMMs are, respectively, a model and an anti-model of the hypotesized language in the verification task. The model GMM is trained using SVM distances from the target language, and if these distances are represented as a vector, it is reasonable to assume that the vectors will position themselves close to eachother in a vector space. Therefore, this GMM is modeled with only one mixture component. Since the anti-model is built using SVM distances from all other languages, it is likely to assume that a GMM with 11 mixture components will fit the data well. To verify this hypothesis, a series of experiments were executed, where the number of mixture components for the anti-model was varied. The GMMs are built using the SciKit-Learn library [9], which uses the EM-algorithm to fit the statistical models.

As seen in figure 4.2, the initial hypothesis of 11 components was not a bad choice. The EER varies from 7.00% to 7.45% when using 4 and 7 components, respectively. When increasing the number of components to 12 the EER

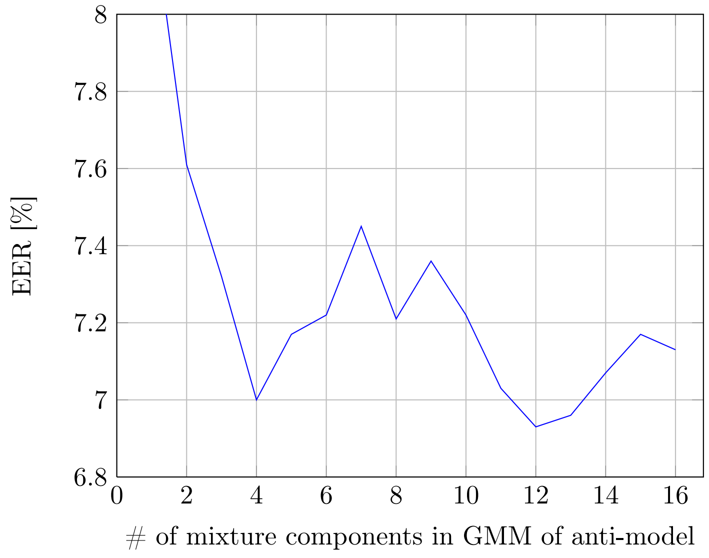

Figure 4.2: Equal Error Rate (EER) of the NIST 2003 task evaluation for different number of mixture components in the anti-model's GMM.

reaches a low point of 6.93%. Although the use of 11 components didn't give the best score, the initial hypothesis will be kept, and the remaining experiments will be executed using 11 components in the GMM of the anti-model.

4.2.4 Language Recognition Evaluation

After investigating the effects of different parameters in the back-end, it is time to evaluate the full LRE system. 200 singular values are retained and the classifiers use a RBF kernel. Figure 4.3 shows the Detection Error Trade-off (DET) curves for the NIST 2003 task when using the UAR transcriptions for manner and place seperately, and when merging the document vectors. The equal error rate is reduced to 6.73% when both manner and place transcriptions are used.

The DET-curves for each individual language can be seen in figure 4.4, and it is quite evident that there are some differences between the languages in terms of performance. German, Tamil and Vietnamese are the three languages

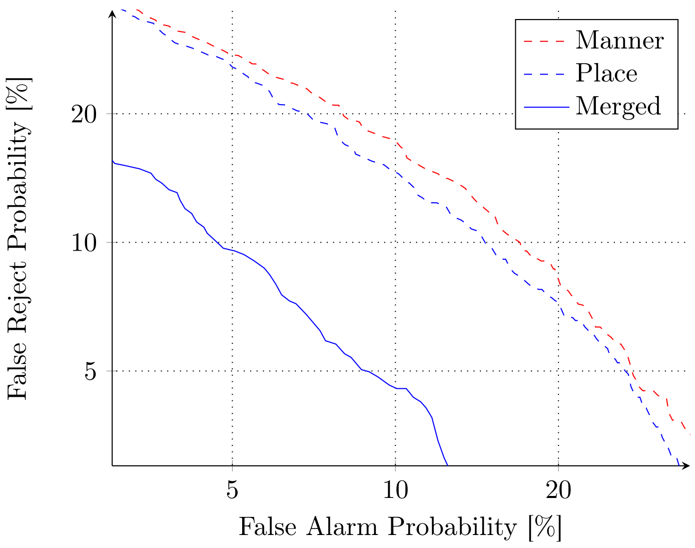

Figure 4.3: DET-curves for the LRE system when using manner and place transcription seperatly (dashed lines), and when merging the document vectors.

with the lowest EER, while Spanish and Hindi have the highest EER. The EERs for each language can also be seen in table 4.3. The performance on the evaluation set of CallFriend has also been investigated (see table 4.4).

Table 4.3: Equal Error Rate (EER) for each language in the NIST 2003 task evaluation.

Language	Arabic	English	Farsi	French	German	Hindi
EER [%]	6.25	5.89	8.75	4.91	2.81	11.25
Language	Japan.	Korean	Mand.	Spanish	Tamil	Viet.
EER [%]	8.15	7.50	8.75	8.88	3.75	3.88

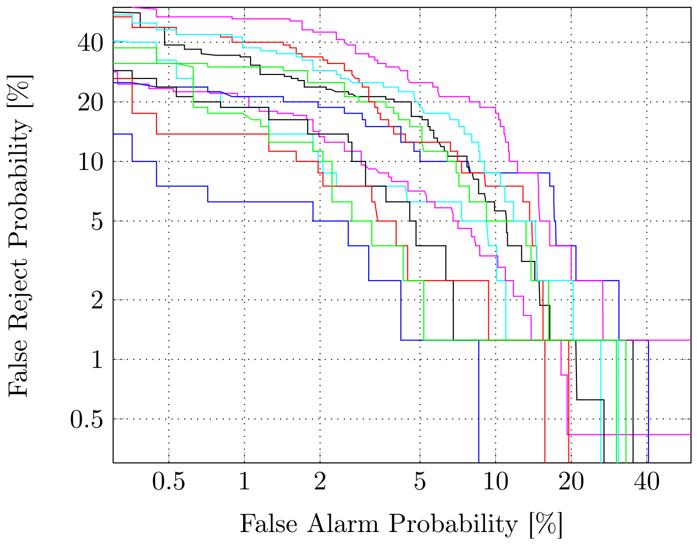

Figure 4.4: DET-curves for each individual language in the NIST 2003 evaluation.

Table 4.4: Evaluation results of the NIST 2003 task and CallFriend evaluation set.

Test set	**EER**
NIST 2003	6.73 %
CallFriend Evaluation	5.13 %

4.2.5 Effect of Out-of-Set Languages in the Evaluation

The LRE system has been trained on a set of 12 different languages, which is a fairly limited number, so it's very likely to encounter a language which is unknown to the system. The NIST 2003 task has included 80 test segments of Russian to evaluate the robustness of the system regarding so-called out-of-set languages. Table 4.5 shows the evaluation results, in terms of rejecting the Russian segments, when they were claimed to be another language. It is worth noting that most of the Indo-European languages have signifigantly lower rejection rate compared to other languages like Arabic, Japanese and Mandarin.

This could be explained by the fact that Russian is also an Indo-European language and therefore harder to discriminate. The average rejection error is quite prominent compared to the EER of the in-set languages. A universal background model was built in an attempt of improving the error rate for out-of-set languages. This was simply done by modeling a GMM using all available SVM distances in the training, and reject all test files that was below a certain threshold when evaluating it. This resulted in a better rejection rate for the out-of-set language. However, it also resulted in a much higher rate of false rejections which lead to worse EER for the overall system.

Table 4.5: Percent of the russian utterances that was rejected when it was claimed to be one of the trained languages in the system.

Claimed	Arabic	English	Farsi	French	German	Hindi
Rejected [%]	81.25	63.75	58.75	21.25	32.50	27.50

Claimed	Japan.	Korean	Mand.	Spanish	Tamil	Viet.
Rejected [%]	80.00	73.75	85.00	21.25	52.5	51.25

Average Accuracy [%]	54.06

4.2.6 Language Identification

Finally, a language identification task have been performed on the NIST 2003 set and the CallFriend evaluation set. It is quite clear that the identification task is a much more difficult task than verification. Almost 1 in 5 (80.25%) test files are identified wrongly for the NIST set, but the results from the CallFriend evaluation set is slightly better.

Table 4.6: LID results of the NIST 2003 task and CallFriend evaluation set.

Test set	**Correct**
NIST 2003	80.25 %
CallFriend Evaluation	85.39 %

4.3 Further Discussion

Recognizing a language based on a recording of a telephone conversation can be a bit challenging, and there are several reasons:

- First of all, a telephone conversation tends to be very unformal, and therefore, sloppy pronounciation is quite common. Laughter can also occur frequently in a conversation between friends.
- Between sentences, a speaker may utter sounds without any lingustic meaning, like *hmm*- and *ehh*-sounds.
- It's not unusual that a speaker produces a lot of unwanted noises, such as coughs and smacking of the lips.
- In addition, the speech of a telephone conversation is degraded and corrupted due to noise in the tranciever or on the channel.

All of these are factors that will cause problems for the front-end tokenizer and the vector space model, and consequently reduce the accuracy of the recognizer.

The accuracy of the tokenizer is probably the most limiting factor in the LRE system in terms of performance. It was shown in [8], that there is a evident correlation between the error rate of the LRE system and the error rate of the tokenizer. According to [10], which used the same attribute sets in the UARs as in this thesis, it was shown that the system achived 100% correct recognition if there were no tokenization errors. This was done by using provided reference documents, since the UARs are not error free, but it proves that the selected set of attributes is sufficient for performing language recognition.

The selection of a kernel function in the classifiers did not prove to be very crucial. The improvement in performance was not very significant when choosing a non-linear kernel, compared to the use of a linear kernel. This is probably due to the fact that the system uses 200 singular values, which means that the data sent to the classifiers are represented as 200-dimensional vectors. In high dimensional vector spaces, the linear separability improves [4] and therefore, the linear kernel achieves an acceptable separation. A known problem with non-linear kernels is the possibility of overfitting the classifier during training. This could result in a smaller margin and/or simply a non-optimal decision boundary.

When investigating what number of mixture components should be chosen for the anti-model GMM, 12 mixture components appeared to give the best result. This coincide well with tests performed while implementing the system, where a Bayesian Information Criterion (BIC) was calculated for the statistical model with respect to the training data [6]. These test showed that, for all trained languages in the system, the use of one mixture component for the model and 11 mixture components for the anti-model fitted the training data best.

The closed set of target languages consisted of 12 languages, which is a fairly small amount compared to the large number of languages that excist in the world. The place of origin for these 12 languages are spread over a large geographical area, and in addition, they appear to sound very different from one another. It would be reasonable to assume that the system would perform worse if it had to differ between languages that have a similiar phonotactic content (say Swedish and Norwegian).

Chapter 5

Conclusion

The two UARs used in this thesis has proven to work well as front-ends in a language recognition system, although the number of attributes are relatively limited compared to other tokenizers. Since the number of attributes are restricted to 12 attributes for both the UMR and the UPR, document terms can be modeled as quad-grams without making the system too computational complex. The accuracy of the front-end tokenizers have a great impact on the overall performance of a language recognition system, but the results from the NIST 2003 task are satisfying. When combining the document vectors generated from the UMR and UPR transcriptions, the system's EER was reduced to 6.73%. In order to improve the dimentionality and the sparsity problem of the document vectors, LSA was used to convert the document vectors into a concept vector space of just 200 dimensions. The use of a non-linear kernel in the SVM classifiers proved to have little profit, because the absolute improvement of using a RBF kernel compared to a linear kernel was just 0.3%, which is not very significant. By investigating the recognition results of the Russian test in the NIST 2003 task it is quite evident that out-of-set languages could be very problematic. Only 54.06% of the Russian utterances were correctly rejected, so there's a lot of room for improvement for this type of problems.

Bibliography

[1] Jerome R. Bellegarda. Exploiting Latent Semantic Information in Statistical Language Modeling. 2000.

[2] Jacob Benesty, M.M. Sandhi, and Yiteng Huang. *Springer Handbook of Speech Processing* . Springer, 2008.

[3] Chang, Chih-Chung, Lin, and Chih-Jen. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[4] Robert Clarke, Habtom W. Ressom, Antai Wang, Jianhua Xuan, Minetta C. Liu, Edmund A. Gehan, and Yue Wang. The Properties of High-Dimensional Data Spaces: Implications For Exploring Gene and Protein Expression Data. 2008.

[5] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[6] Ujjwal Das Gupta, Vinay Menon, and Uday Babbar. Parameter Selection for EM Clustering Using Information Criterion and PDDP. 2010.

[7] Alvin F. Martin and Mark A. Przybocki. NIST 2203 Language Recognition Evaluation. 2003.

[8] Pavel Matějka, Petr Schwarz, Jan *Č*ernock*ý*, and Pavel Chytil. Phonotactic Language Identification using High Quality Phoneme Recognition, PhD Thesis. 2005.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Pas-

sos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Sabato Marco Siniscalch, Jeremy Reed, Torbjørn Svendsen, and Chin-Hui Lee. Universal Attribute Characterization of Spoken Languages for Automatic Spoken Language Recognition. 2012.

[11] Hongbin Suo, Ming Li, Ping Lu, , and Yonghong Yan. Using SVM as Back-End Classifier for Language Identification. 2008.

[12] Marc A. Zissmann. Comparison of Four Approaches to Automatic Language Identification of Telephone Speech . 1996.