# Low-resource Language Identification From Speech Using X-vectors

*Paul Moore*

**MInf Project (Part 1) Report**
Master of Informatics
School of Informatics
University of Edinburgh

2019

# Abstract

Language identification from speech is a very challenging problem to tackle. For this project, the powerful new X-vector approach is taken, and found to achieve very reasonable performance on the GlobalPhone corpus.

The particular applications of X-vectors for low-resource languages are then considered. In the experiments carried out, it is first discovered that the X-vector method can still work well with just a few hours of data per language. Secondly, the possibilities of adapting a low-resource language to an existing model are explored, achieving mixed results. Lastly, different data augmentation strategies are tried, and some are found to be quite effective in improving the performance of languages with limited data.

# Declaration

## Acknowledgements

Steve Renals, my supervisor, for all his help, and support. It was a real privilege to work on this project that he suggested; he has taught me a great deal about speech & language processing and machine learning over this past year.

Samuel Sućik, who collaborated with me in building the baseline system. The work he did was really helpful, and the project would have been far harder to accomplish without his efforts.

My family and friends, for their love and encouragement regardless of whether circumstances have been easy or hard.

My Lord and Saviour, Jesus Christ, to whom I ultimately owe everything.

# Table of Contents

# Chapter 1

# Introduction

"τοσαυτα ει τυχοι γενη φωνων εισιν εν κοσμω και ουδεν αφωνον ·
εαν ουν μη ειδω την δυναμιν της φωνης εσομαι τω λαλουντι βαβαρος
και ο λαλων εν εμοι βαβαρος"*

## 1.1 Motivation

Language identification (LID) from speech is an important problem to solve in this present age of ever-increasing connectivity. As people around the world interact with one another, online or through travelling, there are many cases when being able to know what language is being spoken is useful.

For example, while the task of translating to/from different languages is itself quite different to LID, the translated languages need to be known in advance, otherwise the task is unfeasible. Google Translate [Google, 2019] currently offers the option to translate conversations as they are spoken, but this feature can only work if it is told what the languages being spoken are.

Another example of the need for LID is in automatically generating captions for online videos. While often imperfect, the performance is far more reliable if the video's language is known beforehand. There are numerous examples where captions have clearly been generated using the wrong language, and the results are almost unintelligible.

There are also occasions where LID could be critically important. For instance, what if someone calls emergency services, but can only speak a foreign language? In such a case, being able to direct the call to someone who knows that language could keep them alive. A less dire, but related scenario would be the same issue when calling customer support for a company. It is much better for public relations to ensure that callers are able speak to someone who can converse with them in their own language.

Automatic LID systems can help to address all these issues and more, whether it is to satisfy fairly trivial curiosity (what language *is* that person speaking?) or to save

*1 Corinthians 14:10-11 (SBLGNT) Accents removed; used with permission someone's life. Thus, it is very worthwhile exploring how best to achieve it.

Furthermore, there is not a great deal of consideration for identifying low-resourced languages in much of the current research. However, the LID problem still matters, particularly in linguistically diverse countries such as Papua New Guinea, for most of the reasons discussed above (health, translation, customer service). An LID system which does not require copious amounts of data would be extremely useful in these locations. It could greatly benefit those in the population who can only speak their own local language, but not the national one(s).

Therefore in this project, the goals were to not just build an LID system using recently developed methods, but also to see how it could potentially be useful for low-resourced languages.

## 1.2 Project outline

First of all, the research into LID, both new and old is discussed, especially the new X-vector approach (the technique used in this project). The following chapter covers how the baseline LID model was constructed, and analyses its performance.

The main focus of the project was on the three low-resource LID experiments carried out. The first was to see how much data was required when using X-vectors, by reducing the amount available. The second was to examine how effective language adaptation strategies were. That is, would a model trained on a certain set of languages learn features that could be used to identify languages not in that set? The third and final experiment investigated the usefulness of different data augmentation methods for improving the model's performance.

To conclude, a brief summary of the overall findings are given, and potential future work for the following year is detailed.

## 1.3 Contributions

- Built a baseline LID system for the GlobalPhone corpus. This part was done jointly with my colleague Samuel Sućik; the rest was done independently.
- Investigated using limited amounts of data for training/enrollment.
- Explored language adaptation to existing models.
- Tried various data augmentation strategies to improve the performance.

# Chapter 2

# Language Identification

In this chapter a survey of the existing work in the area of LID will be conducted. The first section outlines most of the traditional and current strategies and methods for building LID systems. This provides a context for all the research conducted in this thesis. The second section explains the recently-developed theory and practice of using X-vectors for LID. The specific application of LID to low-resource languages, and how X-vectors can be utilised for this, is deferred to section 4.1.

## 2.1 Fundamentals

### 2.1.1 Written language identification

The principles behind written LID are briefly reviewed, to contrast with spoken LID.

There are multiple challenges to deal with in written LID. For instance, some cognates are written in the same way in multiple languages (e.g. "information" could be French or English). Errors in spelling and/or grammar, use of slang (e.g. "hunky-dory"), or abbreviations (e.g "m8" instead of "mate") may also pose difficulties.

However, as early as 1974, character statistics could be used to distinguish between English and Spanish [Rau]. The features used in modern written LID research vary from that low character-level to word-level [Babu, 2010], [Adouane and Dobnik, 2017], everything in-between such as morphemes [Marcadet et al., 2005], [Barbaresi, 2016], and often some combination of these.

The written LID models are typically trained on statistics for these features, such as frequencies, or n-grams. The models themselves can vary from relatively simple decision trees [Moodley, 2016] to deep neural networks [Li et al., 2018].

Many systems are able to achieve high accuracies for written LID. For example, Choong et al. [2009] achieved accuracies above 99% on 63 languages using n-gram techniques. There are online systems too, for example Rosette [2014] claimed to get accuracies $>$ 75% per language (out of 25 languages) with as little as 1-4 words in the text.

In this report however, these will not be explored in detail, since the primary focus is spoken language. Jauhiainen et al. [2018] however provide an excellent survey of many of the latest techniques for written LID.

### 2.1.2 The challenges of spoken language identification

As shown in the previous section, written LID is relatively easy to perform using modern methods. Services such as Google Translate [Google, 2019] are able to detect languages for text, with reasonable accuracy. However, they cannot do so for speech. Why is this?

The explanation is summarised well by Muthusamya and Spitz [1997] in their short survey on LID methods:

> ...text does not exhibit the variability associated with speech (e.g., speech habits, speaker emotions, mispronunciations, dialects, channel differences, etc.) that contributes to the problems in speech recognition and spoken language ID.

Additionally, in a speech signal, there is phonological ambiguity ("a nice lolly" vs "an ice lolly"); if words are used as features, this could be an issue. Cognates may also pose a problem since they may sound similar depending on accent e.g. "qualité" (French) vs "quality" (English). One mildly interesting side note is that there may be one universal word which requires no translation: "huh" [Dingemanse et al., 2013].

The challenge posed by spoken LID is similar to that for speaker recognition. This intuitively makes sense: if a model can distinguish between speakers, it is quite likely that such a model could be adapted to distinguish between languages, if each language is treated like a different speaker.

The main difference though, is that each language will have a wide variety of speakers, where there is not only variability between speakers, but variability within individual speakers and channels as quoted earlier. The aim then is to build a model which can in effect ignore these variations, and extract only the language-specific aspects of the speech.

Thus, LID contrasts to speaker recognition, where most systems in this field tend to be focused on the task of speaker verification e.g. [Salehghaffari, 2018]. This is slightly different to identification, since instead of identifying a speaker out of a selection of possible speakers, the goal is to identify whether or not a certain person is speaking. Speaker identification (SID) systems may be more useful for instance, with speaker diarisation (identifying different speakers in an audio segment), e.g. [Zhang et al., 2018].

However, many of the principles and techniques used in all of these systems (for spoken languages and speakers) are similar. Thus, research in any of these areas is often relevant to all of them.

### 2.1.3 General principles for language identification

Regardless of the particular features and models used, there is one key equation that needs to be solved:

$$\hat{L} = \underset{i}{\operatorname{argmax}} P(L_i|O) \tag{2.1}$$

$\hat{L}$ is the predicted language. $O$ is the observation, consisting of the sequence of features which are being used in prediction. $L_i$ is language $i$ out of all possible languages the system has been trained on.

There are some discriminative training methods whereby $P(L_i|O)$ can be modelled directly. For instance, neural networks, which are discussed more in section 2.1.9 allow for this kind of approach. However, the more traditional techniques which use generative models will be covered first.

By Bayes' Rule:

$$P(L|O) = \frac{P(O|L)P(L)}{P(O)} \tag{2.2}$$

Since we can ignore P(O), as the prior probability of an observation can be regarded as virtually the same for every utterance (there are an infinite number of possible sentences), this simplifies to

$$P(L|O) \propto P(O|L)P(L) \tag{2.3}$$

Overall, the final equation used is:

$$\hat{L} = \underset{i}{\operatorname{argmax}} [P(O|L_i)P(L_i)] \tag{2.4}$$

For any LID system then, there must be some way of identifying observational features sentences or utterances, to find $O$. With regard to spoken LID, the features used depend on the level of abstraction of the utterance (section 2.1.4).

LID systems which use a generative approach then require a way to work out the likelihood of $O$, given a certain language. This is where the model training comes in - to learn characteristics of each language and use that to classify test recordings.

The prior probability of a language $P(L_i)$ depends on the context in which an LID system is being used. For instance, it would be different for Russian, depending on whether the languages being identified were only Slavic, or included ones from the Italic family as well.

In the following sections, the methodology for finding $P(O|L)$ will be elaborated upon further, with regard to both the features and the generative modelling.

### 2.1.4 Abstraction levels

One of the first issues to resolve when making an LID system is to decide what features will be used. These are largely determined by the abstraction level used to analyse the utterance.

Li et al. [May 2013] identify 5 abstraction levels at which people identify which language is being spoken, shown in figure 2.1:

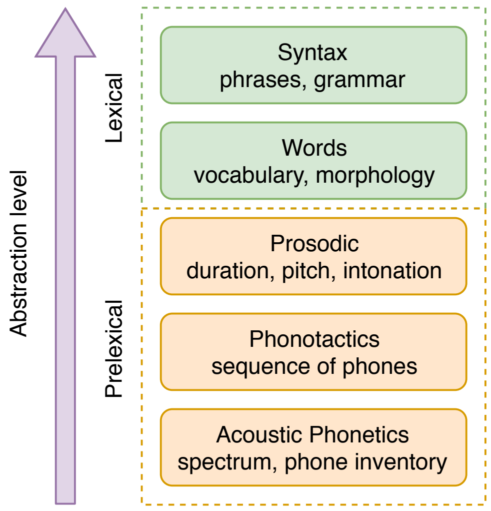


Figure 2.1: Different abstraction levels used in LID (slightly adapted from Li et al. [May 2013])

Lexical levels are not commonly used, since if words/syntax can already be identified correctly using a large vocabulary continuous speech recognition (LVCSR) system then a separate language identifier seems redundant. That is, suppose the system were, for example, being used for transcribing videos, where knowing the language is necessary for any kind of accurate transcription. If the words are already recognised (since the model operates at the word level), why bother with the separate step in-between?

However, such lexical systems can be useful for the more restricted task of language verification e.g. is the language being spoken English or not? This is relatively straightforward - it can be seen how likely an utterance is to have occurred when the system tries to analyse it. If the utterance is from another language, its probability should be very low, and vice versa, thus providing quick verification.

The prosodic level does not have widespread use in LID either, as it has been observed to be less informative than the lower levels; see [Hazen and Zue, 1997] and [Tong et al., 2006]. While features from this level may be useful as an additional features to learn from, they are not suitable for a standalone system.

However, the lower two levels (phonotactics and acoustic-phonetics) have considerably more widespread use, so will be covered in greater detail in the following sections.

### 2.1.5 Phonotactic level methods

At the phonotactic level, it is the sequence of sounds which matters. Different languages use different sequences, so if these can be modelled well, it can be a good way to identify the language.

Firstly, the phones need to be recognised; this can be done using methods like Gaussian Mixture Model (GMM) tokenisation [Torres-Carrasquillo et al., 2002] or deep learning [Tóth, 2015]. Once a sequence of phones has been found, $n$-gram models for each language are used to determine the most likely language the sequence belongs to.

Another technique used in phonotactic methods is to replace the phone sequence with a bag-of-sounds (BOS) model and simply use the counts of each phone: this removes the sequential aspect. Then, vector-space modelling (VSM) is done on these statistics for training/classification; a common way to do this is with support vector machines (SVMs).

These have been found to have some success in LID. Li and Ma [2005] was able to obtain error rates as low as 14.9% on 12 languages using the BOS technique. Tong et al. [2009] built phone tokenisers which were adapted to each language, and found based on universal phone recognisers. The equal error rates were very low, at only 2.73% for 14 languages.

However, the main drawback of phonotactic methods is that in order to train the models, transcribed speech is often used, down to the phone level. This is time-consuming to generate, though there do exist publically available corpora with transcripts e.g. GlobalPhone [Schultz, 2002].

Transcriptions may also be an issue when working with languages that do not use a Latin alphabet, and of course is impossible if the language has no writing system at all. Eberhard et al. [2019] estimate that of the 7,111 living languages in the world, 3,116 are probably unwritten.

There do exist universal phone recongisers [Khler, 2001], so it is still possible to build a phonotactic model for an LID system e.g. [Kwan and Hirose, 1997]. The main issue with doing this, is that such systems are naturally very dependent on the phone recognisers to be accurate if there are no transcriptions available. Instead, it may be better to focus on utilising the acoustic-phonetic level, which makes up the phones themselves.

### 2.1.6 Acoustic-phonetic level features

The bottom level in Figure 2.1, “acoustic phonetics”, is the level utilised in the experiments in this thesis. The main idea is to find an acoustic-phonetic distribution for a language, which can be used to build a model for each language. The exact nature of this distribution depends heavily on the chosen method of feature extraction.

A typical method, which is used in many speech systems is to extract Mel-Frequency Cepstral Coefficients (MFCCs). Zheng et al. [2001] explain how they are calculated while investigating implementation methods. In brief, MFCCs give decorrelated acoustic features for frame, which makes them useful for using in machine learning techniques which require uncorrelated inputs.

Another possibility is to get Linear Prediction [Cepstrum] Coefficients (LPC/LPCCs) [Atal and Hanauer, 1971] for the signal. These are based on the vocal tract and find the power spectrum of the input audio signal. Wong and Sridharan [2001], used LPCCs for LID, obtaining accuracies of up to 60% on 10 languages.

Thirdly, Perceptual Linear Predictive (PLP) features [Hermansky, 1990] are an option. PLP features change the attributes of the spectrum with the aim of modelling how the human hearing system works. Kumar et al. [2010] used these, as well as hybrid features for LID, which achieved accuracies of up to 87.5% for 10 languages.

There exist other variations of these acoustic features, but these are the most commonly used ones. Once a set (or sets) of features has been chosen to use, statistical models are trained to classify utterances as belonging to a particular language (or speaker).

The main advantage of working at this level is that the training data requires no other labels than which language it is; there is no need for a transcript when training. This makes it considerably easier to collect and use audio data. However, more care needs to be taken to ensure that the audio recordings are of reasonable quality. Without any transcript or higher-level information to serve as a guide, the model may not be representative of real-life languages if the audio data for training is poor in nature.

### 2.1.7 Acoustic-phonetic level shallow models

Having discussed what features can be used, the next question to address is how to use them to build a model. A core component of many models for LID and speaker recognition, has been the Gaussian Mixture Model (GMM), since Reynolds and Rose [1995] proposed using it for speaker identification (SID).

Let $\boldsymbol{x}$ be a D-dimensional feature vector, $M$ be the number of Gaussian components in the GMM (which itself is referred to as $\lambda$), $\mathcal{N}$ be a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

Then:

$$P(\boldsymbol{x}|\lambda) = \sum_{i=1}^{M} P(i)\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i) \tag{2.5}$$

The values of $P(i)$ are analogous to the weights associated with each Gaussian. Often the most straightforward algorithm for finding locally optimal values of $P(i)$, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ is the Expectation-Maximisation (EM) algorithm. Without going into extensive detail, the key concept behind it is as follows: generate estimated values for these parameters, and find how likely the training features are given these values (expectation). Then, use these likelihoods to readjust the parameters (maximisation), and repeat until convergence.

The main hyperparameter choice is in the value of $M$. Including more Gaussians will enable each model to capture more details of the training data. However, having too many results in susceptibility to overfitting and/or there may not be enough data when training.

GMMs are also an example of where having decorrelated features is useful; without correlation, the covariance matrices are diagonal. Consequently, it is much easier to perform EM, as it means there are considerably fewer parameters to fit to, and less training data is required. This is one reason why MFCCs are so frequently used with GMMs; they handle this limitation well.

One contrast between the way GMMs are used in LID versus speech recognition is that LID systems require less structure. Speech recognition systems often use Hidden Markov Models (HMMs), and GMMs to build phone models, which are themselves part of a larger architecture. However for LID/SID, it is often just a case of having one or a few very large GMMs, and possibly adapting these slightly.

For example, Reynolds et al. [2000] used adapted GMMs for speaker verification, having used them previously in SID [Reynolds, 1993]. They created a GMM-universal background model (UBM) by generating GMM-UBMs for subpopulations of the training data, and then pooling these together into a single large GMM-UBM. Then they adapted this model to different speakers, thus generating a slightly different GMM for each speaker. Utterances were classified using the log-likelihood ratio for each speaker model vs the UBM.

### 2.1.8 I-vectors

In this section, i-vectors and joint factor analysis (JFA) will be briefly explained. The aim of these methods was to achieve a low-dimensional representation which captured the speaker/language-specific characteristics. I-vectors in particular have dominated the field over the past decade, and are used in some state-of-the-art systems.

An earlier low-dimensional method of language/speaker modelling was JFA [Kenny et al., 2007], [Kenny, 2006]. This involved modelling the universal background model and the speaker/session variability all separately, and combining them to make supervectors which could model speakers.

The JFA approach is described briefly by Dehak et al. [2011]:

> Specifically, the speaker-dependent supervector is defined as
>
> $$M = m + Vy + Ux + Dz \tag{2.6}$$
>
> where $m$ is a speaker- and session-independent supervector [generally from a universal background model (UBM)], $V$ and $D$ define a speaker subspace (eigenvoice matrix and diagonal residual, respectively), and $U$ defines a session subspace (eigenchannel matrix). The vectors $x$, $y$, and $z$ are the speaker and session-dependent factors in their respective subspaces and each is assumed to be a random variable with a normal distribution $N(0,I)$.

However, i-vectors, which were originally made for speaker verification by Dehak et al. [2011], have an advantage over JFA. Rather than try to find all of these separate models, which require more data and computational time to train, a single model is used for both speaker and channel factors. This was a consequence of how Dehak [2009] found in his PhD thesis that the supervector which was meant to model the channel also tended to model the speaker's features.

Thus, in the i-vector approach, a speaker's identifying supervector is instead characterised more simply as:

$$M = m + Tw \tag{2.7}$$

$M$ is the supervector, representing an utterance by the speaker. $m$ is the universal background model (UBM) supervector. This represents the "speaker-independent distribution of features" [Reynolds et al., 2000], i.e. a background model which is not affected by the speaker or channel. $T$ is the total variability matrix, meant to model both variability due to the speaker and the channel. Lastly, $w$ is what is termed the "i-vector", meant to represent the original utterance, with the "noise" filtered out.

The main advantage of such a representation is that there are considerably fewer variables to train than in JFA, due to using only one variability matrix. This makes i-vector models easier to train than JFA ones, and they have been used for the task of LID before [González et al., 2011].

The precise mechanics of learning i-vectors are quite varied and are not particularly relevant to this project, so will not be covered in detail. Suffice to say that developing Probabilistic Linear Discriminant Analysis (PLDA) models for the i-vectors and scoring on the basis of log-likelihoods is the technique used in the current best i-vector models [Renals, 2019].

Other recent work on i-vectors replaced Gaussian Mixture Models (GMMs) for the UBM with Deep Neural Networks (DNNs). The network was developed to extract features from the training data. These features were subsequently used to train a separate GMM for speaker recognition [Lei et al., 2014]. They found that this was able to reduce the Equal Error Rate (EER) by 30% relative to the traditional EM method for finding the UBM.

### 2.1.9 Time delay neural networks (TDNNs)

In this section, familiarity with the basics of DNNs is assumed, though if that is not the case then Nielsen [2015] provides a good introduction.

TDNNs are a neural network architecture designed to take context over an event in time. Originally developed by Waibel et al. [1989], the structure allows for modelling of context (so multiple frames are taken into account). This matters for LID, since the context of frames matters for identifying utterance-level features.

Additionally, TDNNs are able to model shift invariance i.e. the specific timing of a learned feature within an utterance does not affect the output. Again, this is important for LID since language features should be recognisable regardless of where exactly in time they occur in the utterance.

Context is obtained by having the inputs to a layer come from a window over the layer below, for instance, the three frames before and after a certain frame, as illustrated in Figure 2.2

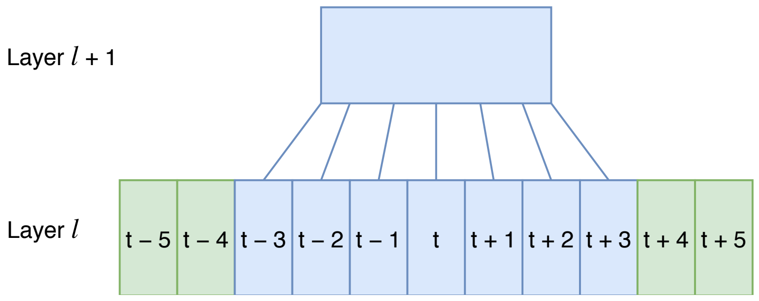


Figure 2.2: Example of context modelling over 3 frames in a TDNN

To get shift invariance, regular forward/backward propagation is done to get the error derivatives for a given TDNN layer. However, when updating the weights, corresponding connection weights (e.g. all the initial frames for each window), are averaged. Consequently, if a feature occurs, it does not matter where exactly it does, since the network is trained to look for it anywhere.

Due to their usefulness, TDNNs have been used in various speech system neural networks. Peddinti et al. [2015] used them for developing a LCVSR system on the Switchboard corpus, and found them to be superior to simpler DNN models. Here they also used sub-sampling i.e. rather than have fully connected layers, the connections grew more sparse at the higher layers. This was found to make training about 5 times faster (though still slower than a simple DNN), and made the model smaller.

While TDNN layers were used in this experiment, sub-sampling was not, since the Kaldi framework [Povey et al., 2011] (which formed the basis of much of the acoustic modelling and training) does not currently provide this option.

## 2.2 X-vectors

X-vectors were a fundamental component of the language recognition experiments. This section will outline the previous research into them, and detail how exactly they are intended to work.

### 2.2.1 Previous usage

X-vectors are a fairly new technique, developed by Snyder et al. [2017]. The key idea in their first system was to extract embeddings from a DNN, for purposes of speaker verification.

Their goal was to ”produce embeddings that generalize well to speakers that have not been seen in the training data.” [Snyder et al., 2017]. That is, rather than train separate i-vectors for each speaker, which is costly and time consuming, learn embeddings for each utterance, which can be used to distinguish between any speakers.

This was later expanded on for speaker recognition [Snyder et al., 2018b] - being able to identify a particular speaker from a selection of possible ones, and then adapted to work with language recognition [Snyder et al., 2018a], where it achieved very reasonable results.

In practice, X-vector systems were found to generally be superior to i-vector based ones, when data augmentation was used [Snyder et al., 2018b] and [Snyder et al., 2018a].

X-vectors also bear a resemblance to d-vectors [Variani et al., 2014]. D-vectors are embeddings extracted from the last hidden layer in a network. However, the network structures are different. The d-vector network is simpler, with fully connected hidden layers, but without time-based structures like TDNNs, or RNNs. Another difference is that there is no statistics pooling with d-vectors, so the network embeddings are based on frame-level information, rather than utterance-level.

There do not seem to have been any direct comparison studies between X-vectors and d-vectors. However, it seems likely that X-vectors are better, since d-vectors were generally observed to be worse than i-vectors [Variani et al., 2014], except for when the false rejection probability was greatly reduced.

### 2.2.2 Getting X-vectors

To learn the embeddings, the following rough network structure is used, shown in figure 2.3.

The initial levels operate at the frame level. These are TDNN layers, which take an increasingly wider context of frames as the layers go up, followed by one or two extra TDNN layers with no additional context.

The input features for each frame can include MFCCs, BNFs, prosodic features, etc. or a combination of these. For LID, BNFs were found to be one of the better features to train on Snyder et al. [2018b]. The feature vector for each frame $f$, is represented by $\boldsymbol{x}_f$.

After the TDNN layers, the mean and standard deviation of the final frame-level layer outputs are computed for each input vector $\in \{\boldsymbol{x}_0,\ldots,\boldsymbol{x}_T\}$. Thus, these statistics are pooled across the entire input segment, and are then used as the input for the final two affine layers, which are intended to find embeddings.

This pooling is significant, because it shifts the following network layers from operating at the level of individual frames, to the level of the whole segment. Thus the

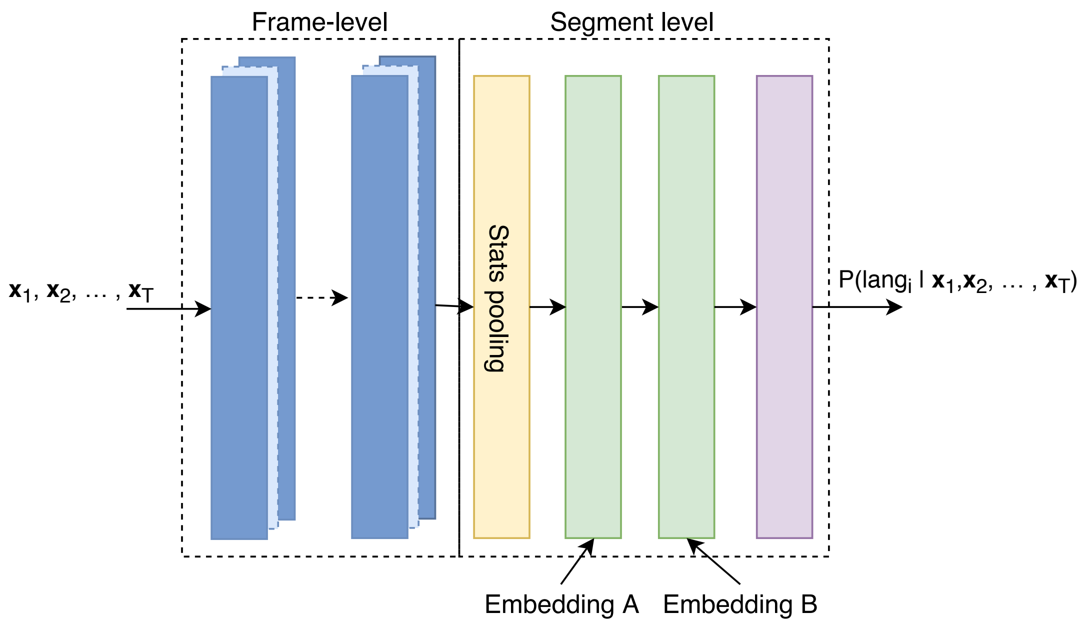

Frame-level
Segment level
$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$
Stats pooling
P(lang$_i$ | $\mathbf{x}_1,\mathbf{x}_2, \ldots, \mathbf{x}_T$)
Embedding A
Embedding B

Figure 2.3: A simple network structure for computing X-vectors. $\mathbf{x}_i$ represents a single frame's feature vector. $\mathbf{x}_1, \mathbf{x}_2, , \mathbf{x}_T$ thus represent the feature vectors for the entire segment

embedding layers are able capture language characteristics, since they are being provided features of entire utterances for different language as inputs.

A final output layer (e.g. softmax) produces predictions for each language given the input segment. Snyder et al. [2018a] found that using the embeddings to build a classifier (section 2.2.3) rather than the output of the network resulted in better performance. This may be because these layers can capture hidden features of the language which can be generalised well for classifying.

### 2.2.3 Using X-vectors

In order to perform classification, a dataset of enrollment utterances are required. These are separate to the training/evaluation/testing sets. Each utterance is propagated through the network, and the embedding output it produces is saved. These outputs are then averaged across each language to get language-level X-vectors.

After obtaining these X-vectors, a classifier is trained. This is recommended by Synder et. al Snyder et al. [2018a] to be a "discriminatively-trained Gaussian classifier". A PLDA model was used for speaker recognition Snyder et al. [2018b]. Sarma et al. [2018] used logistic regression for their i-vector model, which turned out to be suitable for use with X-vectors as well (section 3.3.3).

The $C_{primary}$ score metric described in the NIST Language Recognition Evaluation Plan [NIST, 2017], and used by Snyder et al. [2018a], is used to assess the performance of the system, rather than simply accuracy alone. This is a cost-based metric which takes the probabilities of misses and false alarms into account for each pair of languages. These are weighted by prior probabilities, and averaged to get an overall score. NIST [2017] provides more details about the calculation specifics; there is little point in going into great detail about them here since they are not very relevant.

### 2.2.4 Summary of X-vectors

In conclusion, X-vectors are a powerful deep-learning tool that have great potential for making better LID systems. They are able to capture the frame-level information, and use its context to find utterance-level features.

Additionally, X-vectors seem to be capable of outperforming some of the current state-of-the-art i-vector models. However, i-vectors are currently quite well-established, and are unlikely to go away anytime soon. I-vectors are also a shallow approach, which may make it faster/easier to build models in some cases.

Lastly, the hidden features learned for X-vectors are robust, since they are based on pooled statistics from the lower network levels; a few rogue frames in the training utterances are unlikely to have a great impact. They can address the need for the language features to be speaker invariant speaker too, provided the network is trained on numerous speakers of both genders.

# Chapter 3

# Baseline model and techniques

In this chapter, first of all a description is provided of the language dataset (GlobalPhone), and the justification for using it, since it was used in all the experiments. Next, the methods for processing the data are described, followed by the training and evaluation of the initial model. Finally, the baseline results for this model are discussed.

## 3.1 The GlobalPhone Dataset

### 3.1.1 Description

Originally published by Schultz [2002], Globalphone (sometimes referred to here as GP) consists of audio recordings where participants read various news articles in their native language. Metadata about the speakers, such as gender, age, dialect, the recording environment etc. were recorded for most of them as well. There are transcriptions for each of the news articles, since one of its original purposes was to train LVCSR systems in different languages.

Currently, there are over 1500 speakers in total, and 22 languages in the corpus, with about 20 minutes of speech per speaker. The data recordings were mostly done in quiet locations, with a good recorder/microphone, so the audio is generally of high quality.

The statistical summary for each of the languages in GP that were used in these experiments is presented in Table 3.1. Any minor deviation from the official statistics is likely a result of how the data was preprocessed (see section 3.2.1). For Polish and German, considerable amounts of the metadata was missing in the speaker files.

For most languages, the gender split in the speakers is fairly even, apart from Japanese and German which are majority male, and Thai and Turkish which are majority female. Age distribution varies considerably more, but most tend to fall within the 20-29 range. The length and number of speakers are similar, apart from Shanghainese which has far less than the others. NB: Shanghainese is referred to in GP as Chinese-Shanghai; however in this report the more common moniker is used instead.

| Language | Length (hrs) | No. speakers | Age (mean) | Gender | |
|---|---|---|---|---|---|
| | | | | (m/f) | Ratio |
| Arabic | 19.0 | 79 | 26.7±11.3 | 36/43 | 46:54 |
| Bulgarian | 21.4 | 77 | 32.6±14.1 | 32/45 | 42:58 |
| Croatian | 15.9 | 94 | 32.1±14.2* | 38/56 | 40:60 |
| Czech | 31.9 | 102 | 25.7±11.2 | 57/45 | 56:44 |
| French | 27.0 | 100 | 32.6±12.1* | 49/51 | 49:51 |
| German | 18.3 | 83 | - | 75/8 | 90:10 |
| Japanese | 33.9 | 149 | 23.1±7.6* | 104/44* | 70:30 |
| Korean | 21.0 | 100 | 25.9±5.4* | 50/50 | 50:50 |
| Mandarin | 31.1 | 132 | 23.2±5.4* | 64/68 | 48:52 |
| Polish | 24.6 | 99 | 34.1±10.9** | 12/14** | - |
| Portuguese | 26.2 | 102 | 28.6±10.0* | 54/48 | 53:47 |
| Russian | 24.7 | 115 | 27.9±10.2 | 61/54 | 53:47 |
| Shanghainese | 3.6 | 41 | 43.3±11.6 | 16/25 | 39:61 |
| Spanish | 22.1 | 100 | 27.2±11.3 | 44/56 | 44:56 |
| Swedish | 21.7 | 98 | 32.3±13.7 | 50/48 | 51:49 |
| Thai | 28.2 | 98 | 20.6±1.7 | 27/65 | 28:72 |
| Turkish | 17.1 | 100 | 27.5±10.8 | 28/72 | 28:72 |
| Vietnamese | 19.7 | 129 | 29.6±11.5 | 73/54 | 57:43 |
| Total | 381.2 | 1798 | - | - | - |
| Mean (language) | 22.4±7.0 | 99.9±22.7 | - | - | - |

Table 3.1: Summary of all GlobalPhone data as used in experiments. For mean data, the ± value is the population standard deviation. Gender ratios are rounded to whole numbers. * indicates that there are only a few (1-3) speakers without the listed information. ** indicates that there were many speakers missing it

GlobalPhone has been utilised in numerous language projects before. Ghoshal et al. [2013] used it to train multilingual DNNs, finding that multiple languages improved speech recognition. [Swietojanski et al., 2012] trained cross lingual models, investigating the benefits of having unlabeled data from different languages to enhance a speech recogniser in another language. [Vesel et al., 2012] built a multilingual DNN to obtain multilingual BNFs, which were found to be better at discriminating between languages than monolingual BNFs.

These are just a few examples; there are numerous other projects which have had GP for their experimental data. The nature of the corpus additionally allows for it to be expanded on (originally there were only 15 languages; now as mentioned earlier, there are 22). However, there can be issues that result - for example some later additions have a slightly different file structure. Another problem is that some additions also neglect to include metadata.

In the following section, the reasons for why GlobalPhone was deemed to be an appropriate corpus to use in the context of the experiments here will be explained, and any potential issues with it addressed.

### 3.1.2 Suitability analysis

One of the strengths of the GP corpus is that the data is of high quality. This makes it easier when training models since background noise, or other mistakes, are less of a problem.

This strength may also be a weakness in some cases, as the trained model may not generalise well, or function correctly in noisy environments. To limit the effects of this, the recordings can be:

- sped up or slowed down slightly - there are simple Kaldi scripts for doing this.
- reverbrated using the RIRS database [Povey, 2017]. This database is described as containing: "simulated and real room impulse responses, isotropic and point-source noises". This makes the utterances sound more like they come from a variety of rooms with different background noises.
- augmented with other music, speech and noise, e.g. from the MUSAN corpus [Snyder et al., 2015].

Modifying the data in such a way makes it more representative of real-life data. This means that when the network is trained, it is more likely to learn the important features since there is much more variability in the training utterances. As a result, the network is more robust (Table 2, [Snyder et al., 2018b]) and can generalise better to testing data.

Another potential problem with training on GP is the type of speech - it comes from reading news articles, and the speakers were allowed to repeat each recording until they got it right. This could be a problem when testing on conversational speech, where people tend to speak faster, and to make mistakes.

However, as mentioned earlier, it is possible to modifying the data (speeding up, adding background noises), Such augmentation may mitigate this problem to an extent, though the speed changes are usually recommended to be slight (0.9 or 1.1 times the original).

Furthermore, Globalphone has a very large number and diversity of speakers and articles, and most of these are input into the network for training. This is likely to force the model to learn the fundamental embeddings which are representative of the language, instead of relying too much on any individual utterance.

Lastly, high-level language features learned from read speech data are unlikely to be very different to conversational speech. English and French would still probably sound distinct whether the recordings were of people reading or conversing.

In conclusion, despite its potential flaws, the GP corpus is a suitable source of training/testing data, since these issues can be accounted for through its size, and using data augmentation techniques. Augmentation is also beneficial with regard to low-resource language applications, as is explained in section 4.5.

## 3.2 Data processing

The data processing stage scripts were based on the Kaldi scripts for working with Globalphone (GP) [Povey and Galvez, 2017].

### 3.2.1 File conversion

To begin, all the GP files were converted from the shorten (.shn) to the WAV (.wav) format. 3 of the 22 languages appeared to be broken, or at least failed to be converted using the standard GP scripts for Kaldi. In a few languages some speakers or utterances were also bad or broken, so these were omitted.

These files were saved and stored, so that this step would not be repeated. Kaldi uses .scp files to point to the location of where actual data is stored. The advantage of this is that multiple experiments can then refer to the same files without requiring extensive copying, or wasted space.

### 3.2.2 Splitting datasets

Once the conversion had occurred, the files were assigned to training, enrollment, evaluation and test sets. The training set was for 70% of the data, and would be used to train the network. The enrollment data (10%) was used for training the logistic regression classifier from the embeddings. The evaluation data (10%) was used in both training the network and finding suitable hyperparameters for classification. The test data (10%) was used to judge the general performance of the best model in the third experiment (section 4.5); it was not particularly necessary for the other two.

The Globalphone documentation [Schultz, 2002] has two recommendations when dividing up datasets:

- Each speaker should only appear in one set
- Each news article that is read should also only appear in one set, even if read by different speakers.

It was relatively straightforward to achieve some of this splitting by using the Kaldi GlobalPhone script's lists [Povey and Galvez, 2017] for development/evaluation speakers - which corresponded to our evaluation/test speakers. However these were not present for some of the languages being used, and a further problem was that there was no list for enrollment data (simply because GlobalPhone was not built for LID originally).

It was not possible to get pure data splits for the training/enrollment sets using the official criteria. The first was relatively easy to meet. Nonetheless, the second was much more difficult, since the same article would be often be read multiple times. This meant that the chances that two speakers would read the same article, but be put in different groups were quite high.

To try all the possible splittings was not feasible. As shown in Table 3.1 there were typically $\sim$100 speakers per language and $\sim$16 were used in evaluation/testing, leaving $\sim$84. To select just one training set of $\sim$70 speakers as perfectly as possible would require approximately $\binom{84}{70} = 3.17 \times 10^{15}$ splits. Instead, the splitting was done randomly, where the allowance for article overlap was gradually increased until a split was found.

This was only done once, to generate a list of speakers, which would then be used across experiments. The precise demographics of the splits are shown in the tables in the appendix. Fortunately in most cases, the ages and gender splits for each language did not deviate too far from the original complete data.

The utterances in enrollment, evaluation and testing datasets were further split into segments $< 30$s, as Snyder et al. [2018a] did. Then evaluation/testing data was also split separately into 10s and 3s segments, in order to allow for testing on shorter segments. NB: when referring to evaluation/testing length, for the sake of brevity, statements like "length of 30 seconds" mean that utterances were at most 30 seconds long, (though they may well have been shorter).

### 3.2.3 Feature extraction

While bottleneck features (BNFs) were recommended for LID [Snyder et al., 2018a], these were not readily available, so MFCCs were used instead.

The MFCC configuration is summarised in Table 3.2.

| Feature | Setting |
|---|---|
| Sampling Frequency | 16 kHz |
| Frame Length | 25 ms |
| Low Frequency | 20 Hz |
| High Frequency | 7800 Hz |
| Dimensionality | 23 |

Table 3.2: MFCC configuration

Voice Activation Detection (VAD) was used to detect which frames had speech, and which did not, based on the energy of each frame. This allowed for the removal of silence frames in the training data - they are very unlikely teach the network anything meaningful about a language!

One important expansion on the Kaldi SRE recipes Snyder et al. [2016], was to preprocess all the data in one go, creating MFCC and VAD features for every type of dataset (including the augmented ones made in section 4.5).

The reason for doing so was that many of the initial experiments were carried out in one of the University of Edinburgh's GPU clusters. Repeatedly creating the same features for each experiment separately added at least an hour to the running time. Instead, by preprocessing everything in one go, this step could effectively be skipped in all future experiments.

## 3.3 Neural network training & evaluation

### 3.3.1 Preparing training data for the network

Cepstral mean normalisation (CMN) was applied to the training data first of all. This is a common normalisation technique which gives the MFCCs zero mean to offer some protection against outliers. The variance was not reduced to 1, as was the case in the original SRE experiments [Snyder et al., 2018b].

Since CMN is known to perform poorly on short utterances, utterances of length $< 5$s were removed in the baseline experiments. The minimum length was reduced to 3s for the later experiments, in order to replicate some of the issues possibly faced when working with low-resource languages

After this, examples were generated for training the network. These were allocated as chunks of frames, with a length shorter than 4s, thus breaking up all the training utterances.

The original scripts [Snyder et al., 2016] recommended generating 50-150 archives (where the data is stored) for use in training. This required setting the target number of frames per iteration of the network manually as a hyperparameter, and checking how many archives would be produced. For the baseline experiments which were carried out on all of the training data, this was not a serious issue since the amount of training data was not being varied.

However, for the experiments with low-resource languages, the amount of training data was changed frequently, which would have meant having to repeatedly adjust that hyperparameter to ensure it was appropriate. The solution was simply to work backwards and use the number of archives as the target in order to find the number of frames per iteration. This generated approximately 100 archives each time, and was within the ideal number recommended.

### 3.3.2 Structure and training

The neural network structure used was kept the same as in the original paper [Snyder et al., 2018a], since the main focus of this project was not to build a completely new architecture, but investigate the performance of an existing one. Furthermore, it was relatively safe to assume that Snyder et al. had already done a fair amount of experimentation with the structure during their investigations, particularly early on [Snyder et al., 2017].

The architecture is illustrated in Figure 3.1 and described in Table 3.3. Both of these highlight the division between the levels which operate at the frame-level and the ones at the segment level. This distinction is important, as discussed earlier in section 2.2.3.

All the hyperparameters were left at their default settings, and most are summarised in Table 3.4. However, there are a couple of them, namely “dropout rate” and “proportional shrink” (as described in Kaldi) which require a little more explanation.

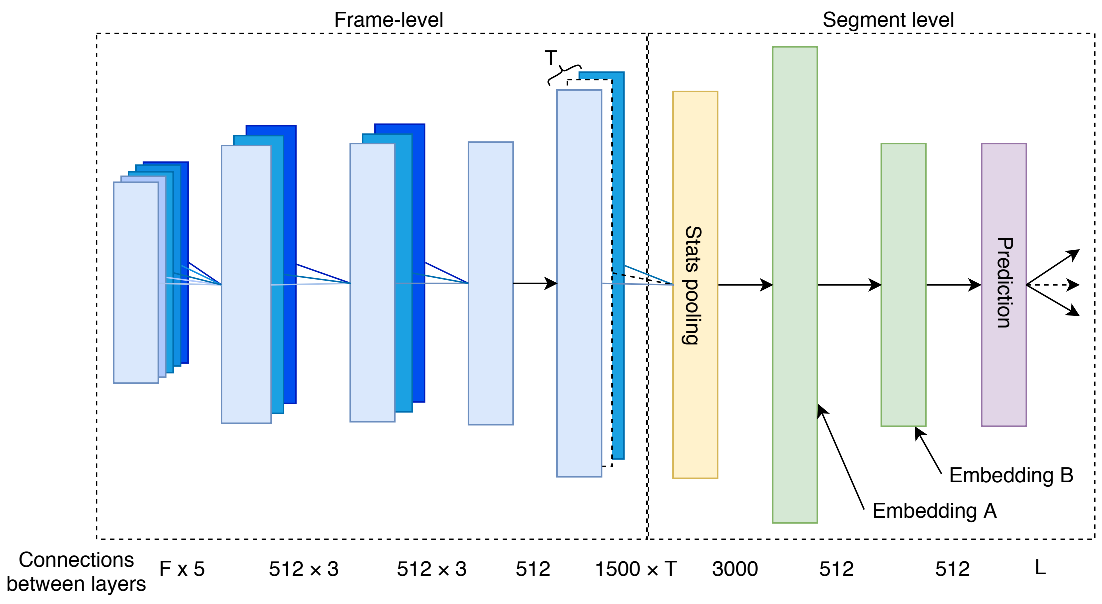


Figure 3.1: X-vector neural network structure. Note that there is only one final frame-level layer (without context) described in the architecture in Table 3.3. $T$ of these layers are used as inputs to the stats pooling layer, hence why the final frame level layer is shown as having $T$ layers.

The dropout rate in the Kaldi `nnet3` scripts depends on how much of the data the network has seen, rather than simply dropping out the same percentage of neurons each time. In this case, it was left at the setting used in speaker recognition. That is, for the first 20% of the data, the rate was 0, then it increased linearly to 0.1 until the network had seen 50%. It finally remaining at this level for the rest of the data input.

Additionally, the proportional shrink was set to 10. This is intended to help avoid the vanishing gradient problem where there is little to no changes at the extremes of the function.

| Level | Layer | Layer context | Tot. context | No. inputs | No.outputs |
|---|---|---|---|---|---|
| frame | context1 | $[t-2, t+2]$ | 5 | 5F | 512 |
| | context2 | $\{t-2, t, t+2\}$ | 9 | 1535 | 512 |
| | context3 | $\{t-3, t, t+3\}$ | 15 | 1536 | 512 |
| | hidden1 | $\{t\}$ | 15 | 512 | 512 |
| | hidden2 | $\{t\}$ | 15 | 512 | 1500 |
| segment | stats | $[0, T)$ | $T$ | $1500T$ | 3000 |
| | embedding1 | $\{0\}$ | $T$ | 3000 | 512 |
| | embedding2 | $\{0\}$ | $T$ | 512 | 512 |
| | softmax | $\{0\}$ | $T$ | 512 | $L$ |

Table 3.3: X-vector network architecture, adapted from Snyder et al. [2018a]

| Hyperparameter | Setting |
|---|---|
| Initial learning rate | 0.001 |
| Final learning rate | 0.0001 |
| Momentum | 0.5 |
| Batch size | 64 |
| No. epochs | 3 |

Table 3.4: Initial network hyperparameter settings

### 3.3.3 Evaluation

X-vectors were computed for the enrollment and evaluation data. As described in section 2.2.3, this was done by passing the utterances through the network. The enrollment X-vectors were then averaged across each language to get language-level X-vectors.

A logistic regression classifier was used to classify the X-vectors for each language, similarly to how Sarma et al. [2018] classified i-vectors extracted from a TDNN. The code was based on that from Synder and Povey [2014] who also used logistic regression as a classifier for LID.

The classifier was first balanced based on the counts of the training and testing languages (some languages like Shanghainese had fewer utterances than others). The default Kaldi scripts were then used for training and evaluating the logistic regression model.

A grid search was done to find reasonable hyperparameters for this classifier. The choices are summarised in Table 3.5.

If some of the names in Table 3.5 seem unfamiliar to "classic" logistic regression, it is because Kaldi's version is slightly different [Synder, 2014]. As is fairly standard, it uses L-BFGS to find a solution, with a specified maximum number of attempts (steps). The normaliser also behaves as expected for logistic regression.

However, the power and target mixture components hyperparameters allow for use of Gaussians to build the classifier. The target is for the final number of Gaussians for modelling all the languages, while the power affects how quickly this is increased during the steps. The minimum would be the number of languages (17), but including more allows for better fitting. Of course, having too many would result in overfitting, so 100 (roughly 5 per language) was found to be a suitable number.

| Hyperparameter | Setting |
|---|---|
| Max. steps | 200 |
| Normaliser | 0.001 |
| Power | 0.15 |
| Target mixture components | 100 |

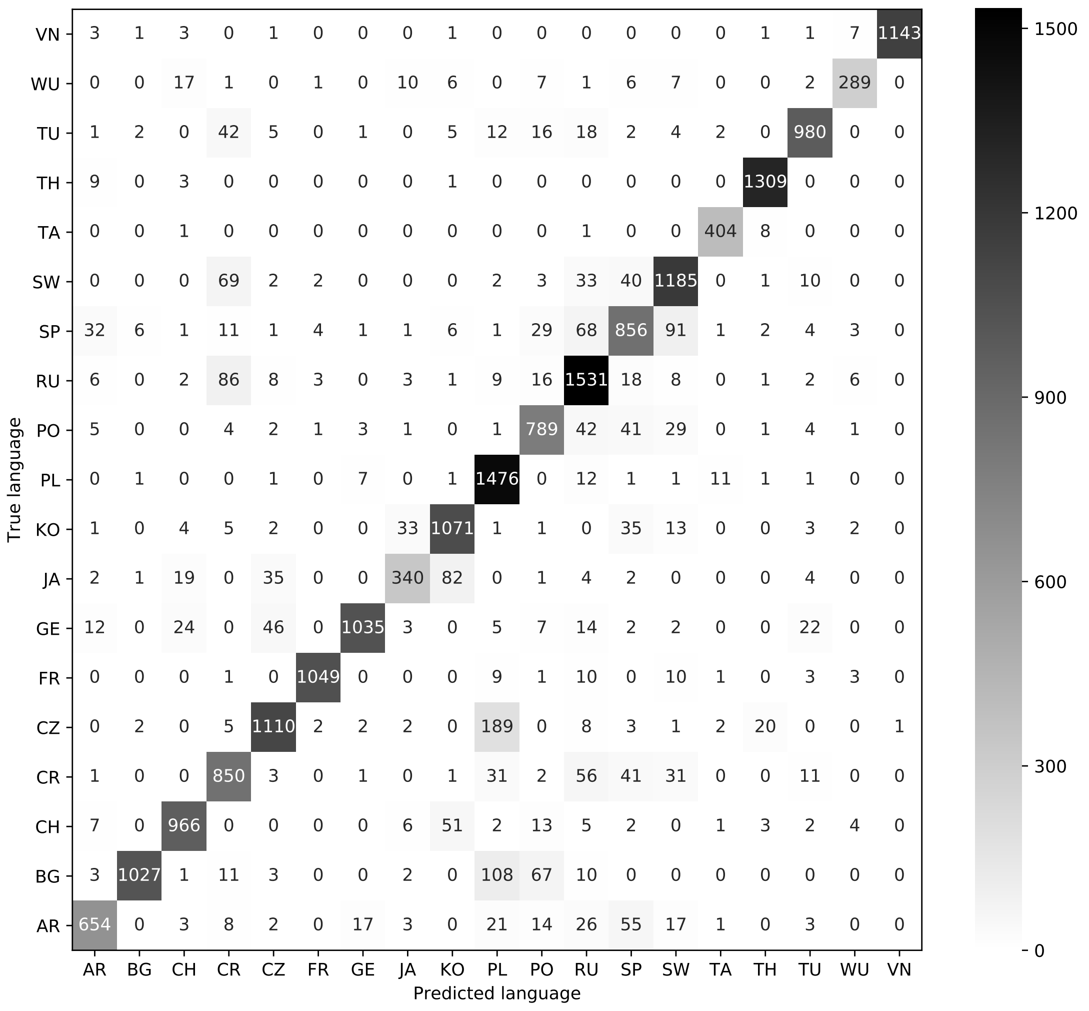
Table 3.5: Logistic regression hyperparameters

## 3.4 Results

### 3.4.1 Initial network

When evaluating the logistic regression model on utterances at most 10 seconds long, the baseline classifier obtained 88.8% accuracy, with a $C_{primary}$ score of 0.150. This was comparable to the score found by Snyder et al. [2018a] which was encouraging.

The raw confusion matrix is shown in Figure 3.2. A normalised version is shown in Figure 3.3. Normalisation was done along the axis of the true language, thus reflecting the precision of the model.



| | AR | BG | CH | CR | CZ | FR | GE | JA | KO | PL | PO | RU | SP | SW | TA | TH | TU | WU | VN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VN | 3 | 1 | 3 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 7 | 1143 |
| WU | 0 | 0 | 17 | 1 | 0 | 1 | 0 | 10 | 6 | 0 | 7 | 1 | 6 | 7 | 0 | 0 | 2 | 289 | 0 |
| TU | 1 | 2 | 0 | 42 | 5 | 0 | 1 | 0 | 5 | 12 | 16 | 18 | 2 | 4 | 2 | 0 | 980 | 0 | 0 |
| TH | 9 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1309 | 0 | 0 | 0 |
| TA | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 404 | 8 | 0 | 0 | 0 |
| SW | 0 | 0 | 0 | 69 | 2 | 2 | 0 | 0 | 0 | 2 | 3 | 33 | 40 | 1185 | 0 | 1 | 10 | 0 | 0 |
| SP | 32 | 6 | 1 | 11 | 1 | 4 | 1 | 1 | 6 | 1 | 29 | 68 | 856 | 91 | 1 | 2 | 4 | 3 | 0 |
| RU | 6 | 0 | 2 | 86 | 8 | 3 | 0 | 3 | 1 | 9 | 16 | 1531 | 18 | 8 | 0 | 1 | 2 | 6 | 0 |
| PO | 5 | 0 | 0 | 4 | 2 | 1 | 3 | 1 | 0 | 1 | 789 | 42 | 41 | 29 | 0 | 1 | 4 | 1 | 0 |
| PL | 0 | 1 | 0 | 0 | 1 | 0 | 7 | 0 | 1 | 1476 | 0 | 12 | 1 | 1 | 11 | 1 | 1 | 0 | 0 |
| KO | 1 | 0 | 4 | 5 | 2 | 0 | 0 | 33 | 1071 | 1 | 1 | 0 | 35 | 13 | 0 | 0 | 3 | 2 | 0 |
| JA | 2 | 1 | 19 | 0 | 35 | 0 | 0 | 340 | 82 | 0 | 1 | 4 | 2 | 0 | 0 | 0 | 4 | 0 | 0 |
| GE | 12 | 0 | 24 | 0 | 46 | 0 | 1035 | 3 | 0 | 5 | 7 | 14 | 2 | 2 | 0 | 0 | 22 | 0 | 0 |
| FR | 0 | 0 | 0 | 1 | 0 | 1049 | 0 | 0 | 0 | 9 | 1 | 10 | 0 | 10 | 1 | 0 | 3 | 3 | 0 |
| CZ | 0 | 2 | 0 | 5 | 1110 | 2 | 2 | 2 | 0 | 189 | 0 | 8 | 3 | 1 | 2 | 20 | 0 | 0 | 1 |
| CR | 1 | 0 | 0 | 850 | 3 | 0 | 1 | 0 | 1 | 31 | 2 | 56 | 41 | 31 | 0 | 0 | 11 | 0 | 0 |
| CH | 7 | 0 | 966 | 0 | 0 | 0 | 0 | 6 | 51 | 2 | 13 | 5 | 2 | 0 | 1 | 3 | 2 | 4 | 0 |
| BG | 3 | 1027 | 1 | 11 | 3 | 0 | 0 | 2 | 0 | 108 | 67 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AR | 654 | 0 | 3 | 8 | 2 | 0 | 17 | 3 | 0 | 21 | 14 | 26 | 55 | 17 | 1 | 0 | 3 | 0 | 0 |

Figure 3.2: Raw confusion matrix for baseline experiment, when evaluated on utterances of length 10s. Language codes are explained in Table 3.6.

Overall the performance was very reasonable. The languages which were confused with one another were for the most part expected. For instance, Japanese and Korean, or Polish and Czech (in fact, all of the Slavic languages) were frequently mistaken.

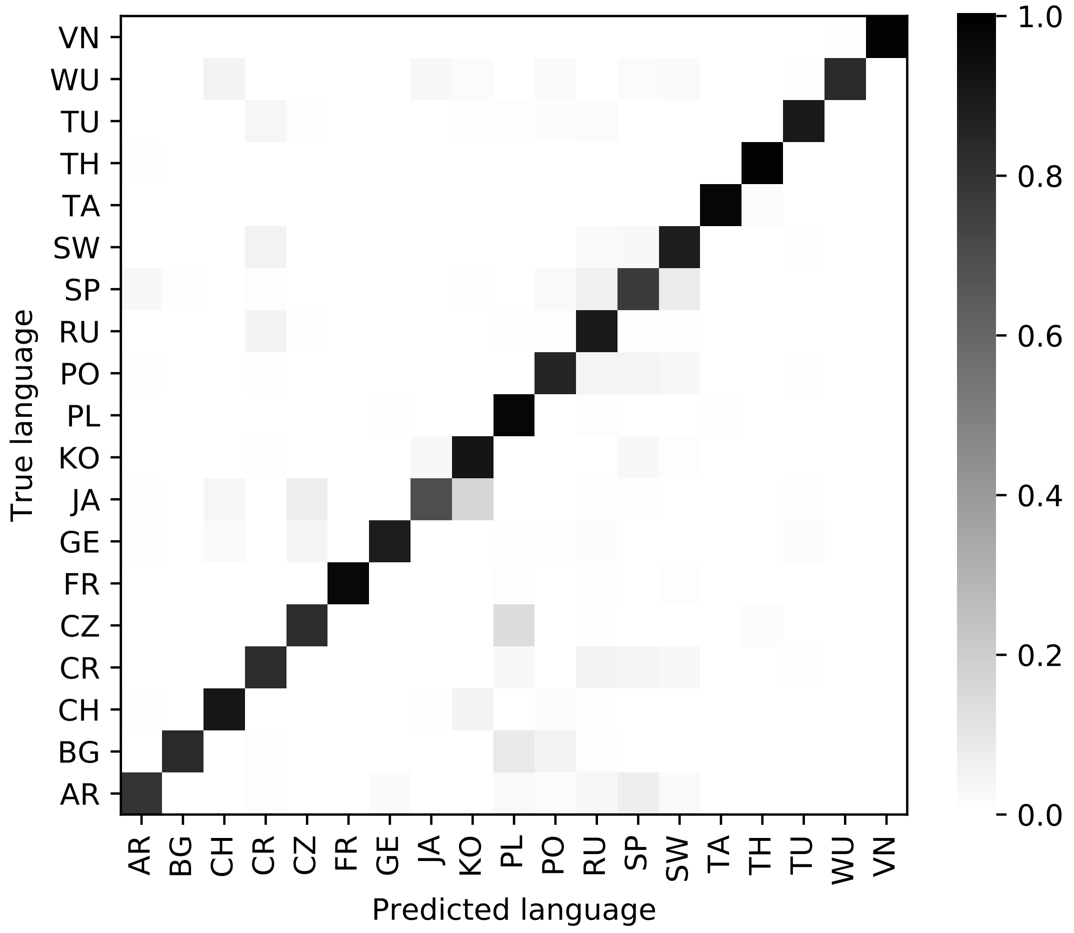


Figure 3.3: Normalised confusion matrix for Figure 3.2. Language codes are explained in Table 3.6. Numbers omitted for clarity

There were a few surprises; for example Spanish was predicted to be Swedish 91 times. Arabic was predicted to be Spanish 55 times. Even Bulgarian was predicted to be Portuguese 67 times!

| Code | Language | Code | Language | Code | Language |
|---|---|---|---|---|---|
| AR | Arabic | BG | Bulgarian | CH | Mandarin |
| CR | Croatian | CZ | Czech | FR | French |
| GE | German | JA | Japanese | KO | Korean |
| PL | Polish | PO | Portuguese | RU | Russian |
| SP | Spanish | SW | Swedish | TA | Tamil |
| TH | Thai | TU | Turkish | WU | Shanghainese |
| VN | Vietnamese | | | | |

Table 3.6: Language codes in Figure 3.2 and Note that Tamil was included in the baseline experiment, but none of the ones after that since it was found to cause numerous problems.

The precise reasons behind these are difficult to surmise. It may be the case that the languages have a comparable rhythm. Or perhaps they use the same (or similar) phones, with a fairly matched distribution. Mismatches in accent or gender could be contributing factors; for instance it was later discovered that all the Bulgarian speakers in the enrollment dataset were female, but there were mixed genders in the evaluation set.

However, regardless of these particular oddities, the overall model seemed very reasonable. Its performance was especially encouraging since there were 19 languages. After this model was made though, the number of language decreased to 18; Tamil was dropped due to it being very difficult to use in the later experiments.

### 3.4.2 Investigating the number of epochs

One final thing that was required before progressing further was to find how many epochs were necessary to train a good model. Since the initial experiments were being performed in the GPU cluster, and each additional epoch took about 3 hours (starting with 5 hours for a 3-epoch experiment), it was important to keep the number as low as possible while still attaining acceptable accuracies.

Results of the search are shown in Figure 3.4 and Figure 3.5. It was decided to use 7 epochs as standard, since after that point, $C_{primary}$ began to decline more slowly. Furthermore, 7 epochs also took slightly under a day to run on the cluster, so although using more epochs could have given a minor benefit to the scores, the time taken would be the main issue.

Later on, when training was shifted to a faster PC, it was kept at this value for the sake of consistency. Having successfully trained a baseline model for LID, the focus could now shift to the applications for low-resource languages.

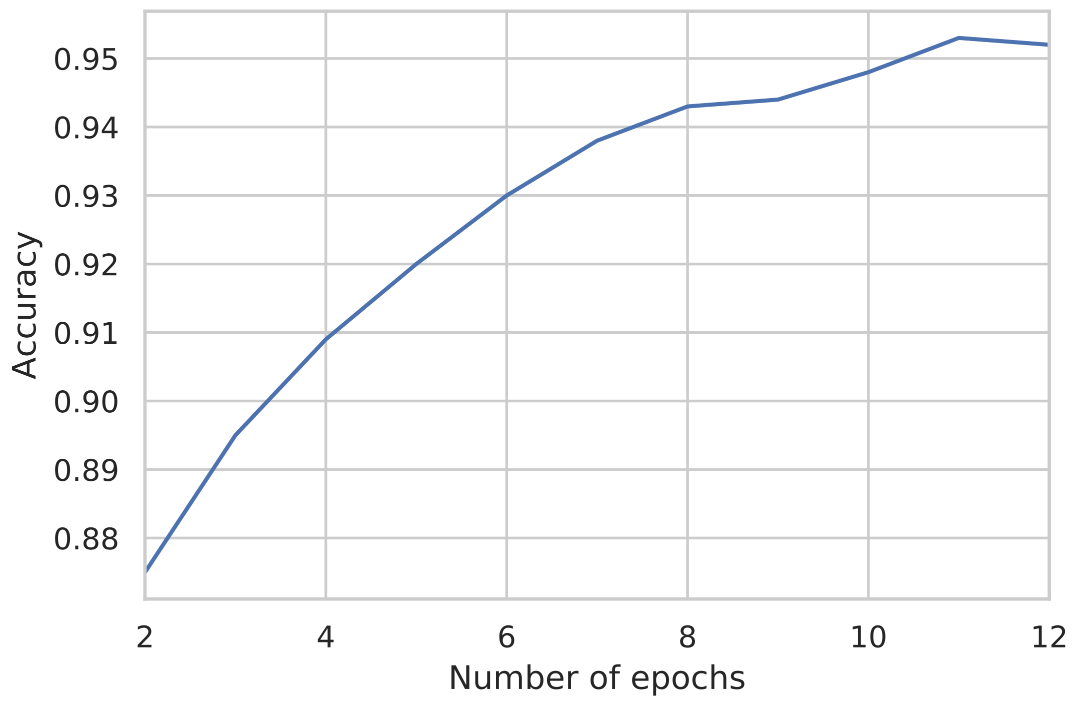


Figure 3.4: Evaluation set accuracy when varying the amount of training

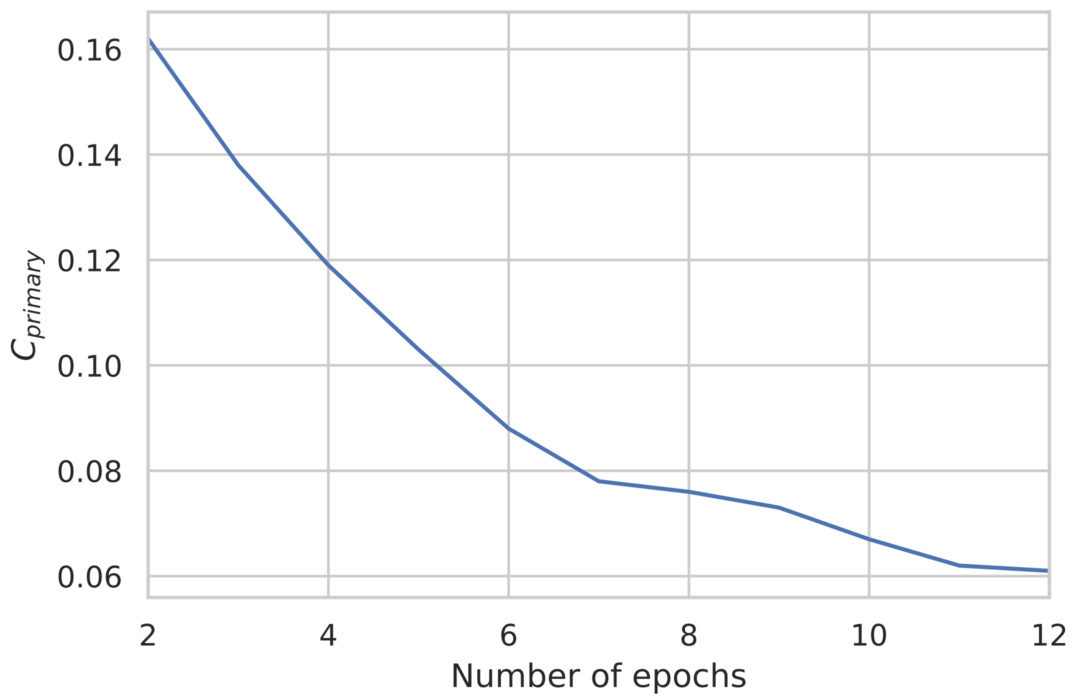


Figure 3.5: Evaluation set $C_{primary}$ score when varying the amount of training

# Chapter 4

# Low-Resource Language Experiments

In this chapter the usefulness of X-vectors for low-resource LID is investigated. To provide some context, the background surrounding low-resource language identification is covered first. Following this, the general changes to the baseline model for the experiments are described. Finally, the actual experiments carried out are each explained and discussed.

## 4.1 Background

### 4.1.1 Challenges of low-resource languages

Low-resource languages pose a unique challenge in natural language processing. While there exist large amounts of spoken data and corpora for common languages like English, or Chinese, there is not nearly as much for others such as Tok Pisin, or Xhosa.

In some cases, the disparity is fairly understandable. According to Eberhard et al. [2019] there are for example relatively few speakers of Tok Pisin ($\sim$4.2 million) - which is only 0.05% of the world's population. It would not make much sense to be heavily invested in producing material for that language, when there are others with considerably more speakers. In fact, Eberhard et al. [2019] claim that just 81 languages (1.2% of the estimated total number of languages), account for 80.5% of the global population's first language. Therefore it would be reasonable to focus more on these.

However, there are some languages which do have tens or hundreds of millions of speakers, but have considerably fewer resources. A few are shown in Table 4.1.

Some corpora such as EMILLE [McEnery et al., 2000], [McEnery et al.] have been built to attempt to address this issue. EMILLE contains written and audio data for languages in India/Pakistan, like Bengali. However, the audio data is taken mostly from the BBC Asian Network radio broadcasts, so there will likely be a domain mismatch between that data and how ordinary speakers sound.

| Language | No. speakers | Global pop (%) |
|---|---|---|
| Bengali | 228 mil. | 2.96 |
| Marathi | 83.1 mil | 1.08 |
| Urdu | 68.6 mil | 0.89 |
| Persian | 52.8 mil | 0.69 |

Table 4.1: A selection of relatively low-resourced languages with at least 50 million speakers. Population statistics taken from Eberhard et al. [2019]

Furthermore, a considerable number of existing low resource language corpora are written, rather than spoken. This is simply because it requires much less time and effort (and sometimes money) to collect written data than to organise recording sessions with tens to hundreds of participants.

Finally, all these complications are further compounded when considering that some dialects of languages could arguably be considered separate languages. One example is Arabic dialects, where Maghrebi Arabic may sound almost unintelligible to Gulf Arabic speakers. Thus it would not be sufficient to regard an LID system trained, for instance, exclusively on the Maghrebi dialect as suitable for classifying any Arabic dialect. If extremely different dialects such as these are taken into account, then the number of low-resource "languages" increases greatly.

### 4.1.2 Low-resource language identification methods

There is not currently a great amount of existing research for low-resource LID. Most of the work with low-resource languages either focuses on dialect identification (which could still be useful for LID), or on topics like automatic speech recognition/machine translation for low-resource languages.

One example of dialect ID, is the work by Biadsy et al. [2009]. They built phonotactic models using parallel phone recognition to distinguish between different Arabic dialects. As mentioned in the previous section, these dialects could possibly be described as low-resource languages themselves. In this case, they were able to achieve accuracies of 81.5% between 5 dialects with 30s segments.

A more direct LID system for low-resource languages was built by D'Haro et al. [2013]. They used a mixture of acoustic-phonetic/phonotactic methods and i-vectors to perform language classification. The data that was used was intended to be similar to that available for low-resource languages. Here it was found that fusing together different models was the most effective technique.

Despite the lack of detailed literature on this topic, there are still certain principles of language and LID which are applicable.

Firstly, a relatively small number of words are used the vast majority of the time, according to Zipf's Law [Piantadosi, 2014]. These specific words should sound quite similar regardless of the speaker, meaning that there are likely to be many similar acoustic-phonetic distributions for the speakers of a language. Therefore it may be possible to build a reasonable language recogniser without many hours of speech, provided enough of the common vocabulary is present.

Secondly, the network structure that was used section 3.3.2 may also provide some advantages. Since it finds utterance-level features in the higher layers, if there is enough variety in the speakers per language, then the utterance level features ought to be robust representations of each language. Thus, it may be the case that quality/variety within training data is more important than sheer quantity. Both of these first two principles are used to investigate how much data is required, in section 4.3.

Thirdly, since some languages are closely related, it may be possible, using the X-vector method, to train the network on languages in a certain family, and add related languages when training the classifier. Or, if the features learned are good enough to distinguish between a wide variety of languages, then adding another language which is not necessarily related may also be feasible. This is researched in section 4.4.

Finally, data augmentation has been discussed earlier when analysing GlobalPhone (section 3.1.2), and was found by Snyder et al. [2018a] to be useful in their experiments with X-vectors. These augmentation strategies may be beneficial to low-resource languages since they could potentially enable better models to be learned from smaller amounts of data. Augmentation is explored in section 4.5.

## 4.2 General Changes to the Baseline System

Once the baseline system had been built, it needed to be adapted to work in the context of low-resouce language experiments. Here, the changes which were applicable to all of them are reported. Any unique changes for an individual experiment will be accounted for in its setup description.

### 4.2.1 Limiting the amount of data

GlobalPhone has hours of training data for each language (Table 3.1). To replicate low-resource language conditions then, it was necessary to find a way to limit the amount of data that could be provided per language.

Furthermore, the selection of utterances for each language should not only be of a prespecified length, but also random. It would not always be possible to find a completely correct solution quickly, since the number of frames for each utterance is highly variable, and there were up to thousands of utterances per language.

To achieve this, firstly configuration files were made to specify the number of seconds allotted for each language. This could be changed for individual languages; however in practice the amount was kept the same for each of them to ensure fairness. Then the frames data for all the languages was obtained, using a standard Kaldi script. 1 second of speech equals 100 frames since the overlap between frames was 10 ms (as normal).

A simple algorithm then found a correct (or nearly correct) solution for each language, provided that there was sufficient training data with a variety of lengths. The utterances for that language were shuffled, and added to a list until going over the limit specified in the configuration. Then the last utterance would be removed, and the remaining data searched for an utterance of appropriate length to fill the gap with as little error as possible. If this perfectly matched the target, then the solution was returned. If not, then the utterances were all shuffled again, and the process repeated until either the limit on the number of attempts was reached or a correct solution was found.

In practice, this was highly effective, and with only up to 20 attempts, the maximum error from the target length was typically as low as 0.002-0.001%, and was frequently 0%. This new list of utterances was used to filter the .scp files used by Kaldi as well as important files with utterance lists, such as `utt2lang`.

Some filtering of the utterances themselves also had to be done before producing a shortened list. The allocation of training examples for the neural network required a min/max number of frames for each chunk of examples. If the max number of frames for a given chunk was greater than an utterance being added to it, then an error would be thrown.

Having larger chunks was beneficial since it would allow the network to get more data on each propagation, and be more likely to learn useful features. If the chunks were too small, then it could be harder to identify distinguishing elements between languages. So having longer utterances would be useful for training.

However, the purpose of the experiments was to investigate low resource languages, for which utterances might well be quite short. A compromise had to be reached, so the maximum chunk size was set to 250 frames, with a minimum utterance length of 300 frames (not including silence frames as mentioned earlier at the end of section 3.3.1).

### 4.2.2 Other changes to the baseline model

For the adaptation and augmentation experiments, different lists of languages were required - for instance, so that certain languages were not present in the training data, but were in the enrollment data. After including this change, it was used in these experiments by simple changes to the configuration files.

There were a whole host of minor bug fixes and improvements made as time went on. However, the overall functionality of the training and evaluation remained the same as in the baseline system (apart from the error with the learning rate).

One problematic mistake in the training was not discovered until late in the experimentation phase. At one point, the initial learning rate had been accidentally lowered to 0.0001 instead of 0.001, as it was in the baseline model. The effects of this change were not clearly visible - some accuracies seemed lower, but this was initially thought to be due to other minor changes in data processing.

Due to the time at which the mistake was found, there was not practically enough time to rerun everything afresh. However, since the error affected nearly all the experiments apart from a few ones early on, only those early experiments needed to be rerun. Any results are therefore consistent, since they all had the same lower training rate, and hence they are still valid for comparisons.

## 4.3 Experiment 1 - Amount of training/enrollment data required

### 4.3.1 Aims and setup

The goal of this experiment was to found out exactly how much language data was required in order to train the network to a reasonable standard. As mentioned earlier, the data used from GlobalPhone had on average, 22 hours of data per language, of which about 17 hours was available for training.

How much of this training data was necessary though, to train the network? In the context of low resource languages, it is obviously advantageous to require as little as possible.

There were effectively two occasions of "training" in the model - once for the network with the training data, and once for the classifier with the enrollment data. Both of these were varied at 500, 1,000, 5,000 and 10,000 seconds of data respectively.

In order to save time, a model trained on, (for example) 500 seconds of data would be used repeatedly for the 500-10,000 second lengths of enrollment data. An additional timesaver was to only extract the evaluation X-vectors once for each trained network, since they would not change in any of the enrollment data variations (the same network was being used). These evaluation X-vectors could simply be copied in those cases, instead of being calculated all over again.

One issue observed was that there was not enough Shanghainese data to get 10,000 seconds of enrollment/training data. In this case, all the available data was used instead, but this did not seem to cause any serious issues for that language.

To act as a reference, a baseline model was trained on all the data, with only utterances shorter than 3s filtered out. Each experiment was repeated 3 times to reduce the chances of the results being random.

### 4.3.2 Results and discussion

There are two broad areas to consider when discussing the results. Firstly, there are the effects of the amount of training data on the training/validation accuracies during the network training phase. Secondly, there are the effects on the evaluation utterances of varying training/enrollment data results, in terms of accuracy and $C_{primary}$.

The training/validation accuracies during network training are shown in Figure 4.1 and Figure 4.2. These curves were smoothed to make the trends easier to discern. The smoothing was done by computing the coefficients of an interpolating B-spline with a degree of 3 (from `scipy.interpolate`).

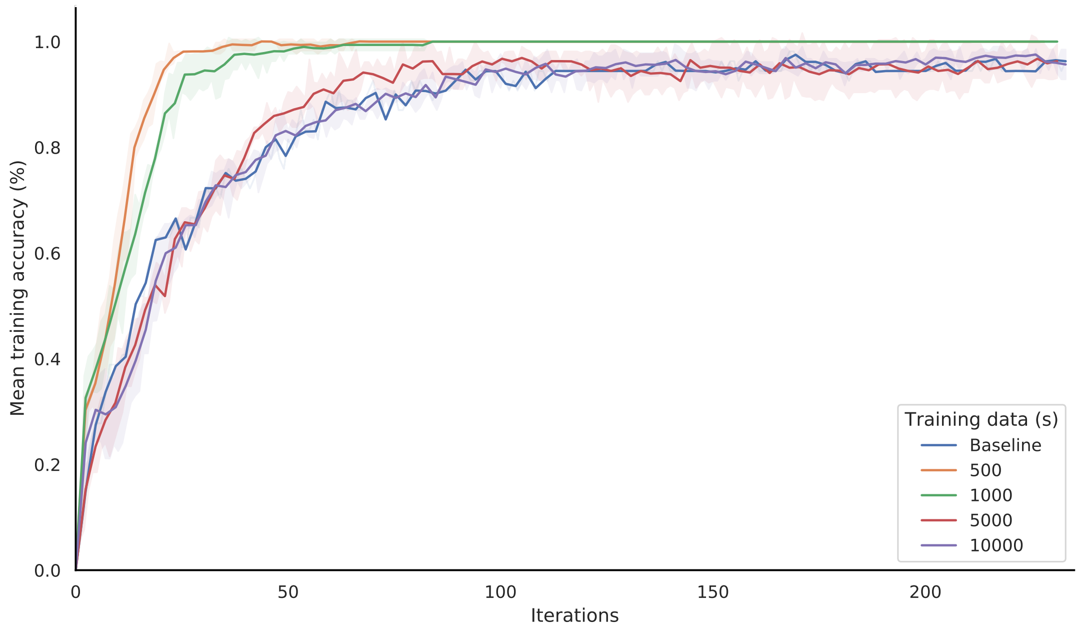


Figure 4.1: Smoothed mean training accuracy during network training for different amounts of training data. The baseline includes all of the training data. Shaded areas are the standard deviation

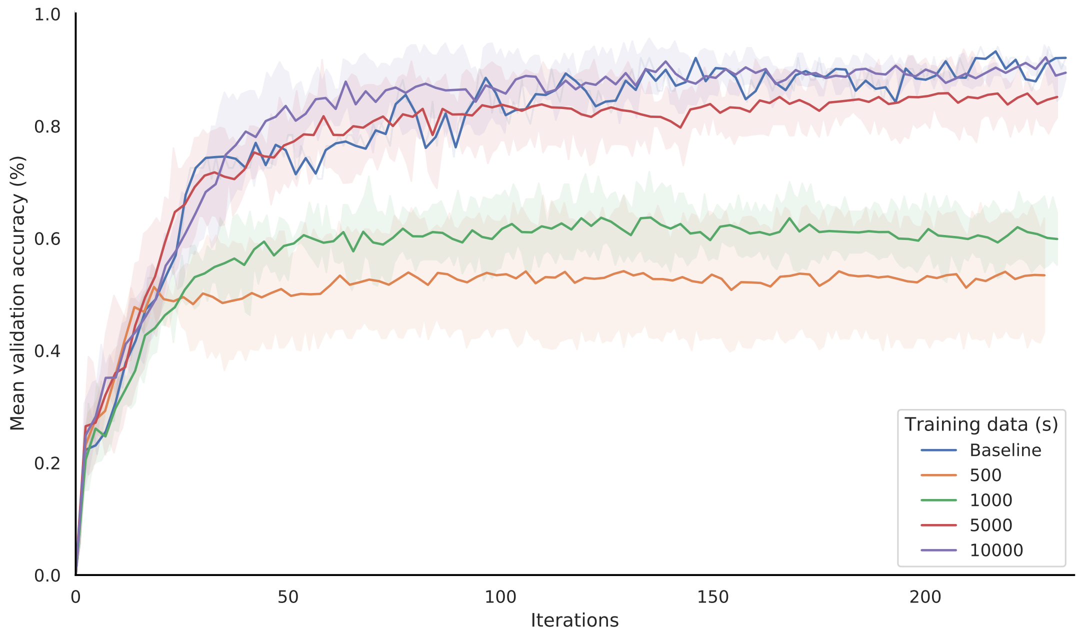


Figure 4.2: Smoothed mean validation accuracy during network training for different amounts of training data. The baseline includes all of the training data. Shaded areas are the standard deviation

One noticeable aspect of the training accuracy curves is that the accuracy seems to become perfect after about 100 iterations when only 500/1000 seconds of data is available per language. It may be the result of the network learning all the features that it possibly can from such limited data. Furthermore, with less training data (at least 5x less than the other experiments), it will be considerably easier for the data to be overfit.

These training accuracy trends explain why the validation accuracy for 500/1000 seconds of training data also plateaus at roughly the same time. By that point, backpropagation will have little to no effect on the network if the training data is almost all perfectly classified.

By contrast, the training/validation accuracy curves with 5000 seconds or more of training data are all very similar. Notably, the training accuracy does seem to plateau, but not at perfect accuracy, which explains why the validation accuracy slowly increases. Unlike before, the network is still learning in some meaningful way, albeit slowly.

One reason for the lower training accuracy with more data, is that the intra-language variety is increased. Thus, the language features that the network is learning have to be more robust to all the extra variance. Another reason is sheer probability - mistakes in training outputs are more likely to occur with greater amounts of data.

Interestingly, doubling the data from 5000 to 10,000 seconds only produced a slight improvement in validation accuracy. Furthermore, when using all of the training data (about 57,000 seconds per language on average (Table A.1)), the validation accuracy seemed to be about the same.

These trends are further confirmed by the results shown in Figure 4.3; the second area of results to be discussed.

For comparison, the baseline system results are provided in Table 4.2. NB: this was a revised baseline with the lower learning rate described in section 4.2.2.

| Evaluation length (s) | Accuracy | $C_{primary}$ |
|---|---|---|
| 3 | 0.741 | 0.344 |
| 10 | 0.852 | 0.204 |
| 30 | 0.873 | 0.189 |

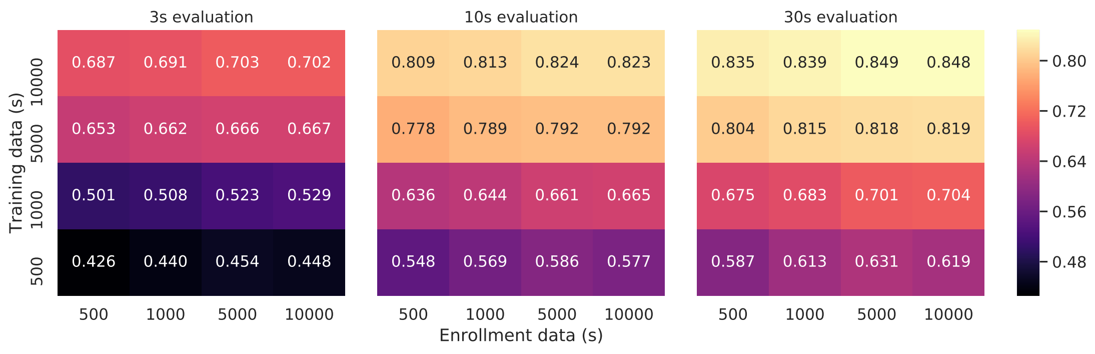
Table 4.2: Performance of the baseline system for various lengths of evaluation utterances

One interesting trend seen in these heatmaps is that the effect that varying the quantity of enrollment data appears to have. It seems that at least 5,000s is required to achieve a good performance. However, doubling this to 10,000s of enrollment data, provides little to no improvement most of the time, regardless of how much training data is provided.

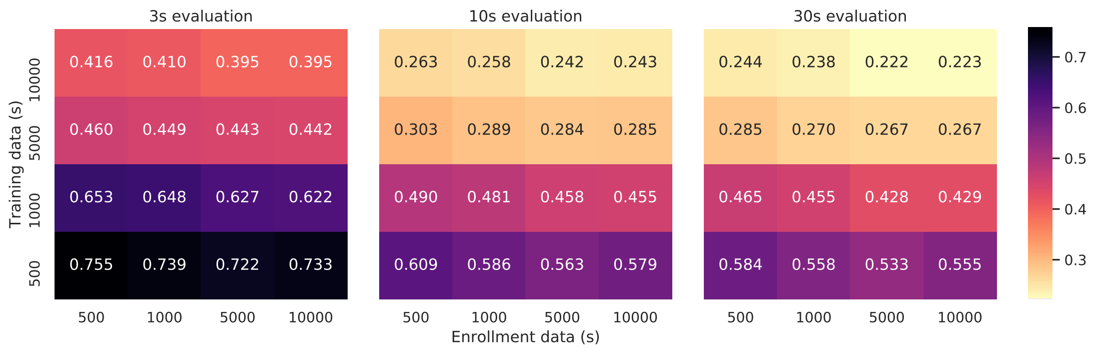
Therefore, the amount of training data appears to be the limiting factor when comparing these restricted models to the baseline. Interestingly, at least 10,000 seconds of training data results in comparable performance to the baseline system. For instance, with evaluation data of 30 seconds, the decrease in accuracy is only 2.9% and the increase in $C_{primary}$ is just 18%. Thus the trend in validation accuracy seen earlier is confirmed - increasing the quantity of training data only helps up to a point, which seems to be close to 10,000 seconds.

Nonetheless, it is worth pointing out that the validation accuracy is only for the network training phase, and not the classification part. This explains why there is a visible difference between 5,000s/10,000s/all of training data in the final classification, but not in the validation accuracies. These classifications are based on the learned embeddings of the network, rather than just the network itself, so will benefit more from learning hidden features. As stated earlier, increased amounts of data with more intra-language variance means that the network's hidden layers will need to be more robust in how they represent each language.

Overall though, the implications for low resource language LID from this experiment are encouraging. With at least 15,000 seconds of data per language (10,000 for training, 5,000 for enrollment), a very decent model can be built. That is only about 4 hours of spoken data, which is considerably less daunting to procure. Training data appears to be the most important factor; about 1 hour of enrollment data is sufficient, and having more does not appear to produce much of a benefit.



(a) Mean classification accuracies



(b) Mean $C_{primary}$ scores

Figure 4.3: Heatmaps showing mean performance of the logistic regression classifier across 3 experiments, when varying the amount of training/enrollment data. Performance is measured in terms of raw accuracy [4.3(a)] and $C_{primary}$ [4.3(b)]. The colouring is reversed between these, since high accuracy, but low $C_{primary}$ are desirable. The left, middle and right heatmaps show results when the evaluation data is up to 3, 10 and 30 seconds in length, respectively.

## 4.4 Experiment 2 - Language adaptation

### 4.4.1 Aims and setup

In the second experiment, the potential applications of X-vectors to language adaptation were investigated. Being able to adapt a prebuilt model to “learn” an extra language would be useful, since this should require less data and training time.

5,000 seconds of training data was used, along with 5,000 seconds of enrollment data. This allowed for a replication of low-resource language conditions. Based on the results from the first experiment (section 4.3.2), these quantities seemed to be a sufficient amount to train the network reasonably well.

Since the experiments were intended to explore the effect of the model being adapted with languages within and without the same family, the language families present had to be identified first. A diagram of the familial relations in the languages used is shown in Figure 4.4.

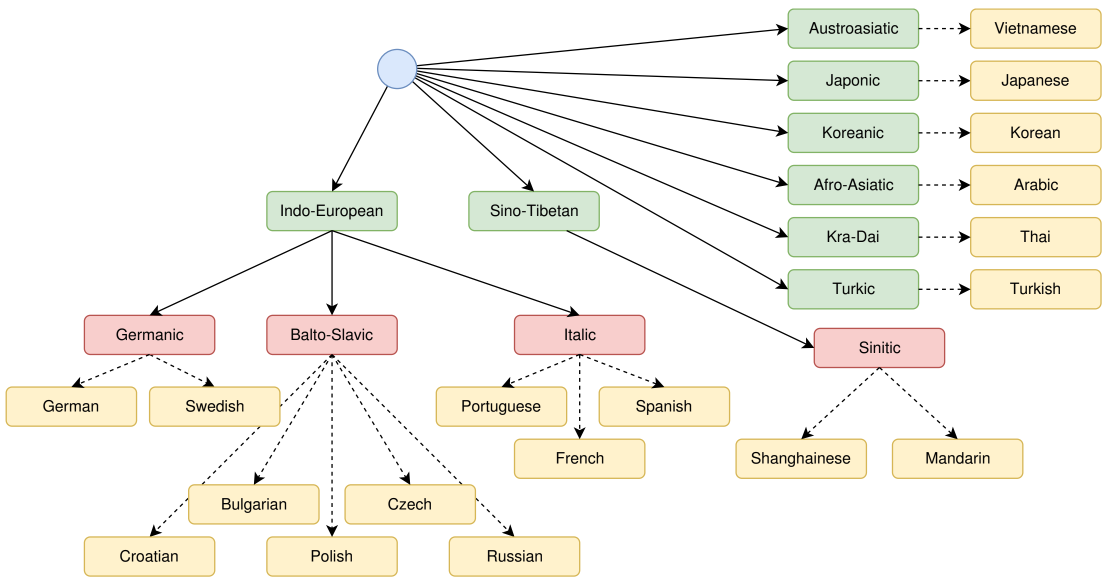


Figure 4.4: Language family relations for the GlobalPhone languages used. Green/red boxes show language families; yellow boxes are the languages. A dashed arrow indicates there are intermediate families which are skipped for clarity

The large number of families and combinations seen in Figure 4.4 meant that it was impractical to investigate all possible combinations. Instead, just two setups were decided upon.

For the first, the network was trained on all the languages, bar one. The missing language was then included in the enrollment data. The aim was to see if the network's learned features were enough to distinguish the new language from the rest in the classifier, even if the language families were different.

The second setup was the same, except instead of all of the languages, only ones from the Balto-Slavic language family were included. NB: all of these were also part of the

Slavic family, so will be referred to as Slavic. This family was chosen because it had the greatest number of languages (Italic, Germanic and Sinitic only had two or three, and the others were unique). The aim here was to see if learning features for a single language family made it easier to adapt the model to another language in that same family.

The baselines were also different for each setup. In the first one, the same 5,000s training/enrollment models used in the previous experiment (section 4.3.2) were taken as the baseline. For the second setup's baseline, a model with the same quantities of data (5,000s training/enrollment) was trained exclusively on the Slavic languages.

All experiments were repeated at least twice, again to improve the reliability of the results.

### 4.4.2 Alternative performance metric

Unlike the other two experiments, accuracy and $C_{primary}$ were not appropriate metrics for this experiment. Those metrics are based on the performance of the model across all the languages. Here, the focus needed to be on how well individual languages were classified.

Therefore, the precision and the recall for each language were a better measure of performance, since they give an idea of how well an individual language is classified. A reasonable metric to use with them was the $F_1$ score. This is the harmonic mean of precision and recall:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.1}$$

The mean $F_1$ score was compared for a language when it was present in training the network ($F_{baseline}$) versus when the language was only present in the enrollment data ($F_{enroll}$). The error was taken to be the standard deviation.

The percentage decrease in $F_1$ gave an idea of the relative performance cost of adding the language later in the enrollment data versus training it in the network:

$$Percentage\ decrease = \frac{F_{baseline} - F_{enroll}}{F_{baseline}} \times 100 \tag{4.2}$$

The error calculation for this decrease was more complicated, since there was error in both $F_{enroll}$ and $F_{baseline}$, which is combined during both the subtraction and division steps. Let $err(b)$ be the error in $F_{baseline}$ and $err(e)$ be the error in $F_{enroll}$. Then the total error in the percentage decrease is:

$$\frac{F_{baseline} - F_{enroll}}{F_{baseline}} \sqrt{\frac{(err(b))^2 + (err(e))^2}{(F_{baseline} - F_{enroll})^2} + \left(\frac{err(b)}{F_{baseline}}\right)^2} \tag{4.3}$$

### 4.4.3 Results

First, the results for the model trained on all the languages (bar one) will be covered. These are summarised in figures Figure 4.5 and Figure 4.6. The evaluation utterances used for classification were at most 30 seconds long.

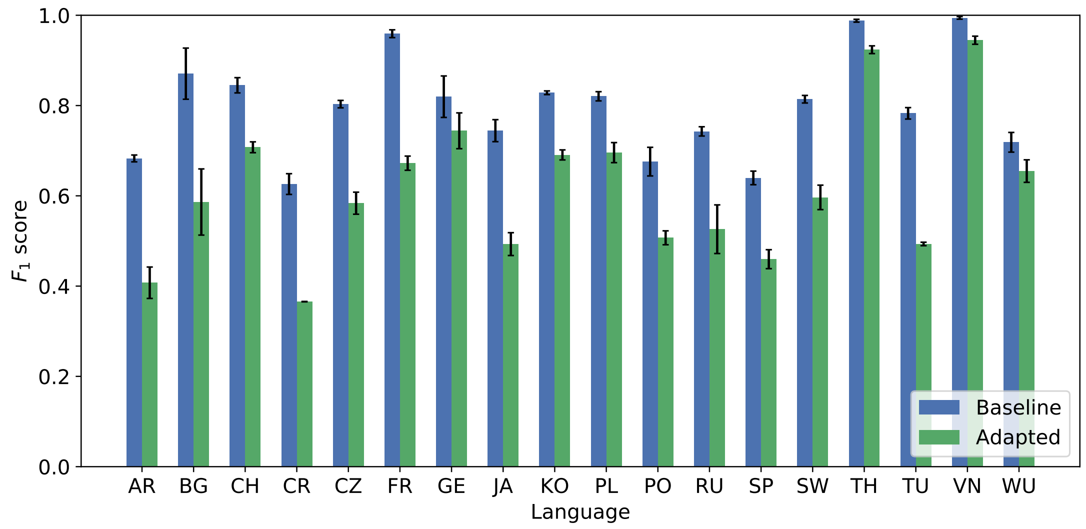


Figure 4.5: F1 scores for the adaptation experiment with all the languages in GP. Evaluation length was 30 seconds. Blue bars are the baseline F1 scores for each language when the model is trained on all languages. Green bars are the F1 score for each language when it is only included in the enrollment data, while the model is trained on all the other languages

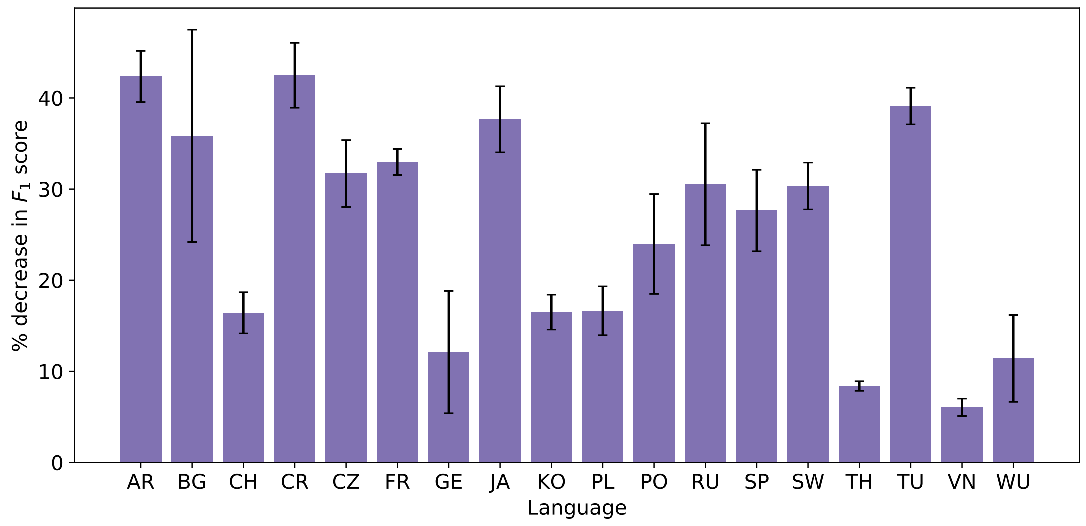


Figure 4.6: Percentage decrease in $F_1$ score for each language when not included in the training data, from the results in Figure 4.5. Evaluation utterance length was at least 30 seconds. Error bars are the combined error in $F_{baseline}$ and $F_{enroll}$

Figure 4.5 is helpful for comparing which languages cope well with being added later, and which do not. Interestingly, the decrease seems to be only around the 20% mark for most languages, suggesting that the embeddings learned are indeed reasonable at capturing the distinguishing features of different languages.

To explore reasons why there might be considerably more errors for some languages, it was necessary to discover where exactly the mistakes were made for each of them. The findings are summarised in Table 4.3.

| Language | Baseline | | Adapted | |
|---|---|---|---|---|
| | Top 3 FP | Top 3 FN | Top 3 FP | Top 3 FN |
| Arabic | German (43) | Russian (59) | German (43) | Russian (56) |
| | Spanish (23) | Swedish (34) | Spanish (31) | Swedish (47) |
| | Chinese (9) | Spanish (27) | Chinese (20) | Portuguese (36) |
| Croatian | Russian (84) | Russian (108) | Russian (138) | Russian (146) |
| | Swedish (62) | Spanish (42) | Swedish (71) | Spanish (77) |
| | Turkish (38) | Swedish (26) | Turkish (35) | Turkish (32) |
| Turkish | German (46) | Croatian (38) | German (64) | Russian (69) |
| | Russian (31) | Russian (37) | Croatian (45) | Croatian (66) |
| | Croatian (23) | Portuguese (25) | Russian (34) | Portuguese (29) |

Table 4.3: The three languages here were the ones with the greatest percentage decrease in $F_1$ in the baseline model versus the adapted one. This table gives a comparison of the top 3 false positives (FP) and false negatives (FN) in these models respectively. Numbers have been rounded to whole numbers to make the table easier to read.

From Table 4.3, it appeared that the specific languages which were confused did not change greatly between the two models (baseline and adapted). The mistakes were simply more frequent in adapted models.

This suggested that the determining factor was the langauges which were chosen for inclusion in the first place. If a language is very distinct, like Vietnamese in this context, then it would be classified correctly quite often. However, if a language was already easy for the model to confuse with others (as evidenced by how frequently it would misclassify that language in the baseline system), then the effects of this were only exacerbated when a model was adapted to that language later.

With regard to the Slavic language setup, the mean confusion matrices for the baseline model are shown in Figure 4.7. The $F_1$ scores and the percentage decrease between the baseline and adapted model are shown in Figure 4.8.

Unfortunately, the $F_1$ score for each Slavic language did not appear to be particularly improved upon, even though the network was trained exclusively within that family: all the $F_1$ scores were still roughly the same as when the model was trained on all languages. This could be explained by the fact that the Slavic languages tended to be confused with one another in the original model. Therefore any difficult-to-classify utterances found in the original model (with all the languages) would still be likely to cause issues in classification in the smaller model, since the same dataset was used for evaluation.

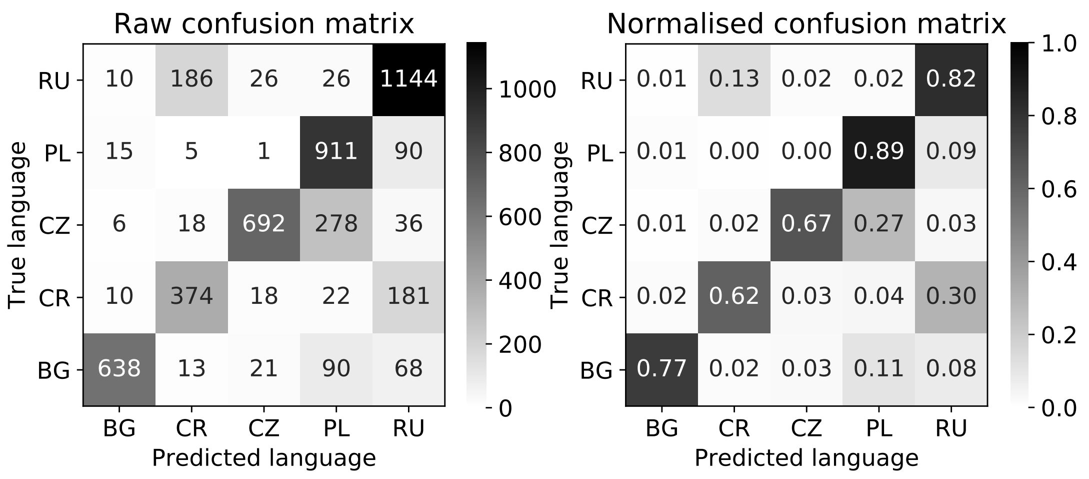


Figure 4.7: Mean confusion matrices for the baseline Slavic model with utterances 30 seconds long in the evaluation data. Numbers in the raw matrix are rounded to whole numbers to make it easier to read. The normalised matrix was normalised along the axis of the true languages, thus showing the precision of the model.

Additionally, the adapted $F_1$ scores tended to be slightly worse in the Slavic model than the full one (apart from Croatian and Russian which saw some improvement). This seems to confirm the earlier conclusion that the benefit of language adaptation is heavily dependent upon the actual langauges in the model, and how similar they are.

In summary, language adaptation can be done with X-vectors, but the results are very hit-and-miss. Instead, it often seems that it would be better to include the language in training, (and work on improving that phase), than to try adding it later.

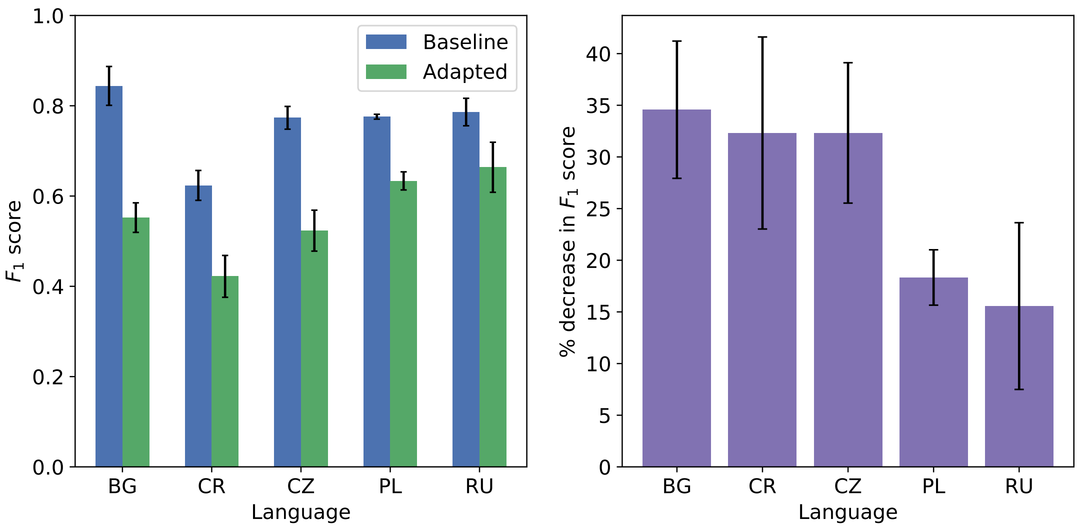


Figure 4.8: The left chart shows the $F_1$ score for each language: both when included during the network training (the baseline), and when not (the adapted model). The right chart shows the percentage decrease in the $F_1$ score between the baseline and adapted model

## 4.5 Experiment 3 - Language augmentation

### 4.5.1 Aims and setup

The goal of the third and final experiment was to explore how useful data augmentation was for improving performance.

There were three methods of augmentation used:

- speed perturbation - at 0.9x and 1.1x speed
- adding reverberation - using the RIRS dataset [Povey, 2017]
- doing both of these and putting the sets all together

To augment the data, it was first shortened to a specified length for each language - 500, 1,000, 5,000 or 10,000 seconds as in the first experiment. Then, each of the remaining utterances were augmented, and combined with the original ones.

The RIRS augmentation did not change the length of the utterances, so the augmented data was simply double the length of the original. With speed perturbation, for any utterance of length $x$, there were utterances of length $0.9x$ and $1.1x$ added, giving a total increased length of 0.9 + 1 + 1.1 = 3 times the original length. Lastly, when using both RIRS and speed perturbation, the overall length was 0.9 + 1 + 1 + 1.1 = 4 times the original.

For the purposes of comparison, clean data was created at the same time as the augmented lists were. The shortened list of utterances was found first, and these utterances set aside. Then an extra set of different utterances were found, and added to the list of utterances which had been selected for augmentation. The total length was kept the same as the augmented data.

This made it possible to see how helpful simply adding more data is versus sticking with less data and augmenting it. The hope was that while augmentation would be unlikely to outperform clean data, it could at least be comparable in terms of performance, and be better than nothing.

One last point of difference in this experiment to the preceding two, was that the Shanghainese language was dropped. It was too short, and could only work if the amount of data was restricted to either 500 or 1000 seconds, but for the longer experiments there was not enough. While this shortness had not been a serious issue in either of the previous two experiments, it was important here since the length of the data was crucial to making clean data for comparing. This meant that the results from the first experiment (section 4.3) could not be used as baselines, so new baselines were trained without Shanghainese, for all the various lengths of training data.

### 4.5.2 Results

Figure 4.9, Figure 4.10 and Figure 4.11 compare the performance of various augmentation methods versus the baseline and clean data. Figure 4.12 compares the $C_{primary}$ score and accuracy respectively for the all the augmentation methods with each other. All of these were evaluated with utterances at most 30 seconds long.

A summary is provided in Table 4.4, which shows how much of an improvement in $C_{primary}$ was achieved using augmented or clean data.

| Augmentation method | Original amount of training data (s) | | | |
|---|---|---|---|---|
| | 500 | 1000 | 5000 | 10000 |
| Reverb. | 4.7±1.9 | 2.0±0.9 | 1.1±1.3 | 0.4±0.1 |
| Reverb. (clean) | **5.5±1.0** | **10.3±1.1** | **4.1±0.5** | **0.9±0.4** |
| Speed | 18.4±0.3 | 12.3±0.6 | **9.0±0.4** | **6.5±0.2** |
| Speed (clean) | **24.8±0.7** | **15.3±1.0** | 4.8±0.4 | 2.3±0.1 |
| Reverb + Speed | 20.5±0.6 | 18.0±0.8 | **12.0±1.0** | **10.1±0.1** |
| Reverb + Speed (clean) | **26.1±0.6** | **18.4±1.2** | 6.0±0.5 | 2.6±0.1 |

Table 4.4: The percentage decrease in $C_{primary}$ from the baseline, when using the various augmentation methods. The amount of training data refers to how much data was available to be augmented. Rows marked with (clean) refer to use of unaugmented data (of the same total length as the augmented data). Errors were computed using the formula in Equation 4.4.2, adapted to $C_{primary}$ instead of $F_1$. Bold entries indicate whether the augmented or clean data were better

Firstly, it can be seen the effectiveness of augmentation is greatly affected by how much training data there is to begin with, decreasing considerably with greater amounts. This is simply because with more training data, the network can be trained more effectively in the first place (up a point as discussed in section 4.3.2), making augmentation relatively less useful.

Secondly, using reverberation alone was not particularly effective. While it did produce a slight decrease in $C_{primary}$, adding clean data was always better. This may be because reverberation does not really provide that much meaningful variety in the training utterances, and/or because the evaluation data will have relatively little audible reverberation due to the way GlobalPhone data was collected. Ko et al. [2017] certainly seemed to find in their initial experiments that there were benefits of adding these reverberations, but that these benefits were relatively small.

Thirdly, adding more clean data always outperformed augmented data when only 500 or 1,000 seconds of training data were used. This was probably due to similar reasons discussed in section 4.3.2, where very low amounts of data were considerably improved by adding more data. Having this extra clean data is likely better than augmentation in these cases because it has greater intra-language variety when there are relatively fewer utterances to begin with. Augmentation is simply not as good at producing similar amounts of variation for these smaller quantities of data.

Fourthly, speed perturbation seems to be a very effective augmentation technique - it achieved the best results when there were 5,000 or more seconds of training data, with or without reverberation. This may be because intra-language variety begins to peak when simply adding more data, when there is already lot to begin with. However, the speed changes provide considerably more variety, hence allowing the network to learn more robust features of the languages. Furthermore, people speak at different speeds, so learning from a wider range of these requires the hidden language features to be more general, and higher-level, thus matching evaluation data better.

Fifthly, it is worth noting that even though clean data is better for 500 or 1,000 seconds of training data, speed augmentation still manages to produce a considerable decrease in $C_{primary}$. As mentioned previously, this is likely because it offers more utterances per language which are noticeably different. This has great potential for being used in a low-resource langauge context - with only 1,000 seconds (16 minutes) of data per language, it was possible to train a model which performed almost as well as one with 5,000 seconds (about 1.4 hours).

In conclusion, data augmentation offers a real benefit to LID systems using X-vectors. While reverberation alone was not very effective, speed perturbation was, and when combined with reverberation was even better, outperforming the same amount of clean data in some cases. The applications to low-resource LID are significant, since these results indicate that data augmentation can give a considerable boost when the amount of data is limited. The best model was tested on the test set, and found to get a $C_{primary}$ score of 0.097, which was in fact better than the evaluation results on that model.

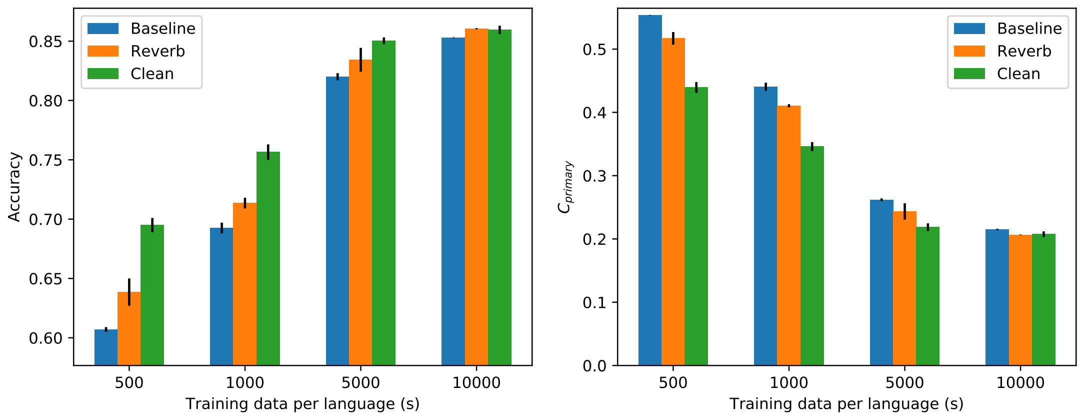


Figure 4.9: Comparison of adding reverberated data or the same amount of clean data, versus the baseline model in terms of overall accuracy and $C_{primary}$. The evaluation length was 30 seconds.

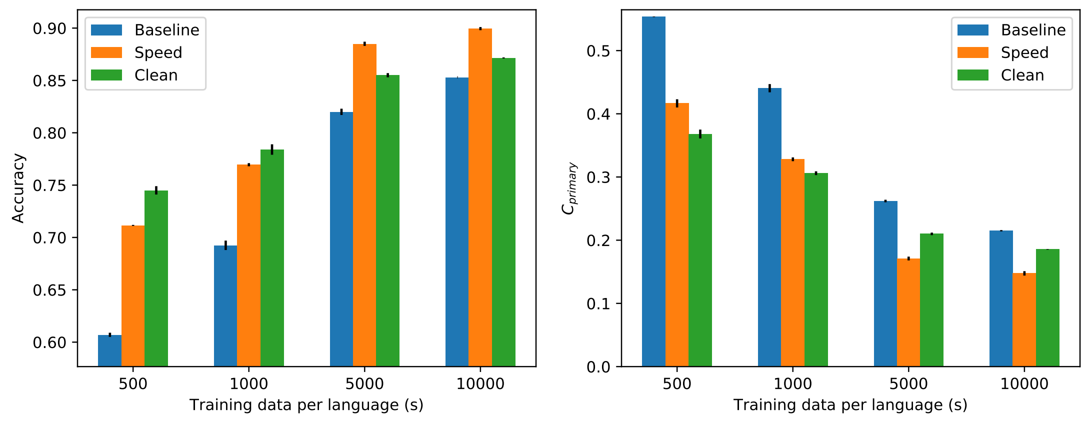


Figure 4.10: Comparison of adding speed-augmented data or the same amount of clean data, versus the baseline model in terms of raw accuracy and $C_{primary}$. The evaluation length was 30 seconds.

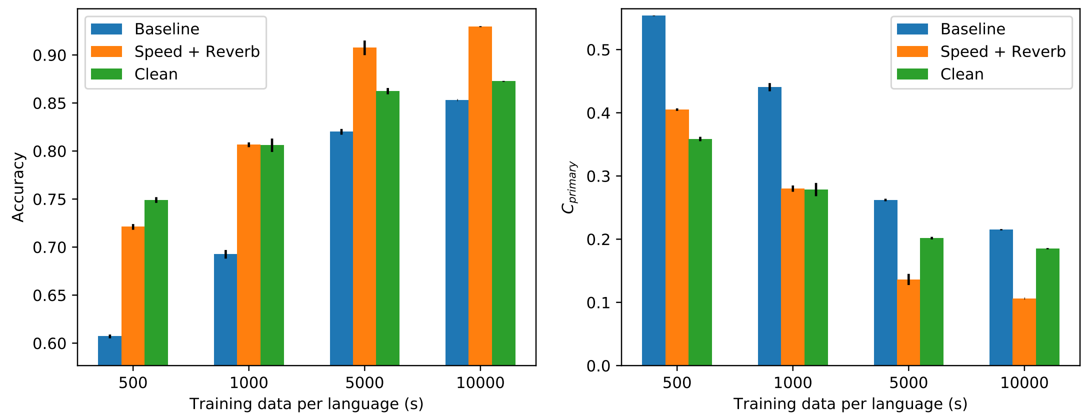


Figure 4.11: Comparison of adding reverberated and speed-augmented data, or the same amount of clean data, versus the baseline model in terms of raw accuracy and $C_{primary}$. The evaluation length was 30 seconds.

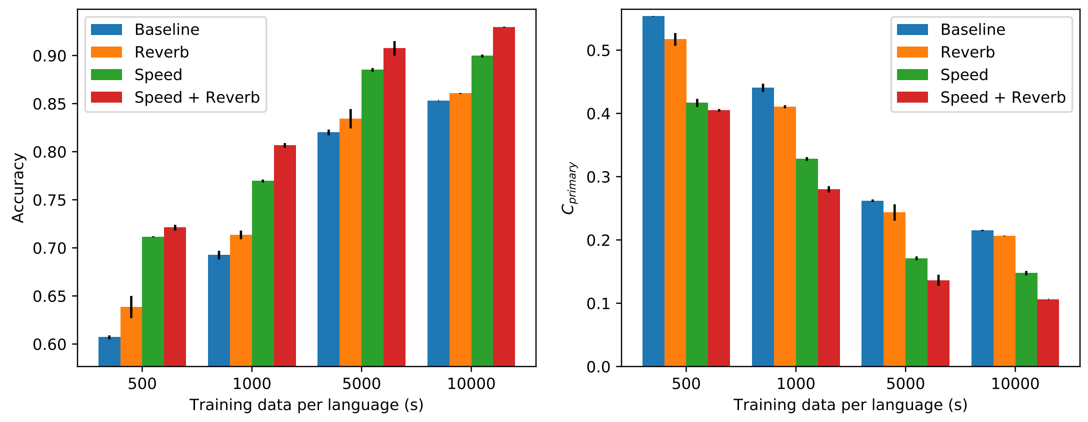


Figure 4.12: Comparison of all the augmentation methods in terms of $C_{primary}$ and accuracy. The evaluation length was 30 seconds.

# Chapter 5

# Conclusion

## 5.1 Results summary

Through the research carried out for this report, it has been found that X-vectors are a powerful tool for LID, particularly in the context of low-resource languages. Even in the baseline system, the results were quite good.

Firstly, they do not require huge amounts of data to perform at a very reasonable standard. With only 5,000 seconds of enrollment data, and 10,000 seconds of training data per language (about 4.17 hours), the model performed at a similar level to one trained on about 16 hours per language. Thus, they can be used with the limited data available for these languages.

Secondly, while there is some potential for adapting low-resource languages to a pre-trained X-vector network, the effects of this were quite temperamental, being very good for some languages, but quite poor for others. Furthermore, the effectiveness was largely determined by the languages already trained in the network, as certain languages were very likely to be confused with one another. It seems that adaptation would be more feasible if the focus is firstly on ensuring that the network and classifier are really strong at classifying languages (especially similar ones) to begin with.

Finally, using speed perturbation combined with reverberation to augment the training data allowed for a much better, more robust network to be trained with relatively little data. In some cases, it even outperformed adding the same amount of clean data, and if not then the performance was still quite close. This is is very useful for low-resource languages since it allows for a model to enjoy the benefits of having more data without actually requiring it.

In summary, the X-vector approach may allow for low-resource LID to be more possible to achieve than it was before. Furthermore, there are many more lines of investigation possible for further improvements, which are described in the following section.

## 5.2 Future Work

### 5.2.1 Network architecture

The same network structure used by Snyder et al. [2018a] in their LID experiments was used here, but there may well exist a superior architecture.

Some simple changes would be to add extra layers or change their dimensionality. Other more complicated ones could involve training networks separately and fusing them together. The fusion could either be networks for the same languages trained on different features, or perhaps for the same features, but different langauges (a different approach to language adaptation).

Gonzalez-Rodriguez [2018] for instance used a fairly similar network to X-vectors, except with different dimensionalities and had BLSTM layers for some of the frame layers instead of TDNNs. It would be worth finding out any of these differences could improve upon the X-vector network.

### 5.2.2 Feature type

Another area of investigation would be in the feature type used. My colleague Samuel Sućik investigated this area in more depth after we branched out separately from the baseline. He found that using different feature types/combinations (e.g. SDCs, prosodic features) could be very effective in improving the training.

Furthermore, multilingual-BNFs (MLBNFs) were recommended by Snyder et al. [2018a] as the best features they found for language recognition. They were not used in the experiments in this thesis because there were none readily available. It would have required a great deal of extra time to set up the design and training of an entirely separate network, which was simply infeasible.

MLBNFs are likely to be better than MFCCs since they capture the linguistic features of a frame rather than just frequency features. This allows the network to learn from better features which are already tuned for distinguishing between different languages in the first place (to an extent, depending on the languages the MLBNFs were trained with).

However, these would likely require more data for training than is available in GlobalPhone, so another area to explore would be trying out new corpora, as described in the next section.

### 5.2.3 Using other corpora

One of the issues with the GlobalPhone corpus mentioned previously in section 3.1.2 is that it consists of controlled read speech. This makes it a relatively “easy” corpus to work with, so LID systems which perform well on it may not actually do that well in general.

Instead, it may be helpful to research how useful X-vectors are on other corpora such as BABEL [Roach et al., 1996], ones published by ELRA, or any others which are easily accessible.

In particular, corpora containing more conversational speech would be useful, because an LID system trained on this type of speech would likely be more reliable in these situations (where it would ideally be used). It would hopefully not require people to slow down as much when speaking, and would feel more natural to them (though people do tend to automatically slow down their speech if they know they're talking to a computer).

### 5.2.4 Dialect identification

This was mentioned in section 4.1 as a related challenge for low-resource LID, since some dialects in certain languages are almost completely different languages from one another.

Shon et al. [2018] explored dialect identification, something that could be investigated using a different corpus. The two main methods they tried were to use acoustic features in an end-to-end system, or to train a Siamese network to find embeddings; discovering that a fusion of the two was the optimal configuration.

It may be the case that X-vectors could be a more useful type of embedding in this situation, and would be worth researching. Another possibility is that redesigning the X-vector architecture to be more like a Siamese network could improve the performance on dialects of the same language.

### 5.2.5 Domain mismatch

Alternatively, rather than leave GlobalPhone altogether, one possibility would be to train models using it, but then test them on data from these other corpora and vice versa. A domain mismatch between training and testing would provide some evidence for whether or not GlobalPhone is too "easy" to learn from. Furthermore, it would show whether or not the X-vector approach is robust against this kind of mismatch - ideally it ought to be, but the real world is often far from ideal!

Domain mismatch is also a very relevant problem in LID: some recent research into dealing with it was done by Lopez et al. [2018]. To reduce overfitting, they adapted the learned emdbeddings to the domain, using two different approaches, a linear Gaussian backend and an end-to-end neural network one. They found this to be useful, improving the model by up to 8%.

Additionally, Mclaren et al. [2018] explored adapting to multiple domains, finding that having a Gaussian backend which was weighted to language/domain, source normalisation of i-vectors, and duration-dependent calibration/fusion when classifying test utterances all provided major benefits.

It would be worthwhile investigating which of these different methods outlined in both papers could be applied or adapted to X-vectors. If domain mismatch issues can be reduced, this would further solidify X-vectors as a competitive approach to LID, and have great benefits for low-resource languages where domain mismatches are quite likely.

### 5.2.6 Summary

In conclusion, there are a great many further directions of potential research for this project for the upcoming year. These include relatively simpler things like tweaking the architecture, using different features, and trying out new corpora. There is also potential to explore the harder challenges of dialect identification, and/or domain mismatch issues, both of which are particularly applicable for low-resource languages.

# Bibliography

Wafia Adouane and Simon Dobnik. Identification of languages in algerian arabic multilingual documents. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 1–8. Association for Computational Linguistics, 2017. doi: 10.18653/v1/W17-1301. URL `http://aclweb.org/anthology/W17-1301`.

Rosette Text Analytics. Accurate language detection for queries & tweets, 2014. URL `https://www.rosette.com/blog/language-detection-for-queries-tweets/`.

B. S. Atal and Suzanne L. Hanauer. Speech analysis and synthesis by linear prediction of the speech wave. *The Journal of the Acoustical Society of America*, 50(2B): 637–655, 1971. doi: 10.1121/1.1912679.

Anoop S. Babu. Comparing neural network approach with n-gram approach for text categorization. *International Journal on Computer Science and Engineering*, 2(1): 80–83, 2010.

Adrien Barbaresi. An unsupervised morphological criterion for discriminating similar languages. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 212–220. The COLING 2016 Organizing Committee, Dec. 2016. URL `http://www.aclweb.org/anthology/W16-4827`.

Fadi Biadsy, Julia Hirschberg, and Nizar Habash. Spoken Arabic dialect identification using phonotactic modeling. In *Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages*, pages 53–61. Association for Computational Linguistics, 2009. URL `http://dl.acm.org/citation.cfm?id=1621774.1621784`.

Chew Y. Choong, Yoshiki Mikami, C. A. Marasinghe, and S. T. Nandasara. Order of an n-gram based language identification algorithm for 68 written languages. *The International Journal on Advances in ICT for Emerging Regions*, 2(2):21–28, 2009.

N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. In *IEEE Trans. Audio, Speech, Lang. Process.*, volume 19, May 2011.

Najim Dehak. *Discriminative and Generative Approaches for Long- and Short-term Speaker Characteristics Modeling: Application to Speaker Verification*. PhD thesis, 2009.

L. F. D'Haro, R. Cordoba, M. A. Caraballo, and J. M. Pardo. Low-resource language recognition using a fusion of phoneme posteriorgram counts, acoustic and glottal-based i-vectors. In *2013 IEEE ICASSP*, May 2013.

Mark Dingemanse, Francisco Torreira, and N. J. Enfield. Is "huh?" a universal word? Conversational infrastructure and the convergent evolution of linguistic items. *PLOS ONE*, 8(11):1–10, Nov. 2013. doi: 10.1371/journal.pone.0078273.

David M. Eberhard, Gary F. Simons, and Charles D. Fennig. Ethnologue: Languages of the world. twenty-second edition., 2019. URL `https://www.ethnologue.com`.

ELRA. European Language Resources Association, 2019. URL `http://portal.elda.org/en/`.

A. Ghoshal, P. Swietojanski, and S. Renals. Multilingual training of deep neural networks. In *2013 IEEE ICASSP*, pages 7319–7323, May 2013.

David Martínez González, Oldrich Plchot, Lukás Burget, Ondrej Glembek, and Pavel Matejka. Language recognition in iVectors space. In *INTERSPEECH*, 2011.

Alicia Lozano-Diez; Oldrich Plchot; Pavel Matejka; Joaquin Gonzalez-Rodriguez. DNN based embeddings for language recognition. 2018. URL `http://sigport.org/2399`.

Google. Google translate, 2019. URL `https://translate.google.com/`.

Timothy J. Hazen and Victor W. Zue. Segment-based automatic language identification. *The Journal of the Acoustical Society of America*, 101(4):2323–2331, 1997. doi: 10.1121/1.418211.

Hynek Hermansky. Perceptual linear predictive (PLP) analysis of speech. *The Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990. doi: 10.1121/1.399423.

Tommi Jauhiainen, Marco Lui, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. Automatic language identification in texts: A survey. *CoRR*, abs/1804.08186, 2018. URL `http://arxiv.org/abs/1804.08186`.

P. Kenny. Joint factor analysis of speaker and session variability: Theory and algorithms. Technical report, Jan. 2006.

P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel. Joint factor analysis versus eigenchannels in speaker recognition. In *IEEE Trans. Audio, Speech, Lang. Process.*, volume 15, pages 1435–1447, May 2007.

T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur. A study on data augmentation of reverberant speech for robust speech recognition. In *2017 IEEE ICASSP*, pages 5220–5224, Mar. 2017. doi: 10.1109/ICASSP.2017.7953152.

Pawan Kumar, Astik Biswas, Achyuta Nand Mishra, and Mahesh Chandra. Spoken language identification using hybrid feature extraction methods. *CoRR*, abs/1003.5623, 2010. URL `http://arxiv.org/abs/1003.5623`.

HingKeung Kwan and Keikichi Hirose. Use of recurrent network for unknown language rejection in language identification system. *EUROSPEECH-1997*, pages 63–66, 1997.

Joachim Khler. Multilingual phone models for vocabulary-independent speech recognition tasks. *Speech Communication*, 35(1):21 – 30, 2001. ISSN 0167-6393. doi: https://doi.org/10.1016/S0167-6393(00)00093-5. MIST.

Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren. A novel scheme for speaker recognition using a phonetically-aware deep neural network. In *2014 IEEE ICASSP*, pages 1695–1699. IEEE, 2014.

Haizhou Li and Bin Ma. A phonotactic language model for spoken language identification. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 515–522. Association for Computational Linguistics, 2005. doi: 10.3115/1219840.1219904.

Haizhou Li, Bin Ma, and Kong Aik Lee. Spoken language recognition: From fundamentals to practice. In *Proceedings of the IEEE*, volume 101, pages 1136–1159, May 2013.

Yitong Li, Timothy Baldwin, and Trevor Cohn. Whats in a domain? Learning domain-robust text representations using adversarial training. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018. doi: 10.18653/v1/n18-2076.

Jesus Antonio Villalba Lopez, Niko Brummer, and Najim Dehak. End-to-end versus embedding neural networks for language recognition in mismatched conditions. In *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pages 112–119, 2018. doi: 10.21437/Odyssey.2018-16.

J. C. Marcadet, Volker Fischer, and Claire Waast-Richard. A transformation-based learning approach to language identification for mixed-lingual text-to-speech synthesis. In *INTERSPEECH 2005 - Eurospeech, 9th European Conference on Speech Communication and Technology, Lisbon, Portugal, Sept. 4-8, 2005*, pages 2249–2252, 2005. URL `http://www.isca-speech.org/archive/interspeech_2005/i05_2249.html`.

Anthony McEnery, Paul Baker, Rob Gaizauskas, Hamish Cunningham, and Andrew Hardie. The EMILLE Project. URL `http://www.emille.lancs.ac.uk/`.

Anthony McEnery, Paul Baker, Rob Gaizauskas, and Hamish Cunningham. EMILLE: Building a corpus of South Asian languages. 2000.

Mitchell Mclaren, Mahesh Kumar Nandwana, Diego Castn, and Luciana Ferrer. Approaches to multi-domain language recognition. In *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pages 90–97, 2018. doi: 10.21437/Odyssey.2018-13.

Avashlin Moodley. *Language Identification With Decision Trees: Identification Of Individual Words In The South African Languages.* PhD thesis, Jan. 2016.

Yeshwant K. Muthusamya and A. Lawrence Spitz. Automatic language identification. In *Survey of the state of the art in human language technology*, volume 13. Cambridge University Press, 1997.

Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

NIST. NIST 2017 Language Recognition Evaluation Plan, May 2017. URL `https://www.nist.gov/sites/default/files/documents/2017/06/01/lre17_eval_plan-2017-05-31_v2.pdf`.

Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *INTERSPEECH 2015*, 2015.

Steven T Piantadosi. Zipf's word frequency law in natural language: A critical review and future directions. *Psychon Bull Rev*, 21:1112–1130, Oct. 2014. doi: 10.3758/s13423-014-0585-6.

Daniel Povey. Room Impulse Response and Noise Database, 2017. URL `http://www.openslr.org/28/`.

Daniel Povey and Daniel Galvez. GlobalPhone scripts, 2017. URL `https://github.com/kaldi-asr/kaldi/tree/master/egs/gp`.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011.

Morton D Rau. Language identification by statistical analysis. Technical report, Naval Postgraduate School, Monterey, CA, 1974.

Steve Renals. Speaker verification lecture slides, Mar. 2019. URL `http://www.inf.ed.ac.uk/teaching/courses/asr/2018-19/asr17-speaker.pdf`.

D. Reynolds, T. Quatieri, and R. Dunn. Speaker verification using adapted Gaussian mixture models. In *Digital Signal Processing*, volume 10, pages 19–41. Academic Press, 2000.

D. A. Reynolds and R. C. Rose. Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3(1):72–83, Jan 1995. ISSN 1063-6676.

Douglas A Reynolds. *A Gaussian mixture modeling approach to text-independent speaker identification.* PhD thesis, 1993.

P. Roach, S. Arnfield, W. Barry, J. Baltova, M. Boldea, A. Fourcin, W. Gonet, R. Gubrynowicz, E. Hallum, L. Lamel, K. Marasek, A. Mar.al, E. Meister, and K. Vicsi. BABEL: an Eastern European multi-language database. In *Proceedings of the Fourth International Conference on Spoken Language Processing. ICSLP '96*, volume 3, pages 1892–1893, Oct. 1996. doi: 10.1109/ICSLP.1996.608002.

Hossein Salehghaffari. Speaker verification using convolutional neural networks. *arXiv preprint arXiv:1803.05427*, 2018.

Mousmita Sarma, Kandarpa Kumar Sarma, and Nagendra Kumar Goel. Language recognition using time delay deep neural network, 2018.

Tanja Schultz. GlobalPhone: A Multilingual Speech and Text Database Developed at Karlsruhe University. Technical report, Interactive Systems Laboratories, Karlsruhe University, Carnegie Mellon University, 2002.

Suwon Shon, Ahmed Ali, and James Glass. Convolutional neural network and language embeddings for end-to-end dialect recognition. In *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pages 98–104, 2018. doi: 10.21437/Odyssey.2018-14.

David Snyder, Guoguo Chen, and Daniel Povey. MUSAN: A Music, Speech, and Noise Corpus, 2015. URL `http://www.openslr.org/17/`. arXiv:1510.08484v1.

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. NIST SRE 2016 example scripts, 2016. URL `https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16`.

David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur. Deep neural network embeddings for text-independent speaker verification. In *Conference: Interspeech 2017*, pages 999–1003, Aug. 2017. doi: 10.21437/Interspeech.2017-620.

David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. Spoken Language Recognition using X-vectors. pages 105–111, June 2018a. doi: 10.21437/Odyssey.2018-15.

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. *2018 IEEE ICASSP*, pages 5329–5333, Apr. 2018b. doi: 10.1109/ICASSP.2018.8461375.

P. Swietojanski, A. Ghoshal, and S. Renals. Unsupervised cross-lingual knowledge transfer in DNN-based LVCSR. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 246–251, Dec 2012. doi: 10.1109/SLT.2012.6424230.

David Synder. Kaldi's logistic regression source code, 2014. URL `http://kaldi-asr.org/doc/logistic-regression_8cc_source.html`.

David Synder and Daniel Povey. Logistic regression script, 2014. URL `https://github.com/kaldi-asr/kaldi/blob/master/egs/lre07/v1/lid/run_logistic_regression.sh`.

R. Tong, B. Ma, H. Li, and E. S. Chng. A target-oriented phonotactic front-end for spoken language recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(7):1335–1347, Sept. 2009. ISSN 1558-7916. doi: 10.1109/TASL.2009.2016731.

Rong Tong, Bin Ma, Donglai Zhu, Haizhou Li, and Eng Siong Chng. Integrating acoustic, prosodic and phonotactic features for spoken language identification. In

*ICASSP, 2006 IEEE International Conference*. IEEE, May 2006. doi: 10.1109/ICASSP.2006.1659993.

P. A. Torres-Carrasquillo, D. A. Reynolds, and R. J. Deller Jr. Language identification using Gaussian mixture model tokenization. In *ICASSP, 2002 IEEE International Conference*, pages 757–760. IEEE, 2002.

László Tóth. Phone recognition with hierarchical convolutional deep maxout networks. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1):25, Sep 2015. ISSN 1687-4722. doi: 10.1186/s13636-015-0068-3.

E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In *2014 IEEE ICASSP*, pages 4052–4056, May 2014. doi: 10.1109/ICASSP.2014.6854363.

K. Vesel, M. Karafit, F. Grzl, M. Janda, and E. Egorova. The language-independent bottleneck features. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 336–341, Dec 2012. doi: 10.1109/SLT.2012.6424246.

Alex Waibel, T. Hanazawa, Geoffrey E. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. Technical Report TR-I-0006, Advanced Telecommunication Research Institute, International Interpreting Telephony Research Laboratories, 1989.

E. Wong and S. Sridharan. Comparison of linear prediction cepstrum coefficients and mel-frequency cepstrum coefficients for language identification. In *Proceedings of ISIMP 2001*, pages 95–98, May 2001. doi: 10.1109/ISIMP.2001.925340.

Aonan Zhang, Quan Wang, Zhenyao Zhu, John Paisley, and Chong Wang. Fully supervised speaker diarization. *arXiv preprint arXiv:1810.04719*, 2018.

Fang Zheng, Guoliang Zhang, and Zhanjiang Song. Comparison of different implementations of MFCC. *Journal of Computer Science and Technology*, 16(6):582–589, Nov 2001. ISSN 1860-4749. doi: 10.1007/BF02943243.

# Appendix A

# GlobalPhone dataset splits

| Language | Length (hrs) | No. speakers | Age (mean) | Gender | |
|---|---|---|---|---|---|
| | | | | (m/f) | Ratio |
| Arabic | 14.5 | 57 | 27.2±12.5 | 27/30 | 47:53 |
| Bulgarian | 15.1 | 56 | 33.3±14.2 | 27/29 | 48:52 |
| Croatian | 10.5 | 63 | 35.3±15.5* | 22/41 | 36:64 |
| Czech | 23.8 | 72 | 26.6±11.3 | 41/31 | 57:43 |
| French | 20.1 | 74 | 32.2±12.0* | 36/38 | 49:51 |
| German | 13.2 | 58 | - | 56/2 | 97:3 |
| Japanese | 29.2 | 119 | 23.3±8.1* | 87/31* | 74:26 |
| Korean | 14.6 | 70 | 25.6±4.9* | 35/35 | 50:50 |
| Mandarin | 23.7 | 98 | 23.4±5.7 | 50/48 | 51:49 |
| Polish | 17.0 | 69 | 35.4±9.7** | 9/8** | - |
| Portuguese | 21.0 | 76 | 29.6±9.0* | 38/38 | 50:50 |
| Russian | 17.3 | 84 | 27.1±9.7 | 48/36 | 57:43 |
| Shanghainese | 1.0 | 6 | 41.3±6.8 | 4/2 | 67:33 |
| Spanish | 16.0 | 72 | 28.2±12.3 | 31/41 | 43:57 |
| Swedish | 15.4 | 70 | 31.6±13.7 | 34/36 | 49:51 |
| Thai | 22.0 | 73 | 20.2±1.5 | 21/48* | 28:72 |
| Turkish | 11.6 | 69 | 27.5±10.9 | 20/49 | 30:70 |
| Vietnamese | 15.8 | 103 | 29.9±11.7 | 56/45 | 55:45 |
| Total | 301.8 | 1289 | - | - | - |
| Mean (language) | 16.8±6.0 | 71.6±22.5 | - | - | - |

Table A.1: Summary of training GlobalPhone data as used in experiments. For mean data, the ± value is the population standard deviation. Gender ratios are rounded to whole numbers. * indicates that there are only a few (1-4) speakers without the listed information. ** indicates that there were many speakers missing it

| Language | Length (hrs) | No. speakers | Age (mean) | Gender | |
|---|---|---|---|---|---|
| | | | | (m/f) | Ratio |
| Arabic | 1.4 | 7 | 26.0±6.7 | 3/4 | 43:57 |
| Bulgarian | 2.0 | 7 | 33.0±12.9 | 0/7 | 0:100 |
| Croatian | 1.6 | 9 | 25.2±3.8 | 6/3 | 67:33 |
| Czech | 3.1 | 10 | 21.2±2.5 | 6/4 | 60:40 |
| French | 2.7 | 10 | 35.6±13.6 | 5/5 | 50:50 |
| German | 1.7 | 7 | - | 6/1 | 86:14 |
| Japanese | 3.0 | 14 | 22.2±4.1 | 7/7 | 50:50 |
| Korean | 2.2 | 10 | 26.7±6.1 | 5/5 | 50:50 |
| Mandarin | 3.1 | 13 | 25.7±5.0 | 3/10 | 23:77 |
| Polish | 2.3 | 9 | 39.0±14.0* | 2/0* | - |
| Portuguese | 1.9 | 10 | 24.2±6.5 | 7/3 | 70:30 |
| Russian | 2.5 | 11 | 26.6±10.5 | 3/8 | 27:73 |
| Shanghainese | 0.8 | 4 | 38.3±7.6 | 2/2 | 50:50 |
| Spanish | 2.3 | 10 | 29.9±10.9 | 4/6 | 40:60 |
| Swedish | 2.0 | 9 | 33.6±11.2 | 6/3 | 67:33 |
| Thai | 2.9 | 9 | 20.8±1.6 | 2/7 | 22:78 |
| Turkish | 1.7 | 10 | 29.9±12.0 | 2/8 | 20:80 |
| Vietnamese | 1.7 | 12 | 32.1±12.5 | 8/4 | 67:33 |
| Total | 38.9 | 171 | - | - | - |
| Mean (language) | 2.2±0.4 | 9.5±2.3 | - | - | - |

Table A.2: Summary of enrollment GlobalPhone data as used in experiments. For mean data, the ± value is the population standard deviation. Gender ratios are rounded to whole numbers. * indicates that there were many speakers missing the listed information

| Language | Length (hrs) | No. speakers | Age (mean) | Gender | |
|---|---|---|---|---|---|
| | | | | (m/f) | Ratio |
| Arabic | 1.7 | 7 | 23.1±4.7 | 3/4 | 43:57 |
| Bulgarian | 2.3 | 7 | 36.1±16.4 | 3/4 | 43:57 |
| Croatian | 2.0 | 10 | 25.9±9.0 | 5/5 | 50:50 |
| Czech | 2.4 | 10 | 21.6±7.6 | 5/5 | 50:50 |
| French | 2.1 | 8 | 32.9±11.2 | 4/4 | 50:50 |
| German | 2.0 | 6 | - | 4/2 | 67:33 |
| Japanese | 0.9 | 5 | 23.2±2.9 | 3/2 | 60:40 |
| Korean | 2.2 | 10 | 28.1±7.9 | 5/5 | 50:50 |
| Mandarin | 2.0 | 11 | 21.5±1.9 | 6/5 | 55:45 |
| Polish | 2.8 | 10 | 22.0±0.8** | 1/2** | - |
| Portuguese | 1.6 | 8 | 28.8±16.2 | 4/4 | 50:50 |
| Russian | 2.5 | 10 | 28.4±11.1 | 6/4 | 60:40 |
| Shanghainese | 0.7 | 4 | 36.8±4.3 | 1/3 | 25:75 |
| Spanish | 2.1 | 10 | 21.7±2.0 | 5/5 | 50:50 |
| Swedish | 2.1 | 9 | 33.2±13.2 | 5/4 | 56:44 |
| Thai | 2.3 | 8 | 20.4±1.4 | 2/4* | 33:67 |
| Turkish | 2.0 | 11 | 24.4±6.2 | 3/8 | 27:73 |
| Vietnamese | 1.4 | 9 | 27.7±6.1 | 6/3 | 67:33 |
| Total | 35.1 | 153 | - | - | - |
| Mean (language) | 2.0±0.5 | 8.5±2.0 | - | - | - |

Table A.3: Summary of evaluation GlobalPhone data as used in experiments. For mean data, the ± value is the population standard deviation. Gender ratios are rounded to whole numbers. * indicates that there are only a few (1-2) speakers without the listed information. ** indicates that there were many speakers missing it

| Language | Length (hrs) | No. speakers | Age (mean) | Gender | |
|---|---|---|---|---|---|
| | | | | (m/f) | Ratio |
| Arabic | 1.4 | 7 | 27.0±9.1 | 2/5 | 29:71 |
| Bulgarian | 2.0 | 7 | 23.14±5.5 | 2/5 | 29:71 |
| Croatian | 1.8 | 10 | 28.1±9.4 | 4/6 | 40:60 |
| Czech | 2.7 | 10 | 27.6±16.1 | 5/5 | 50:50 |
| French | 2.0 | 8 | 32.25±11.4 | 4/4 | 50:50 |
| German | 1.5 | 6 | - | 4/2 | 67:33 |
| Japanese | 0.8 | 6 | 24.83±5.6 | 5/1 | 83:17 |
| Korean | 2.1 | 10 | 25.6±4.3 | 5/5 | 50:50 |
| Mandarin | 2.4 | 10 | 19.9±0.7 | 5/5 | 50:50 |
| Polish | 2.3 | 10 | 35.0±12.2* | 0/4* | - |
| Portuguese | 1.8 | 7 | 25.57±12.8 | 4/3 | 57:43 |
| Russian | 2.4 | 10 | 35.7±10.7 | 4/6 | 40:60 |
| Shanghainese | 1.1 | 4 | 42.0±10.5 | 1/3 | 25:75 |
| Spanish | 1.7 | 8 | 21.88±1.5 | 4/4 | 50:50 |
| Swedish | 2.2 | 10 | 35.0±15.8 | 5/5 | 50:50 |
| Thai | 0.9 | 8 | 23.12±0.8 | 2/6 | 25:75 |
| Turkish | 1.9 | 10 | 28.4±11.8 | 3/7 | 30:70 |
| Vietnamese | 0.8 | 5 | 20.8±2.2 | 3/2 | 60:40 |
| Total | 31.8 | 146 | - | - | - |
| Mean (language) | 1.8±0.6 | 8.1±1.9 | - | - | - |

Table A.4: Summary of test GlobalPhone data as used in experiments. For mean data, the ± value is the population standard deviation. Gender ratios are rounded to whole numbers. * indicates that there were many speakers missing the listed information

# Appendix B

# Additional figures

These are additional graphs for the language adaptation and data augmentation experiments, when either 10 or 3 seconds of evaluation data was used.

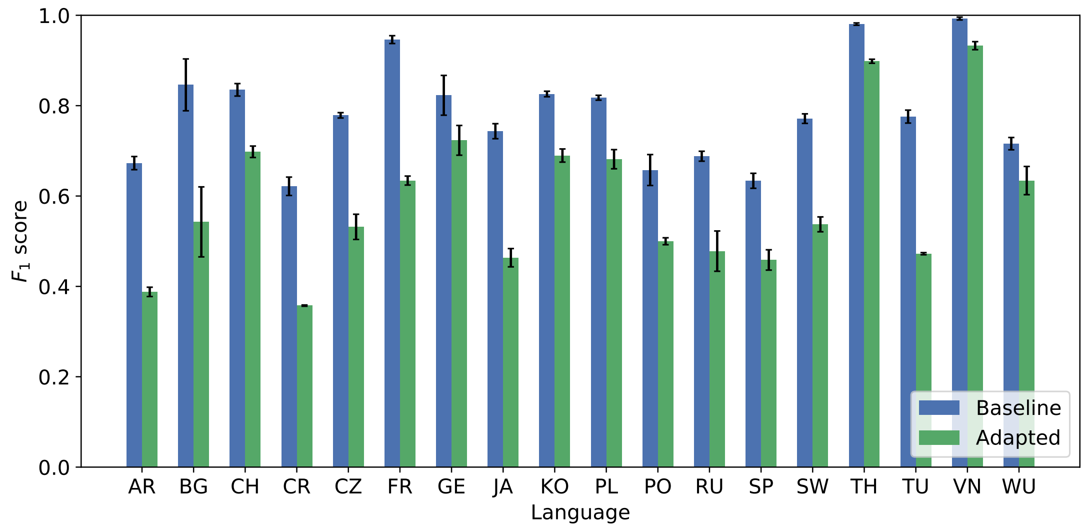


Figure B.1: F1 scores for the adaptation experiment with all the languages in GP. Evaluation lengths were 10 seconds. Blue bars are the baseline F1 scores for each language when the model is trained on all languages. Green bars are the F1 score for each language when it is only included in the enrollment data, while the model is trained on all the other languages

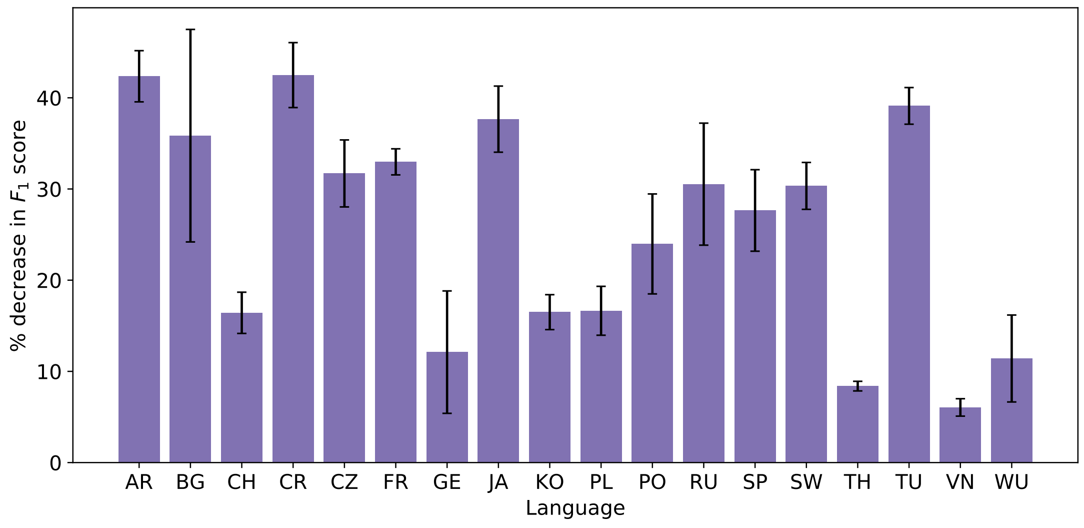


Figure B.2: Percentage decrease in $F_1$ score for each language when not included in the training data, from the results in Figure B.1. Evaluation lengths were 10 seconds. Error bars are the combined error in $F_{baseline}$ and $F_{enroll}$

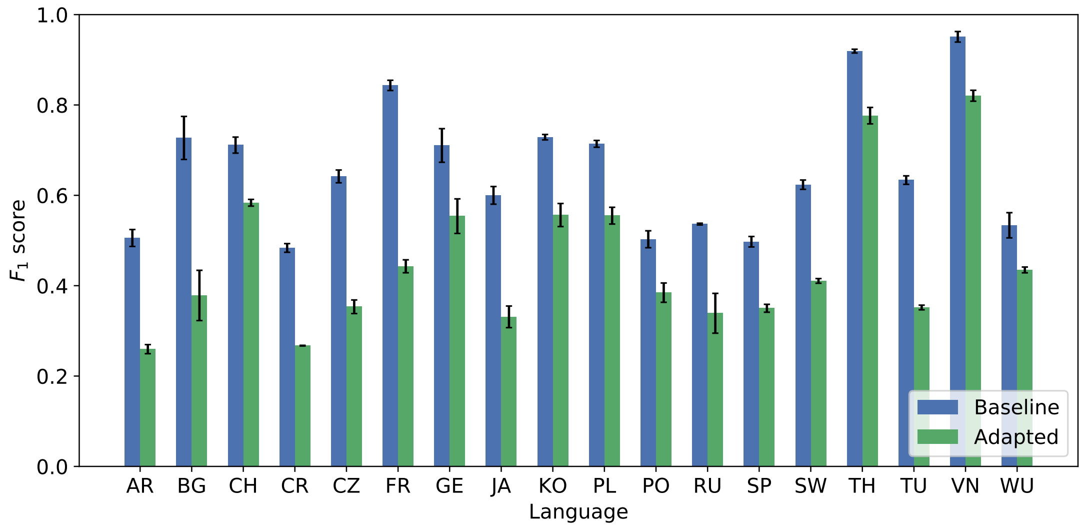


Figure B.3: F1 scores for the adaptation experiment with all the languages in GP. Evaluation lengths were 3 seconds. Blue bars are the baseline F1 scores for each language when the model is trained on all languages. Green bars are the F1 score for each language when it is only included in the enrollment data, while the model is trained on all the other languages

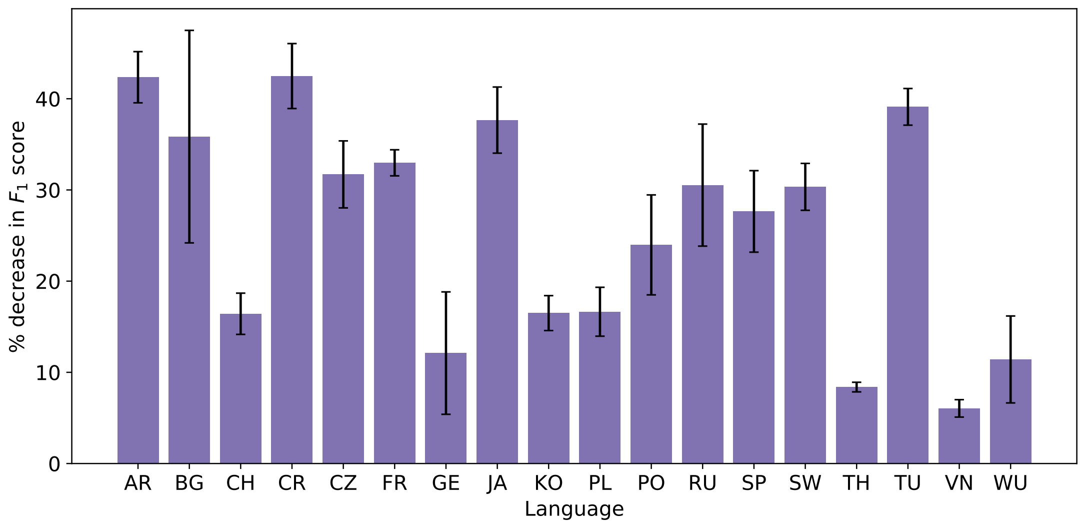


Figure B.4: Percentage decrease in $F_1$ score for each language when not included in the training data, from the results in Figure B.3. Evaluation lengths were 3 seconds. Error bars are the combined error in $F_{baseline}$ and $F_{enroll}$

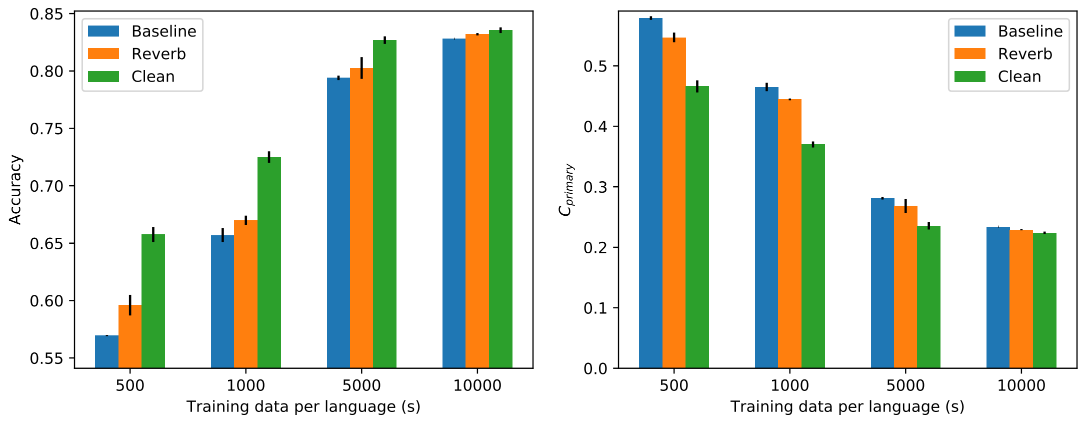


Figure B.5: Comparison of adding reverberated data or the same amount of clean data, versus the baseline model in terms of overall accuracy and $C_{primary}$. The evaluation length was 10 seconds.

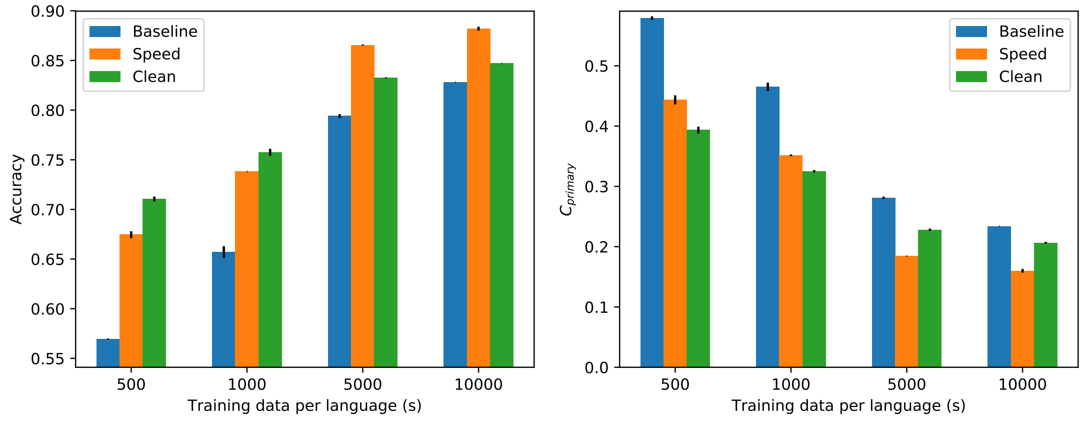


Figure B.6: Comparison of adding speed-augmented data or the same amount of clean data, versus the baseline model in terms of raw accuracy and $C_{primary}$. The evaluation length was 10 seconds.

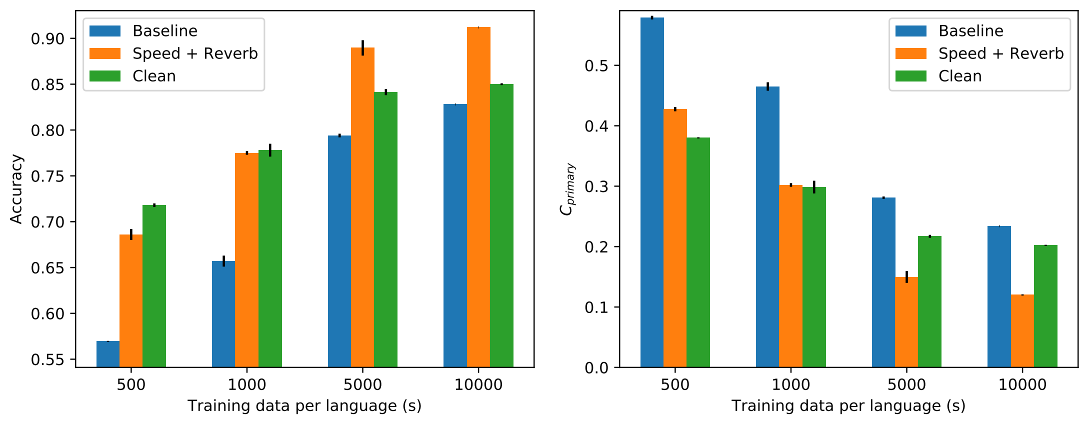


Figure B.7: Comparison of adding reverberated and speed-augmented data, or the same amount of clean data, versus the baseline model in terms of raw accuracy and $C_{primary}$. The evaluation length was 10 seconds.

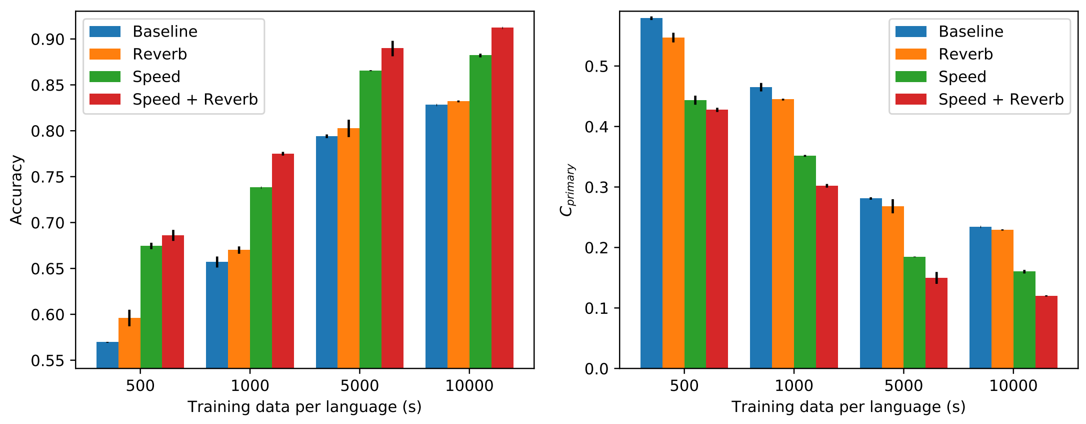


Figure B.8: Comparison of all the augmentation methods in terms of $C_{primary}$ and accuracy. The evaluation length was 10 seconds.

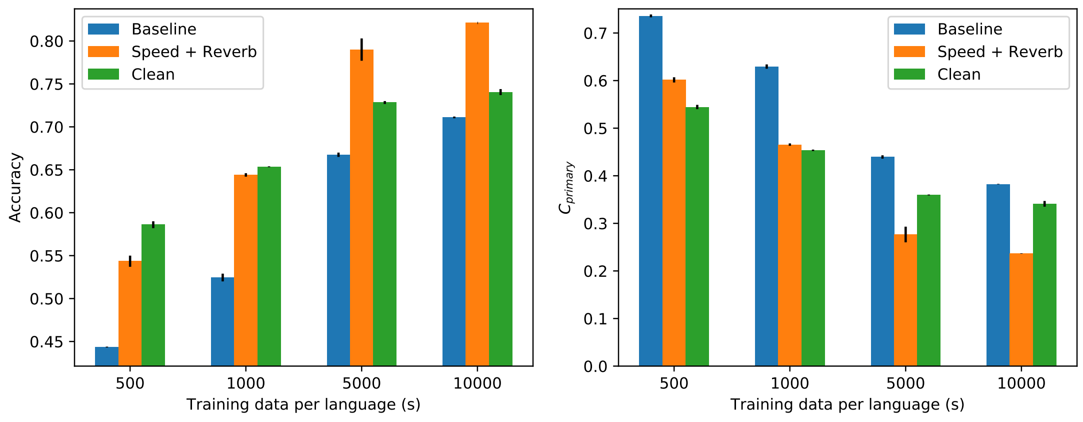


Figure B.9: Comparison of adding reverberated data or the same amount of clean data, versus the baseline model in terms of overall accuracy and $C_{primary}$. The evaluation length was 3 seconds.

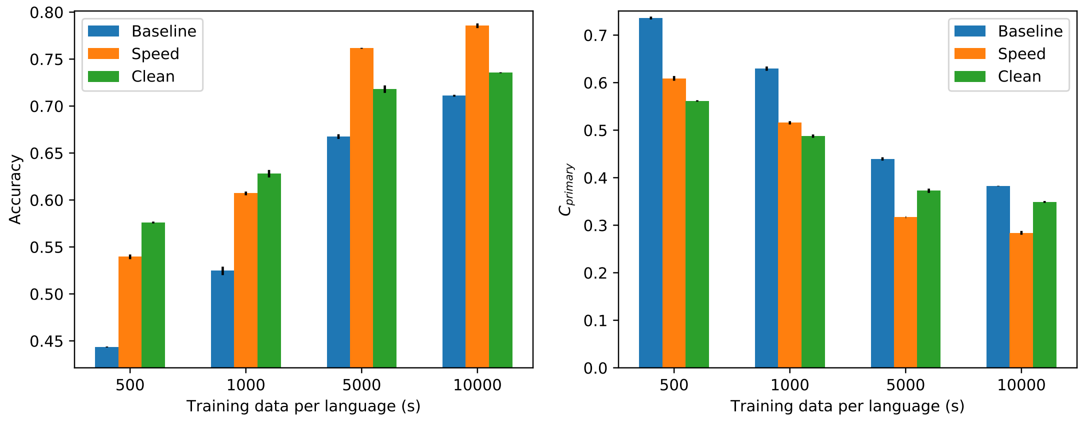


Figure B.10: Comparison of adding speed-augmented data or the same amount of clean data, versus the baseline model in terms of raw accuracy and $C_{primary}$. The evaluation length was 10 seconds.

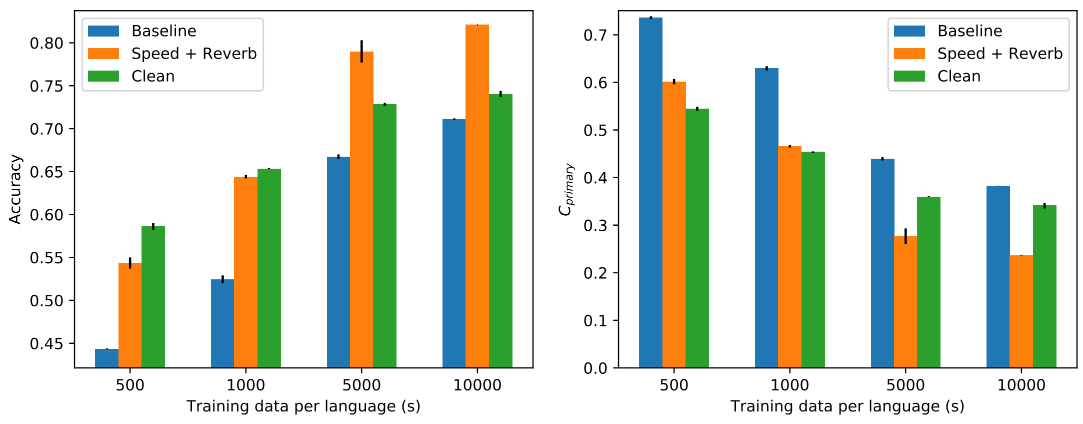


Figure B.11: Comparison of adding reverberated and speed-augmented data, or the same amount of clean data, versus the baseline model in terms of raw accuracy and $C_{primary}$. The evaluation length was 3 seconds.

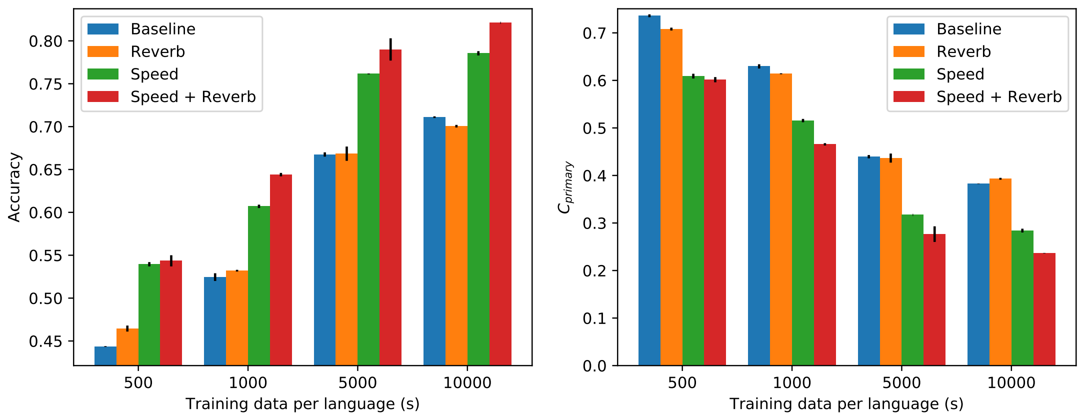


Figure B.12: Comparison of all the augmentation methods in terms of $C_{primary}$ and accuracy. The evaluation length was 3 seconds.