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

The main point here is to show my intended structure and some idea of what each section would contain. Even where there is a paragraph or two written in a section (rather than just a bullet list or a few notes), it should be taken as a very early draft of the section and feedback should focus on outlined content (as opposed to the form).

Section 5.6 outlines the timeline for the next few weeks to finish all experiments.

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 containes 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

1.4 Overview

Describing the structure of the report that follows. (Is this useful at all?)

Chapter 2

Background

2.1 Spoken language identification

Spoken language identification (in this work also simply language identification or LID) means recognising the language of a speech segment. In a way, it is analogous to speaker identification (SID). Importantly, LID is typically realised as two different tasks: Simple classification (answering the question ”Which of these languages is X?”), or verification (”Is X this language?”). Simple classification is more suitable for use cases with a small and stable set of targets – such as a set of supported languages. There, computing the probability of X being each of the target languages is feasible. Verification, on the other hand, is more suitable for use cases where the number of targets is very high (such as a set of possible speakers) and even changes over time – there, it is often infeasible to compute the probability of X being each of the targets; instead, the system typically focuses only on evaluating the probability of X being the hypothesised language (or speaker). Throughout this work, I focus on LID as a classification task, with a closed set of target languages (i.e. not including the option to classify a speech segment as ”unknown language”).

2.2 Traditional approach – i-vectors

TO-DO: add a diagram of the pipeline

Since 2011, the state of the art in language and speaker identification was dominated by *i-vector* systems, first introduced by Dehak et al. (2011). These systems simulate the observation space using a huge Gaussian Mixture Model (GMM) – often consisting of thousands of multivariate Gaussians, typically trained using maximum likelihood estimation. To score a speech segment, it is processed one frame at a time, each frame’s feature vector is mapped onto the trained GMM, producing a high-dimensional vector representation. This representation is subsequently dimensionality-reduced and scored by a trained classifier.

2.3 State of the art – x-vectors

TO-DO: add a diagram of the pipeline, define TDNN and x-vectors properly

In 2018, the Johns Hopkins University team – Snyder et al. (2018a) – introduced a system which uses deep convolutional neural networks (referred to as time-delay neural networks – TDNN) to aggregate and process information across the frames of a variable-length speech segment. Such TDNN is trained to do direct language classification. Empirically, however, better results are achieved when low-dimensional embeddings called *x-vectors*, representing entire segments, are extracted from the trained TDNN and scored by a separate trained classifier. This system is attractive not just because of the promising performance: It no longer operates on the frame level; instead, it has the potential to effectively aggregate low-level information over multiple frames and do further processing. As demonstrated by the authors, when using a separate classifier, the architecture also supports – without much impact on the LID performance – extending the set of target languages without having to expensively re-train the x-vector TDNN. It is enough to train the light-weight classifier.

2.4 Features used in language identification

TO-DO: Add more references.

Historically, ASR has been the main point of focus among all speech processing areas, which is reflected in today's dominant use of acoustic features – primarily developed for ASR – in other tasks like LID and SID. These acoustic features, the most popular ones being MFCC, Mel filterbank and SDC, are not necessarily best-suited for the other tasks. For instance, all three disregard any pitch information – which makes sense in the context of ASR, where intonation is typically ignored, but is certainly not desirable in speaker identification, where intonation and the pitch range contribute to speaker-specific characteristics. Another ASR-inspired approach to LID is the *phonotactic* one, which exploits the different phone distributions across languages.

In this work, I acknowledge *acoustic* LID (using acoustic features), but *prosodic* LID receives the most attention – using features which are disregarded in ASR, but have a great potential in language and speaker identification tasks.

2.4.1 Acoustic features

Definitions of MFCCs and SDCs, probably ignore PLPs, Mel filterbanks, etc

SDC used by Ferrer et al. (2016) and Sarma et al. (2018) (7D MFCC + 7-1-3-7 SDCs = 56D) and by Torres-Carrasquillo et al. (2002) (10-1-3-3)

19 MFCCs + energy + 20 Δ + 20 $\Delta\Delta$ = 60D by Sarma et al. (2018) – outperformed by SDCs, but note that they trained an ASR TDNN for generating i-vectors

MFCCs also used inderectly to train ASR DNNs which then provide bottle-neck features which give very promising results in both LID and SID

2.4.2 Prosodic features

Elaborate on prosody as consisting of intonation (F0), stress (energy) and rhythm (segment durations).

39D MFCC vectors combined with 4D pitch features (Song et al., 2013)

Lin and Wang (2005) do LID just from the pitch contour parametrised by Legendre polynomials

Ghahremani et al. (2014) show ASR improvements with a pitch-tracking algorithm that calculates pitch even for unvoiced frames

Chapter 3

Data

Intro: Historically, speech corpora were meant for ASR, but now used for LID and SID as well. Since 1996, NIST has also been organising Language Recognition Challenges (LREs), and the corpora used for LREs are now the typically used ones when it comes to evaluating different systems. However, NIST traditionally focuses on telephone, narrowband (8kHz) speech, which represents only one part of all the possible settings in which LID or SID are deployed. In this work, I use a relatively small (¡ 40GB) corpus which, however, has many qualities that the various NIST corpora lack.

3.1 GlobalPhone

Presented by Schultz et al. (2013), originally for ASR. Wideband (16kHz), recorded native speakers of 23 languages reading newspaper articles – very similar to the English corpora such as CSR-I, which are based on Wall Street Journal articles. Unlike in NIST corpora, the recording equipment and conditions vary very little. One disadvantage of GlobalPhone is the lack of spontaneous speech. On the other hand, the language goes well beyond the simple conversational language found in the NIST telephone speech corpora.

3.2 Data partitioning

GlobalPhone comes with a partitioning of each language's data into training, development and evaluation datasets, with the sizes of the datasets being roughly 80%, 10%, 10%, respectively. However, for the purposes of the x-vector architecture, 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 ('development' in the GlobalPhone terminology) – for tuning the hyperparameters of both the TDNN and the classifier,
4. testing set – for final performance evaluation on unseen data.

In order to use GlobalPhone with the x-vector system, I allocated part of the training data for enrollment. I tried to preserve the GlobalPhone development sets and evaluation sets (which I refer to as evaluation and testing sets, respectively), in order to enable fair comparison of my results obtained on those data portions with any other works that use the GlobalPhone corpus.

Taking into account the relatively small number of parameters of the x-vector classifier when compared to the x-vector TDNN, I split the training data such that approximately only one 8th is used for enrollment. This way, the training data accounts for roughly 70% of the whole corpus, while the 3 other portions are roughly 10% each.

Importantly, GlobalPhone (as of the GlobalPhone Documentation v4.1) still misses the partitioning for certain languages:

1. no partitioning for Czech, Polish, Tamil, Swahili, Ukrainian, Vietnamese and Shanghai Chinese,
2. only partial partitioning for Arabic, French, Japanese, Russian.

The GlobalPhone Kaldi recipe contains an extended partitioning, which fixes Czech, French, Japanese, Polish, Russian, Swahili and Vietnamese. For the rest of the languages with incomplete partitioning (Arabic, Tamil, Ukrainian and Shanghai Chinese), I created the partitioning myself, following the same approach as the GlobalPhone authors: "No speaker appears in more than one group and no article was read by two speakers from different groups" (TO-DO: reference the GP docs).

Some languages didn't have speaker-article data; for those, the partitioning was done randomly.

For some languages, I could not construct a partitioning with zero article overlap; for those, I tried to at least minimise the overlap.

3.3 Initial data preprocessing

Basically, SHN to WAV

Splitting long utterances into shorter ones uniformly, to be able to do classifier training and end-to-end evaluation on segments of different lengths – like in Snyder et al. (2018a). One future improvement would be to not do uniform splitting, but rather split on breaks – to prevent potentially bad edge effects.

3.4 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.
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).

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

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.

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.

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

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

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 *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

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

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

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.

Song, Y., Jiang, B., Bao, Y., Wei, S., and Dai, L. (2013). I-vector representation based on bottleneck features for language identification. *Electronics Letters*, 49(24):1569–1570.

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 *Interspeech*.