Prosodic features in state-of-the-art spoken language identification

Sam Sucik

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

2019

Abstract

TO-DO

Acknowledgements

Thanks to Paul Moore for productive collaboration while building the baseline system, to Steve Renals for his supervision and optimism, and to David Snyder for his advice.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

LID is useful in ASR, in voice assistants, emergency call routing, etc. Traditionally, acoustic features are used (influence of ASR on LID and SID). Prosodic LID is much rarer, although results show that prosodic information can help identify language (Lin and Wang, 2005), and that both LID and ASR can benefit from using acoustic *and* prosodic features (González et al., 2013; Ghahremani et al., 2014). Just last year, a novel architecture for LID utilising *x-vectors* was proposed by Snyder et al. (2018a), dramatically improving the state-of-the-art results. Although the authors find that using bottleneck features from an ASR DNN yields better results than using the standard acoustic MFCC features, even the ASR DNN was trained just using MFCCs. Thus the work ignores the potential of speech information other than that captured by MFCCs.

1.2 Aims

In this work, I aim to reproduce the state-of-the-art x-vector LID system and explore the use of prosodic features in addition to acoustic ones. Because the system uses a relatively novel architecture, in which a TDNN aggregates information across a speech segment, I also compare two types of acoustic features, one which has such aggregation over time encoded (SDC) and one that only contains information about single frames (vanilla MFCC).

1.3 Contributions

1. Adapted an existing x-vector speaker verification implementation (based on Snyder et al. (2018b)) for language identification

2. Explored the choice of classifiers and chose a different one than Snyder et al. (2018a)
3. Prepared the Global Phone corpus for LID with the x-vector system, extending the original partitioning of the corpus into datasets and analysing invalid data
4. Built and evaluated a baseline, end-to-end x-vector LID system using 19 languages of the Global Phone corpus
5. Explored, set and tuned important hyperparameters of the system, mainly the number of training epochs of the x-vector TDNN
6. Researched literature concerning the use of acoustic and prosodic features in language identification, speaker verification and ASR
7. Designed, run and evaluated experiments comparing two types of acoustic features (MFCC and SDC) and two prosodic features (pitch, energy), and their combinations
8. Built a system which has the potential to be open-sourced as part of the Kaldi ASR toolkit, to be used by a wider community

Chapter 2

Background

This chapter elaborates on the key concepts relevant to my work, as shown in this condensed description of the project: Exploring **spoken language identification** in the context of the recently proposed **x-vector** system (contrasted with the more established state-of-the-art **i-vector** approach, followed by the more novel **d-vector** systems), focusing on utilising **prosodic information** in addition to the standard **acoustic information**.

2.1 The task: spoken language recognition

Spoken language recognition means recognising the language of a speech segment. The task is similar to speaker recognition and, in the past, similar systems have been used for the two tasks. Importantly, recognition is typically realised as one of two different tasks:

- Identification (multi-class classification): answering the question "For a speech segment X, which one (from a number of supported targets) is its target (language or speaker) T_x?"
- Verification (binary classification): "Is T_x the target (language or speaker) of the speech segment X?"

Identification is more suitable for use cases with a small and stable set of possible targets – such as the set of supported languages. There, computing the probability of T_x being each of the target languages is feasible. Verification, on the other hand, is more suitable for cases where the set of possible targets less constrained – such as the large and changing set of possible speakers in speaker verification systems. There, it is often infeasible to compute the probability distribution over all possible values of T_x; instead, the system typically focuses only on evaluating the probability of T_x being the hypothesised speaker. Throughout this work, I focus on *language identificaion* (LID) with a *closed set* of target languages (i.e. not including the option to identify a speech segment's language as unknown/other).

2.2 Shallow utterance-level approach to LID: i-vectors

This approach, with its numerous variations, has now been the state of the art for 8 years – since first introduced by Dehak et al. (2011) for speaker recognition and later applied by Martinez et al. (2011) in language recognition.

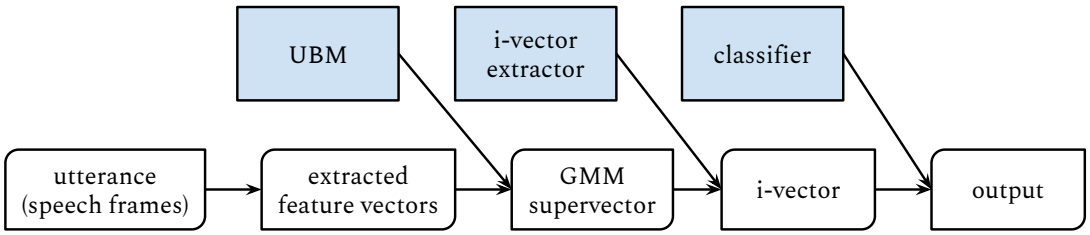

Figure 2.1: Language identification using a typical i-vector system.

The main components of a typical i-vector system, together with their use for prediction, are shown in Fig. 2.1. The universal background model (UBM) is a Gaussian mixture model (GMM) consisting of a large number (e.g. 2048) multivariate mixture components. The UBM is trained using the Expectation-Maximisation algorithm to model the observation space of frame-level feature vectors $\mathbf{x}$ computed from all training utterances (typically 39-dimensional vector using the MFCC features, see Section 2.5.1). Given the trained UBM, an utterance X can be represented by a language- and channel-specific GMM supervector M as:

$$M_X = m + Tw_X \tag{2.1}$$

Here, m is the language- and channel-independent GMM supervector of the UBM (consisting of the stacked means of all mixture components). T is the *total variability matrix* – also called the i-vector extractor, and w_x is the *identity vector* (i-vector) of sample X. The total variability matrix T is "a matrix of bases spanning the subspace covering the important [language and channel] variability in the supervector space" (Martinez et al., p.862). Effectively, T projects between the high-dimensional GMM supervector space and the low-dimensional *total variability subspace* (also called the *i-vector space*). The i-vector w_x is then a low-dimensional set of factors projecting the supervector onto the total variability subspace base vectors. The i-vector extractor T is trained again using Expectation-Maximisation (for details see Mak and Chien (2016, p. 100)). Without providing too much detail, I highlight the aspect that is most relevant to my work: training T is based on calculating and further combining the 0^{th}, 1^{st} and 2^{nd} order statistics which are computed by *summing over the frames* of an utterance.

Once the i-vector extractor has been trained, any utterance can be represented by its i-vector. This enables training a relatively simple and fast classifier operating over the low-dimensional i-vectors. Different classifiers have been successfully used with the i-vector back end; for example, Martinez et al. initially tried using these, all achieving roughly equal performance:

1. a linear generative model – modelling the i-vectors of each language by a Gaussian, with a shared full covariance matrix across all language-modelling Gaussians
2. a linear Support Vector Machine computing scores for all languages by doing binary classification in the one-versus-all fashion
3. a two-class logistic regression also doing one-versus-all classification.

At test time, a sequence of feature vectors for utterance X is projected using the UBM into the high-dimensional *supervector space*, producing M_X. Then, T is used to extract the utterance-level i-vector, which is processed by the classifier.

Despite producing utterance-level scores, I describe the i-vector pipeline as shallow because it aggregates frame-level information over time very early (when projecting X into the supervector space), effectively treating an utterance as a bag of frames and disregarding any temporal patterns spanning over multiple frames.

2.3 Less shallow approach: d-vectors

Introduced for speaker verification by Variani et al. (2014) and later adapted and applied to language identification by Tkachenko et al. (2016), this approach uses neural networks to extract frame-level information, which is then aggregated across frames to produce utterance- or language-level vectors. Importantly, nothing changes about the final classification stage itself: It is the differences in producing the vector representations what makes i-vectors, d-vectors and x-vectors differ from each other.

The biggest change introduced in d-vector systems compared to i-vectors is the notion of frame-level processing while considering each frame's temporal context, i.e. the sequence of a few preceding and following frames. While Variani et al. achieve this by simply feeding the feature vectors of the neighbouring frames together with the frame of interest into a standard deep neural network, Tkachenko et al. use a more suited architecture commonly used for processing temporal sequences in automatic speech recognition: the *time-delay neural network* (TDNN).

A TDNN works like a convolutional neural network (CNN) with 1-dimensional convolutions: only along the time axis, as opposed to the more common 2-dimensional convolutions in image CNNs. Fig. 2.2 shows a TDNN with three layers which processes information over 9-frame contexts. Note that each blue circle from any layer corresponds to the layer itself (a collection of neurons, i.e. *convolutional kernels*), and all blue circles drawn above each other as corresponding to a particular layer are in fact just one circle – the layer itself – sliding over multiple inputs. For example, the convolutional kernels from layer 1 are first applied to feature vectors $\mathbf{x_1}$-$\mathbf{x_5}$, then to vectors $\mathbf{x_3}$-$\mathbf{x_7}$ and then to $\mathbf{x_5}$-$\mathbf{x_9}$. Notice how the network is made more sparse by using *subsampling*, i.e. not using the connections drawn in light grey. A concise and commonly used description of the sketched architecture is shown in Tab. 2.1 (notice the difference between the interval and set notation); "layer context" meant relative to the preceding layer.

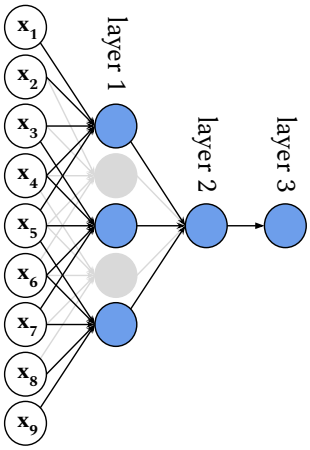

Figure 2.2: Example of a time-delay neural network.

	LAYER CONTEXT	TOTAL CONTEXT
LAYER 1	$[t-2,\ t+2]$	5
LAYER 2	$\{t-2,\ t,\ t+2\}$	9
LAYER 3	$\{t\}$	9

Table 2.1: Example of architecture description of a TDNN (corresponds to Fig. 2.2).

In a d-vector system, a TDNN is trained to do direct frame-wise language identification; for this purpose, an additional softmax layer can be added to produce classification outputs. Often, between the convolutional layers and the output layer the architecture contains a few fully connected layers. After training, the classification layer is disregarded and a frame is instead represented by the activation values from the last hidden layer. The TDNN thus serves as a feature extractor, and the representations produced are referred to as *embeddings*. A d-vector representing an entire utterance is then simply the average of the embeddings of all frames from the given utterance.

Despite the naive averaging, utterances are no longer treated merely as bags of frames because each embeddings contains features extracted over short temporal window: 21-frame windows in the case of Tkachenko et al. and 41-frame windows used by Variani et al., making d-vectors less shallow than i-vectors.

2.4 Deep utterance-level approach: x-vectors

The x-vector approach, introduced last year by the John Hopkins University team first for speaker verification (Snyder et al., 2018b) and subsequently for language identification (Snyder et al., 2018a), can be viewed as an extension to d-vectors, producing utterance-level embeddings even for variable-length speech segments. The architecture consists of (see also Fig. 2.3):

1. the context-aggregating TDNN layers operating at frame level (with the final context window of ± 7 frames),

2. a *statistics pooling layer* which computes the mean and the standard deviation over all frames, effectively changing a variable-length sequence of frame-level activations into a fix-length vector, and

3. an utterance-level part consisting of 2 fully connected bottleneck layers which extract more sophisticated features and compress the information into a lower-dimensional space, and an additional softmax output layer.

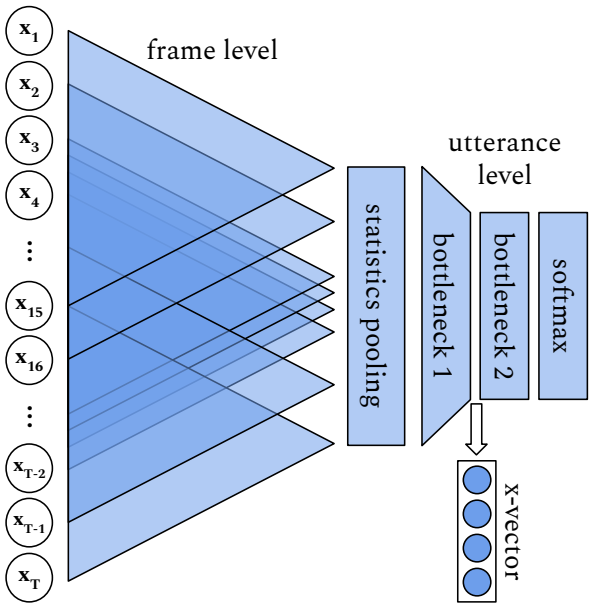

Figure 2.3: A high-level sketch of the x-vector TDNN from Snyder et al..

After the TDNN has been trained, embeddings extracted as the activations of the first bottleneck layer – named x-vectors – are then utterance-level representations and can be used directly as input for a final classifier. The biggest difference compared to the d-vector neural networks discussed in Section 2.3 is that the x-vector TDNN is trained to do not frame-level, but utterance-level classification. The utterance-level statistics (mean and standard deviation) are computed *as part of the network* and are further processed by the additional fully connected layers with non-linear activation functions. These bottleneck layers also make it possible to preserve a higher number of useful features after the statistics-pooling layer: By enabling the last frame-level layer to inflate the representations to 1500-D. The bottleneck layers then learn a transform back to the 512-dimensional space (high-dimensional vectors would be inconvenient for the final classifier) while extracting from the aggregated statistics the features most useful for utterance-level classification (not just for frame-level decisions).

Because of being an utterance-level classifier, the x-vector TDNN can be used directly as an end-to-end system, although Snyder et al. found that extracting x-vectors and training a separate classifier to classify them yielded lightly better results. Perhaps an even stronger argument in favour of the two-stage process is that, by using an external light-weight classifier, one can easily change the set of supported languages without having to expensively re-train the TDNN. Snyder et al. showed that simply re-training the classifier is enough to gain very good results for languages the TDNN has not

		Layer context	Total context	Number of units
frame level	layer 1	$[t-2, t+2]$	5	512
	layer 2	$\{t-2, t, t+2\}$	9	512
	layer 3	$\{t-3, t, t+3\}$	15	512
	layer 4	$\{t\}$	15	512
	layer 5	$\{t\}$	15	1500
statistics pooling layer		$[1, T]$	utterance	3000-dimensional output
utter. level	bottleneck 1	N/A	utterance	512
	bottleneck 2	N/A	utterance	512
	softmax	N/A	utterance	L-dimensional output

Table 2.2: Description of the x-vector TDNN from Snyder et al. and used in this work. T is the number of frames in a given utterance. Table adapted from Snyder et al. (2018a, p. 106).

observed during training (although the results are indeed worse than those for observed languages).

The x-vector system consistently beat several state-of-the-art i-vector architectures, which is a particularly interesting finding because i-vector systems have for long been dominant despite the nowadays so ”fashionable” and successful deep learning approaches. Even so, the 2-stage x-vector system using a simple classifier still dominates the end-to-end neural network alternative.

2.5 Features used in language identification

Historically, automatic speech recognition (ASR) has been the most important area of speech processing and it has driven forward other areas including language and speaker recognition. In particular, LID systems still exist mostly as part of ASR systems where the language of speech needs to be identified before attempting to transcribe the speech. Hence, data preprocessing and input feature types for LID systems often come from the ASR area. In particular, the feature extraction processes follow the ASR objectives: To extract the language- and speaker-*independent* information important for discriminating between different phonemes produced by a speaker's vocal tract. The feature types devised for this aim are termed *acoustic features*. However, these features can also be characterised by the information they disregard or at least to not capture explicitly: the language- and/or speaker-*dependent* information: intonation, stress, pitch range, tone, and others, collectively referred to as *prosodic information*.

As this work focuses on exploiting prosodic information to improve LID, it is desirable to understand both acoustic and prosodic features: how they differ, why is prosody

useful and how can prosodic information be modelled in systems that process frame sequences rather than isolated speech frames.

2.5.1 Acoustic features

The most commonly used acoustic feature type are the Mel-frequency cepstral coefficients (MFCCs). These are coefficients that aim to characterise the shape of the vocal tract, which corresponds to characterising the spectral envelope of produced sounds or, in a way, to the actual phones produced. Without providing too much technical detail, I remind the reader of the main steps in computing MFCCs in Fig. 2.4 (for details, see for example Chapter 10 of Holmes and Holmes (2002)).

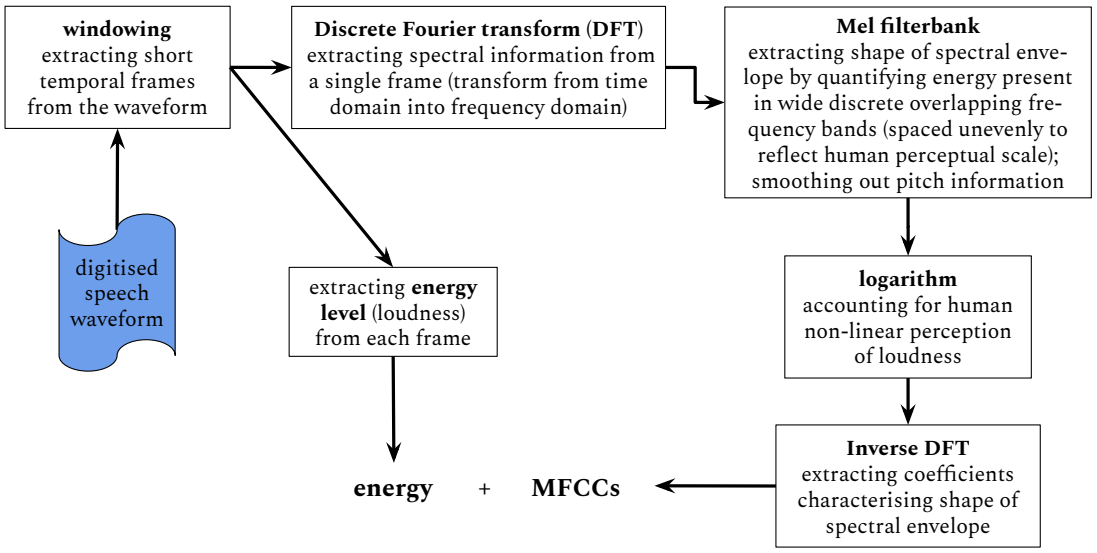

Figure 2.4: Main steps in computing the MFCC features including energy. Adapted from Shimodaira and Renals (2019, p. 10).

To capture local temporal trends going beyond the single frame level, computed features such as MFCCs are often augmented with frame-specific *dynamic* features – *delta cepstra* (also just *deltas* or Δ) and double deltas ($\Delta\Delta$). These are simple approximations of the first and second derivatives of the original acoustic feature, and are computed for frame t like this ($\mathbf{x}_t$ is the MFCC feature vector):

$$\Delta(t) = \mathbf{x}_{t+1} - \mathbf{x}_{t-1}, \quad \Delta\Delta(t) = \Delta(t+1) - \Delta(t-1) \tag{2.2}$$

using the context of ± 2 frames. Such dynamic features are justified especially in systems which apply the bag-of-frames approach, not extracting temporal trends (such as the standard i-vector architectures).

For modelling even longer-range trends specifically in the LID task, Torres-Carrasquillo et al. (2002) used an extension of simple deltas called the *shifted delta cepstra* (SDC) features – essentially, deltas computed at multiple points and concatenated with the

original MFCC vector as previously. SDCs can be configured by setting their 4 parameters: N, d, p, k corresponding to a N-d-p-k, a typical one being 7-1-3-7. N refers to the number of cepstral coefficients taken into account (not necessarily the full MFCC vector), p denotes the distance between consecutive points at which deltas are computed, d determines the size of the window for computing deltas ($d = 1$ corresponds to Eq. 2.2) and k is the number of points at which to compute deltas. The SDCs added to the original MFCC vector are then of the form:

$$\Delta_{SDC}(t+ip) = \mathbf{x}_{t+ip+d} - \mathbf{x}_{t+ip-d} \qquad \forall i \in \{0,...,k-1\} \tag{2.3}$$

As with standard deltas, capturing long-range trends with SDCs can certainly be beneficial for i-vector systems. Ferrer et al. (2016) used an i-vector system with SDC features as the state-of-the-art baseline system for their novel experiments. Additionally – following the successful application of SDCs by Torres-Carrasquillo et al. in a pre-i-vector system, Sarma et al. (2018) report slightly better results with SDCs compared to using high-resolution MFCCs when training an i-vector architecture with a deep neural network UBM. Still, the usefulness of SDCs is not so clear in architectures inherently able to extract temporal trends.

Compared to the prosodic features described next, the discussed classic acoustic features have one clear advantage where the LID system is part of a bigger ASR pipelines: The two systems can share the same feature vectors without the need to compute additional features solely for the purpose of language identification.

2.5.2 Prosodic features

Prosody is about information other than that directly describing the produced phones; the standard elements of prosody are: intonation/tone, stress and rhythm (Prieto and Roseano, 2018). As discussed below, importantly, each of these elements can be used to differentiate between languages.

- *Intonation* and *tone* are perceived qualities typically associated with the variation of the physical quality called pitch, i.e. the fundamental frequency of vibration of the vocal folds (F_0). The term 'tone' is used where pitch variation is contrastive (i.e. changes to it discriminate between different words): in the so-called *tonal languages*, such as Chinese, Vietnamese and Thai. In each of these languages, a small 'alphabet' of distinct tone levels or tone contours can be observed as it conveys meaning. Intonation, on the other hand, denotes the non-contrastive pitch variation mostly associated with the speaker's emotions, language, dialect, social background, speaking style, and other factors (see, for example, the study by Grabe (2004) on the intonational differences between different British English dialects).
- *Stress* (also termed *accent*), despite being a somewhat ambiguous term, generally refers to the relative emphasis on certain syllables within a words or words within a sentence, realised as a variation in loudness (*dynamic accent*), pitch (*pitch accent*) or vowel length (*quantitative accent*). Specific *lexical* (intra-word, i.e

syllable-level) stress patterns or rules are often characteristic of a language. For instance, the within-word location of stress is fixed but different for Czech and Polish (*fixed stress*, see Goedemans and van der Hulst (2013a)), or describable by a set of rules in Arabic (*rule-based stress*, see Goedemans and van der Hulst (2013b)).

- *Rhythm* refers to regularity in sub-word unit lengths, with the most recognised theory of *isochrony* categorising languages into three types based on their rhythm: *syllable-timed* where all syllables have roughly equal durations (e.g. French or Turkish), *mora-timed* where all moras[1] have equal durations (e.g. Slovak or Japanese), and *stress-timed* languages, e.g. German and Arabic (here, the intervals between stressed syllables are of equal durations).

Before turning to bespoke alternative features for modelling prosody, it should be noted that some prosodic aspects can be captured by acoustic features like MFCCs. Because prosody surfaces as temporal variation, systems which extract information from frame sequences (such as TDNN-based systems) have the potential to capture it. MFCCs can be (and often are) used together with energy (see Fig. 2.4), which enables capturing dynamic accent. Even without using energy as part of MFCCs, the quantitative accent should be possible to capture because durations of vowels are preserved as a result of windowing extracting the frames at evenly spaced points of a speech waveform. Regarding rhythm, the duration regularity (isochrony) should be possible to capture for syllable-timed and mora-timed languages because syllable boundaries occur at phoneme boundaries, which are captured by acoustic features. For stress-timed languages, however, the units with equal durations are marked by stressed syllables, meaning that this regularity may not be fully captured – in particular if an energy measure is not included in MFCCs, or if the stress is realised as a variation in pitch rather than in loudness. Considering the prosodic aspects that acoustic features alone *could* theoretically capture, I conclude that the most important prosodic feature that can be deemed complementary to acoustic features and added to the input is pitch: to enable modelling pitch accent, tone, and language-specific intonation.

Perhaps unsurprisingly, extracting information from pitch for LID has been explored in the past – although the number of studies is very small compared to those using the conventional acoustic features. Long before i-vectors, Lin and Wang (2005) attempted to do language identification solely from the pitch contour. This was segmented into syllable-like units (around 100-200ms), each contour segment subsequently represented as the first few coefficients of its approximation by Legendre polynomials. Even though the results were far from those achieved by today's systems, it was demonstrated that LID can be done on pitch information alone. Later, Metze et al. (2013) showed improvements in ASR on tonal (and partly on non-tonal) languages when augmenting MFCCs with custom-designed F_0 variation features. Nevertheless, one disadvantage of using pitch persisted: The fact that F_0 is typically considered to be only present in voiced sounds and undefined for unvoiced ones – making a pitch contour based on F_0 discontinuous. This inconvenience has been recently addressed by Ghahre-

[1] Mora is a basic timing unit; a long syllable having two moras and a short syllable having one mora. See Crystal (2008, p. 312) for a broader definition and discussion.

mani et al. (2014) who developed a pitch-tracking algorithm that approximates pitch values even for unvoiced regions and produces a continuous pitch contour. Adding this feature to MFCCs, the authors report improved ASR performance mainly on tonal languages. To the best of my knowledge, this feature type has not yet been used for language identification.

In this work, I explore two feature types that explicitly help to capture prosodic information: *energy* and *pitch*. I refer to these as the prosodic features.

- By energy I mean the logarithm[2] of the raw energy extracted in the MFCC pipeline right after windowing (as shown in Fig. 2.4). To isolate the effects of using this prosodic feature, I do not include it in any of the acoustic features used in this work. However, I do not exclude the 0^{th} cepstral coefficient from MFCCs, even though it bears some loudness-related information; more specifically, it can be considered as "a collection of average energies of each frequency bands in the signal that is being analyzed" (Zheng and Zhang, 2000).
- By pitch I mean the continuous pitch (and features derived from it) extracted using the Kaldi pitch algorithm presented by Ghahremani et al.. Because this feature type is not well known, I elaborate on it in the next section.

2.5.3 The continuous Kaldi pitch as a prosodic feature

Even though the original paper (Ghahremani et al., 2014) contains a detailed description of the Kaldi pitch algorithm (along with a working implementation as part of the Kaldi ASR toolkit), I provide an additional, more holistic view to give the reader an intuitive understanding of the algorithm and the features it produces (see Fig. 2.5). Perhaps the biggest contribution lies in abandoning the binary decision making about voiced/unvoiced speech, and only calculating pitch for the voiced frames. Instead, soft decisions are made based on the values of the normalised cross-correlation function (NCCF). For two pieces of digitised waveform separated by temporal distance l from each other and represented as vectors $\mathbf{v}_{t,0}$ and $\mathbf{v}_{t,l}$, the NCCF is computed as follows (adapted from Ghahremani et al., Eq. 2):

$$\Phi_{t,l} = \frac{\mathbf{v}_{t,0}^T \mathbf{v}_{t,l}}{\sqrt{(\mathbf{v}_{t,0}^T \mathbf{v}_{t,0})(\mathbf{v}_{t,l}^T \mathbf{v}_{t,l}) + B}} \tag{2.4}$$

where normally $B = 0$, but the authors use a non-zero value to push the NCCF values towards 0 where the cross-correlation on its own is very low. By computing NCCF for different spacings l – called *lags* by the authors, and being effectively the periods of different hypothesised frequencies – one obtains continuous confidence values about the presence of the corresponding frequencies in the signal for the time position t. After computing these for all temporal positions, the algorithm finds the optimal

[2]The use of logarithm is given solely by the Kaldi toolkit I use for feature extraction, although it is by no means an arbitrary design decision: Human perception of loudness is non-linear and logarithm is used also in the standard MFCC computation pipeline (see Fig. 2.4).

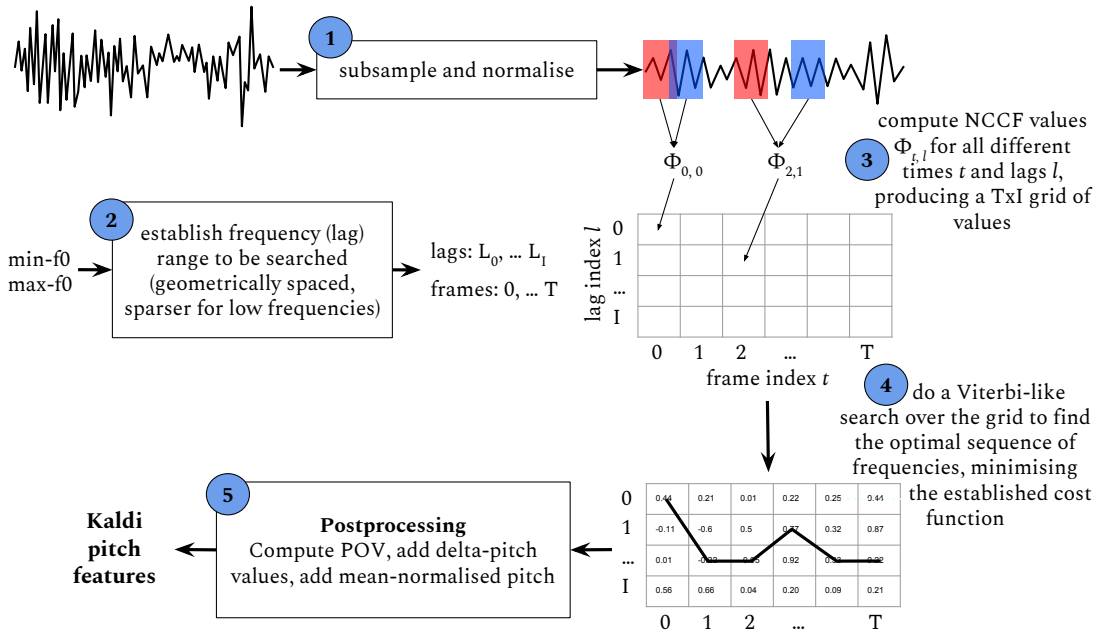

Figure 2.5: The steps of the Kaldi pitch algorithm.

continuous contour by minimising the proposed cost function (see Ghahremani et al., Eq. 4), which prefers higher NCCF values and penalises too low frequencies and too big frequency jumps.

Finally, the algorithm post-processes the raw pitch contour to provide these additional features:

1. A mean-normalised pitch contour, with the mean computed over 151-frame windows and weighted by a measure roughly corresponding to the log likelihood of a particular frame being voiced (i.e. putting more weight on voiced frames).
2. *Probability of voicing* (POV): A feature which is not a probability, but acts similarly and has been empirically found by the authors to provide improvements in ASR performance.
3. Delta-pitch terms, computed in the same way as the simple deltas in Eq. 2.2.

In this work, I work with all four feature types (the raw pitch contour and the three derived feature types) and refer to them collectively as the Kaldi pitch.

2.5.4 Illustrating the prosodic features used in this work

Having familiarised the reader with the two prosodic features I will use (the energy and the Kaldi pitch), I now illustrate them on a realistic example. Because the multilingual corpus used for this project does not contain English, I used the freely accessible sample utterance from the CSR-I (WSJ0) Complete corpus[3], which has important

[3] https://catalog.ldc.upenn.edu/LDC93S6A

characteristics such as the recording conditions and the speaking style the same as the corpus I later use (see Section 3.1).

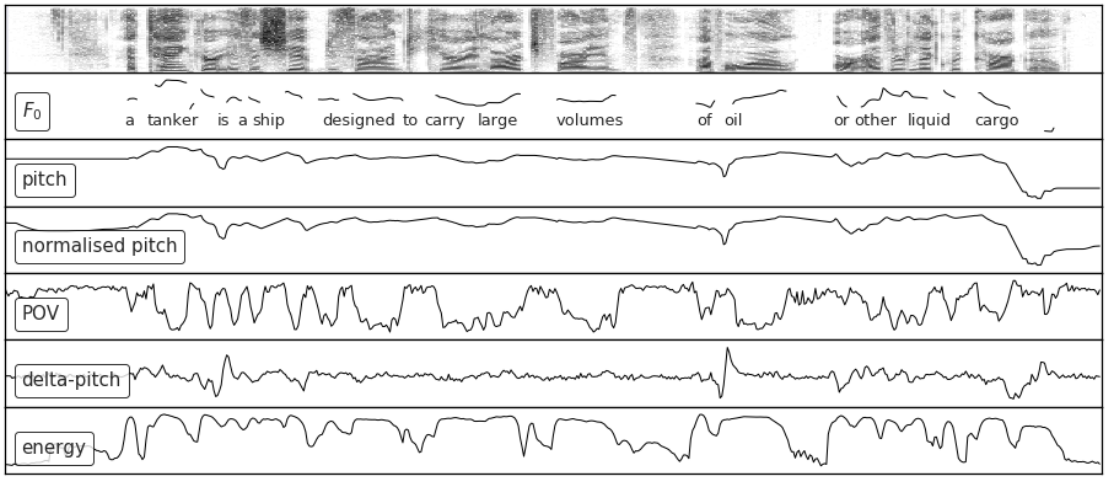

Figure 2.6: The prosodic features I use in this project (together with the spectrogram and the fundamental frequency F_0), illustrated on an English utterance. I manually created the time-aligned transcription for better understanding.

Fig. 2.6 shows the prosodic features on the English utterance. For clarity and comparison, I added the spectrogram as well as the discontinuous F_0 contour – both generated in Praat (Boersma and Weenink, 2019). As expected, the F_0 segments match the Kaldi pitch contour. Notice how the Kaldi pitch does not drop in unvoiced regions, but rather behaves like a smooth bridging between the regions. Also note that the POV values are effectively negated: High values corresponding to the regions where true F_0 is absent and low values marking the voiced regions.

Chapter 3

Data

Because LID systems are often part of ASR systems, it makes sense to use the same datasets to train both. Additionally, however, the U.S. National Institute of Standards and Technology (NIST) has been organising dedicated Language Recognition Evaluation[1] (LRE) challenges since 1996, providing multilingual datasets which are today the standard benchmark for evaluating language recognition systems (used also by Snyder et al. (2018a)). The NIST datasets, however, only focus on narrowband (8kHz) conversational telephone speech and include strong channel variability as a result of the uncontrolled recording environment, which makes it more difficult to analyse any observed effects and attribute them to particular language pair similarities or particular prosodic aspects. In this work, I use a relatively compact ($\sim$400 hours of speech) corpus which is in many aspects different from the NIST datasets.

3.1 The corpus: GlobalPhone

The corpus I use is the GlobalPhone multilingual ASR corpus (Schultz et al., 2013), more precisely its newest version – updated in 2016 and covering 22 languages from around the world: Arabic, Bulgarian, Chinese-Mandarin, Chinese-Shanghai (Wu), Croatian, Czech, French, German, Hausa, Japanese, Korean, Brazilian Portuguese, Polish, Russian, Spanish, Swahili, Swedish, Tamil, Thai, Turkish, Ukrainian, and Vietnamese. The corpus follows the style established for English by the Wall Street Journal-based corpora (Paul and Baker, 1992) such as the CSR-I (WSJ0) Complete corpus[2]. It consists of newspaper articles read by native speakers (around 100 speakers per language, each speaker producing about 20 minutes of audio by reading 3-5 articles), recorded as wideband (16kHz) audio in a controlled environment, using a close-talking microphone (in the case of GlobalPhone the Sennheiser microphone HD-440-6).

The controlled environment, speaking style and nature of the texts (all are major newspaper articles) make the data better suited for analysing effects of, for example,

[1] https://www.nist.gov/itl/iad/mig/language-recognition

[2] https://catalog.ldc.upenn.edu/LDC93S6A

prosodic features: Any observed trends can be more reliably attributed to my controlled independent variables (such as the feature type) or to the known language differences. Another advantage of the GlobalPhone corpus is the rich vocabulary (compare with short telephone conversations in the NIST corpora). Finally, from a linguistic view, the corpus covers a wide range of language families and prosody-based categories, while also providing an interesting sample of related languages in the form of 5 Slavic languages (which the author – as a native speaker of Slovak – can relate to): Bulgarian, Czech, Croatian, Russian, and Ukrainian.

Unfortunately, the corpus does not cover spontaneous speech (which – as known in ASR – is much more challenging than read text). The extremely low channel variability is also a tight constraint and it is possible that, even if prosodic features improve LID performance on this dataset, the results will not necessarily generalise well to more realistic scenarios with greater channel variability and noise.

3.2 Data partitioning

GlobalPhone ships with a partitioning of each language's data into training, evaluation and testing (in GlobalPhone documentation referred to as training, development and evaluation, respectively) sets, with the sizes of the 3 partitions being roughly 80%, 10%, 10%, respectively. However, for the purposes of the x-vector architecture, a 4-way partitioning is required:

1. training set – for training the x-vector TDNN,
2. enrollment set – for training the x-vector classifier,
3. evaluation set – for tuning the hyperparameters of both the TDNN and the classifier,
4. testing set – for final end-to-end evaluation on unseen data.

To create the desired 4-way partitioning, I allocated part of the training data for enrollment. This way, I preserved the original evaluation and testing, enabling fairer comparisons of my results with any other future works that use the GlobalPhone corpus. Because the classifier is typically a much simpler model (with many fewer learnable parameters) than the x-vector TDNN, I allocate only $1/8$ of the training set for enrollment (splitting each language's data in equal proportion). This way, my new training set accounts for roughly 70% of the whole corpus, while the 3 other sets are roughly 10% each.

Importantly, GlobalPhone[3] still misses partitioning for certain languages:

1. no partitioning for Czech, Polish, Tamil, Swahili, Ukrainian, Vietnamese and Wu,
2. incomplete partitioning for Arabic (incomplete evaluation and test sets), and for French, Japanese, Russian (missing evaluation sets for all three languages).

[3] as of newest version of its documentation I had access to: v4.1 from January 2016

The ASR GlobalPhone Kaldi recipe[4] extends the partitioning, providing it for Czech, French, Japanese, Polish, Russian, Swahili and Vietnamese. I utilise this partitioning in order to stay consistent with previous work using the recipe. For the rest of the languages with incomplete or missing partitioning (Arabic, Tamil, Ukrainian and Wu), I partitioned the data myself, following the same approach as the GlobalPhone authors – the only imposed constraint being: "no speaker appears in more than one group [partition] and no article is read by two speakers from different groups" (Schultz, 2002, p. 348). This constraint, however, could only be satisfied for languages which contained speaker metadata information (i.e. which articles were read by the particular speaker). Because this metadata was missing for Bulgarian, French, German, Polish, Tamil, Thai and Vietnamese, the splitting of the original training set into training and enrollment portions (and, in the case of Tamil, the entire 4-way partitioning) for these languages was done solely on a random basis.

I automated the entire partitioning process such that it would never allow one speaker to appear in multiple sets. However, I somewhat relaxed the constraint that the same article cannot appear in multiple sets. My implementation attempts to construct such partitioning, but – if it cannot be constructed – the allowed number of article overlaps is iteratively increased (from 0, incrementing by 1) until a satisfying partitioning is found. From all the languages for which the partitioning was generated non-randomly, the following languages had non-zero article overlaps: Arabic (385), Turkish (16), and Wu (49).

For the final partitioning (after excluding languages and speakers with corrupt data, see the next section) is shown in Tab. 3.1.

3.3 Data preprocessing

SHN to WAV creating lists (wav files, utterance-to-speaker, utterance-to-language, utterance-to-length) cutting into short segments: leave training data as it is (the TDNN chops it up into 1000-frame chunks anyway), split enrollment data into ¡30s segments, eval and test data into 3s or 10s (depending on the experiment) compute VAD (used across all experiments) from MFCC feats (based on C0)

3.4 Dealing with invalid data

This was discovered while preprocessing the data from .shn to .wav:

1. Hausa, Swahili and Ukrainian have broken data.
2. Bulgarian, German, Russian, Turkish and Vietnamese have one broken utterance recording each – not a big deal, as the number of utterances per language is in hundreds.

[4] https://github.com/kaldi-asr/kaldi/tree/master/egs/gp/s5

3. Portuguese has 2 speakers with almost all data broken (these were discarded), and other 10 speakers with up to 3 broken utterance recordings each (these were keps)

This leaves us with 19 languages, which are used further in this work: Arabic (AR), Bulgarian (BG), Mandarin Chinese (CH), Croatian (CR), Czech (CZ), French (FR), German (GE), Japanese (JA), Korean (KO), Polish (PL), Portuguese (PO), Russian (RU), Spanish (SP), Swedish (SW), Tamil (TA), Thai (TH), Turkish (TU), Vietnamese (VN) and Shanghai Chinese (WU).

3.5 Overview of the dataset

TRAINING	ENROLLMENT	EVALUATION	TESTING

Table 3.1: Example of architecture description of a TDNN (corresponds to Fig. 2.2).

Chapter 4

Implementation

4.1 The Kaldi toolkit

System was built in Kaldi (Povey et al., 2011). (Introduce Kaldi.)

4.2 Adapted implementations

Describe the SRE16 Kaldi recipe, the GlobalPhone ASR recipe, and how they were combined (also using the information from Snyder et al. (2018a)) to reproduce the LID x-vector system.

4.3 Choice of classifier

The SRE16 recipe uses PLDA, but for verification, not for classification. For our purposes, we needed a proper classifier. Current popular and well-performing classifiers are various flavours of GMM and logistic regression, with no clear winner. Snyder et al. (2018a), for instance, used GMM trained using MMI – based on McCree (2014). I decided to re-use a model which is already implemented in the LRE07/v2 Kaldi recipe – logistic regression. The decision was also consulted with Snyder (r 24). One theoretical downside of logistic regression is that the resulting decision boundaries are linear, whereas with GMM (trained with full co-variance matrix), one can achieve more complex quadratic decision boundaries. On the other hand though, training a full covariance requires much data, and the enrollment set would likely be not big enough. Additionally, the Kaldi logit can describe each class using more than linear boundary (see next section), so the decision boundaries should be complex enough to be able to model the observed data well.

4.4 Final architecture

Description of x-vector+GP architecture (highlighting own contributions)

how the whole pipeline works: go into much more detail than in the Related work section

description of the Kaldi logit, which models each class using multiple ”mixtures”, i.e. multiple boundaries

mention possibility of direct classification with TDNN and that I focus on using separate classifier because it provides extra flexibility and because it was shown to perform slightly better

4.5 Computing environment

cluster, Slurm, GPUs, parallelisation, rough runtimes of the different stages (TDNN training, x-vector extraction, logit training, inference)

4.6 Hyperparameters

decided: TDNN layers, activation function, learning algorithm (TO-DO: read about shrinking)

tuned (using the baseline setting, see next chapter): number of TDNN training epochs, logit hyperparameters

Chapter 5

Experiments

Intro: I will compare a selection of acoustic and prosodic features. Despite their reported potential, I don't use BNFs in this work, as they are basically just a higher-level feature based on the acoustic MFCC information (at least the BNFs in Snyder et al. (2018a)).

Evaluation metric: $C_{primary}$, consistent with NIST LREs. Elaborate a bit more on the meaning of the metric, maybe compare with accuracy and other simpler metrics.

5.1 Baseline

vanilla MFCCs (no deltas): also comment on the decisions made regarding MFCC MFCC configuration

$\leq 30s$ enrollment segments, $\leq 10s$ eval/test segments (should be possible to also report exact average segment length for each set)

5.2 Shifted delta cepstra

Want to see whether context aggregation in the form of added deltas in SDCs (compared to vanilla MFCCs) can improve performance, since the TDNN does context aggregation of its own.

5.3 KaldiPitch+Energy vectors

Calculating pitch even for unvoiced frames using the algorithm presented by Ghahremani et al. (2014). Adding raw energy to model stress. Extremely low-dimensionality feature vectors, but will see how the TDNN and classifier trained on these can do

prosodic LID. Hoping to achieve some sensible results, probably much worse than with MFCCs or SDCs.

5.4 MFCC/SDC + KaldiPitch+Energy

Taking the winner from MFCC/SDC, and concatenating those features with Kaldi pitch and raw energy values. Training the TDNN and classifier on that. Hopefully, seeing an improvement.

5.5 Fusion of MFCC/SDC and KaldiPitch+Energy scores

Stretch goal, likely to be delayed for Year 5 (or abandoned if there are more attractive ideas)

Same as previous section, but scores computed separately (using two systems, one trained on acoustic features, the other one on prosodic) and fused using a logit fuser. Probably using evaluation set for training the fuser (although, ideally, a separate data portion would be reserved for that).

5.6 Timeline for the experiments

1. Finished: System using MFCC and SDC.
2. By January 28th: Choose the TDNN and logit hyperparameters. TDNN using MFCCs has already been trained for, 2, 3, ... 10 epochs, now need to 1) establish reasonable logit parameters (driven by evaluation-set performance on x-vectors from TDNN trained for 3 epochs), 2) use that logit config to score the TDNNs with different number of training epochs, 3) choose a number of epochs that is a reasonable compromise between performance and training runtime
3. By February 4th: Have MFCC vs SDC comparison carried out (includes baseline results using MFCCs). Have KaldiPitch and raw energy features implemented (both are straightforward to compute with Kaldi – the energy is just computing MFCCs with raw log energy instead of C0, and discarding all but the energy-corresponding resulting coefficient). Have splicing of MFCC/SDC features with prosodic features implemented.
4. By February 11th: Have results of KaldiPitch+Energy experiments and of the acoustic+prosodic experiment (MFCC/SDC + KaldiPitch+Energy)

Chapter 6

Results

Reporting: overall $C_{primary}$, accuracy (for illustrative purposes), confusion matrix (to see which language pairs are confusing)

Focus on Slavic languages, since there is so many of them (Czech, Croatian, Polish, Russian, Bulgarian) and intonation can be very characteristic and important here (my own intuition, based on my knowledge of Slavic languages).

Chapter 7

Discussion

Chapter 8

Future work

8.1 Plans for Part 2 of the MInf project

8.2 Other future ideas

Everything that is sensible but unlikely in Year 5.

Chapter 9

Conclusions

Bibliography

Boersma, P. and Weenink, D. (2019). Praat: doing phonetics by computer [computer program]. version 6.0.49. http://www.praat.org/.

Crystal, D. (2008). *A Dictionary of Linguistics and Phonetics*. Blackwell, 6th edition.

Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., and Ouellet, P. (2011). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19:788–798.

Ferrer, L., Lei, Y., McLaren, M., and Scheffer, N. (2016). Study of senone-based deep neural network approaches for spoken language recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24:105–116.

Ghahremani, P., BabaAli, B., Povey, D., Riedhammer, K., Trmal, J., and Khudanpur, S. (2014). A pitch extraction algorithm tuned for automatic speech recognition. *2014 IEEE ICASSP*, pages 2494–2498.

Goedemans, R. and van der Hulst, H. (2013a). Fixed stress locations. In Dryer, M. S. and Haspelmath, M., editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Goedemans, R. and van der Hulst, H. (2013b). Weight-sensitive stress. In Dryer, M. S. and Haspelmath, M., editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

González, D. M., Lleida, E., Ortega, A., and Miguel, A. (2013). Prosodic features and formant modeling for an ivector-based language recognition system. *2013 IEEE ICASSP*, pages 6847–6851.

Grabe, E. (2004). Intonational variation in urban dialects of English spoken in the British Isles. *Regional variation in intonation*, pages 9–31.

Holmes, J. and Holmes, W. (2002). *Speech Synthesis and Recognition*. Taylor & Francis, Inc., Bristol, PA, USA, 2nd edition.

Lin, C.-Y. and Wang, H.-C. (2005). Language identification using pitch contour information. *Proc. 2005 IEEE ICASSP*, 1:I/601–I/604.

Mak, M.-W. and Chien, J.-T. (2016). Interspeech 2016 tutorial: Machine learning for speaker recognition. http://www1.icsi.berkeley.edu/Speech/presentations/AFRL_ICSI_visit2_JFA_tutorial_icsitalk.pdf.

Martinez, D., Plchot, O., Burget, L., Glembek, O., and Matějka, P. (2011). Language recognition in iVectors space. In *Proc. Interspeech 2011*.

McCree, A. (2014). Multiclass discriminative training of i-vector language recognition. In *Proc. Odyssey 2014*, pages 166–172.

Metze, F., Sheikh, Z. A. W., Waibel, A., Gehring, J., Kilgour, K., Nguyen, Q. B., and Nguyen, V. H. (2013). Models of tone for tonal and non-tonal languages. In *Proc. 2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 261–266.

Paul, D. B. and Baker, J. M. (1992). The design for the Wall Street Journal-based CSR corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. (2011). The Kaldi speech recognition toolkit. In *Proc. IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society.

Prieto, P. and Roseano, P. (2018). *Prosody: Stress, rhythm, and intonation*, pages 211–236. Cambridge University Press. Retrieved from `https://www.researchgate.net/publication/329541457_Prosody_Stress_rhythm_and_intonation`.

Sarma, M., Sarma, K. K., and Goel, N. K. (2018). Language recognition using time delay deep neural network. *arXiv*, abs/1804.05000.

Schultz, T. (2002). Globalphone: a multilingual speech and text database developed at Karlsruhe University. In *Proc. ICSLP 2002*, pages 345–348.

Schultz, T., Vu, N. T., and Schlippe, T. (2013). GlobalPhone: A multilingual text & speech database in 20 languages. *Proc. 2013 IEEE ICASSP*, pages 8126–8130.

Shimodaira, H. and Renals, S. (January 17, 2019). Speech signal analysis: Asr lectures 2&3. `http://www.inf.ed.ac.uk/teaching/courses/asr/2018-19/asr02-signal-handout.pdf`.

Snyder, D. (2018, December 24). Language identification with x-vectors: Choice of classifier [online forum comment]. `https://groups.google.com/forum/#!topic/kaldi-help/v6Uh7avv-cY`.

Snyder, D., Garcia-Romero, D., McCree, A., Sell, G., Povey, D., and Khudanpur, S. (2018a). Spoken language recognition using x-vectors. In *Proc. Odyssey 2018*, pages 105–111.

Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., and Khudanpur, S. (2018b). X-vectors: Robust DNN embeddings for speaker recognition. *2018 IEEE ICASSP*, pages 5329–5333.

Tkachenko, M., Yamshinin, A., Lyubimov, N., Kotov, M., and Nastasenko, M. (2016). Language identification using time delay neural network d-vector on short utterances. In *Proc. SPECOM*, pages 443–449. Springer.

Torres-Carrasquillo, P. A., Singer, E., Kohler, M. A., Greene, R. J., Reynolds, D. A., and Deller, J. R. (2002). Approaches to language identification using Gaussian mixture models and shifted delta cepstral features. In *Proc. Interspeech 2002*.

Variani, E., Lei, X., McDermott, E., Moreno, I. L., and Gonzalez-Dominguez, J. (2014). Deep neural networks for small footprint text-dependent speaker verification. pages 4052–4056.

Zheng, F. and Zhang, G. (2000). Integrating the energy information into MFCC. In *Proc. ICSLP 2000*.