COMPRISE

Cost effective, Multilingual, Privacy-driven voice-enabled Services

www.compriseh2020.eu

Call: H2020-ICT-2018-2020
Topic: ICT-29-2018
Type of action: RIA
Grant agreement Nº: 825081

WP Nº3: Multilingual personalised voice interaction

Deliverable Nº3.2: Initial personalised learning library for speech-to-text

Lead partner: INRIA

Version Nº: 1.0

Date: 30/04/2020

European Commission

Document information	
Deliverable Nº and title:	D3.2 – Initial personalised learning library for speech-to-text
Version Nº:	1.0
Lead beneficiary:	Inria
Author(s):	Denis Jouvet (INRIA) and Tuğtekin Turan (INRIA)
Reviewers:	Askar Salimbaev (Tilde) and Dietrich Klakow (USAAR)
Submission date:	30/04/2020
Due date:	30/04/2020
Type[1]:	OTHER
Dissemination level[2]:	PU

Document history			
Date	Version	Author(s)	Comments
08/04/2020	0.1	Denis Jouvet & Tuğtekin Turan	Draft deliverable
20/04/2020	0.2	Denis Jouvet & Tuğtekin Turan	Modifications to take into account reviewers' comments
30/04/2020	1.0	Zaineb Chelly & Emmanuel Vincent	Final version revised by the project manager and the coordinator

[1] **R**: Report, **DEC:** Websites, patent filling, videos; **DEM:** Demonstrator, pilot, prototype; **ORDP:** Open Research Data Pilot; **ETHICS:** Ethics requirement. **OTHER:** Software Tools

[2] **PU:** Public**; CO:** Confidential, only for members of the consortium (including the Commission Services)

Document summary

This deliverable is devoted to the design, implementation, and evaluation of an initial personalised learning library for Speech-to-Text. It consists of this report and software components that are available to the public on the COMPRISE git repository[3]. Some of these software components will be integrated in the COMPRISE Cloud Platform and others in the COMPRISE Software Development Kit (SDK) Client Library. The report first recalls in Section 2 the COMPRISE modules that are impacted by this first version of personalised learning for Speech-to-Text. Section 3 describes the scientific approaches that have been investigated for this task. The approaches rely on taking into account the user's characteristics through the addition of speaker embedding vectors which are provided to acoustic models along with the conventional spectral features. Experimental results are reported and discussed in Section 4. The introduction of speaker embedding vectors leads to 10% to 15% reduction in the word error rates and, in most cases, the x-vector embeddings yield better performance than the conventional i-vector embeddings. We describe the main functionalities of the software library in Section 5.

[3] https://gitlab.inria.fr/comprise/deliverables/deliverable_d32

Table of contents

1. Introduction 5
2. Architecture 5
3. Scientific approach 7
3.1. State-of-the-art and related work 8
3.1.1. Accent variability 8
3.1.2. Speaker variability 8
3.1.3. Proposed approach 9
3.2. Speaker embedding 9
3.2.1. i-vector embedding 10
3.2.2. x-vector(speaker) embedding 10
3.2.3. x-vector(accent) embedding 11
3.3. Acoustic modeling 11
3.4. Semi-supervised learning 11
4. Scientific results 12
4.1. Experimental setup 12
4.1.1. Dataset 12
4.1.2. Acoustic model and lexicon 13
4.1.3. Extraction of speaker embeddings 13
4.2. Results and discussion 14
4.2.1. Speaker embedding and training data 14
4.2.2. Window length for speaker embedding 16
4.2.3. Evaluation using privacy transformed speech data 18
4.2.4. Inclusiveness 18
4.2.5. Semi-supervised learning 19
5. Software library 20
5.1. Initial Steps 20
5.1.1. Prerequisites 20
5.1.2. Setup 21
5.2. Run the Library 21
6. Conclusion 22
7. Bibliography 23

1. Introduction

The objective of Work Package 3 is to enable any user to interact with dialog systems in any language. To achieve this objective, Work Package 3 focuses on combining machine translation with Speech-to-Text (STT) and text-to-speech, on combining machine translation with the components of a dialog system, and on adapting these models to every user. Deliverable D3.2 is concerned by this last point, and deals with personalised learning approaches.

More precisely, Deliverable D3.2 describes the initial personalised learning library for STT. As STT in the COMPRISE library is based on Kaldi [Povey *et al.*, 2011], the personalised learning approaches investigated and described in this deliverable also use the Kaldi toolkit and framework. To avoid storing user's speech data on the user's device, and thus to minimise memory requirements on the user's device, the approaches investigated here rely on introducing speaker embedding features along with the usual spectral features in the STT processing. Several types of speaker embeddings are investigated including a conventional one based on i-vectors, and two other based on x-vectors. i-vectors were initially developed for speaker recognition. Speaker embedding through x-vectors is a rather recent approach of speaker embedding, and was also developed for speaker recognition. The x-vectors are estimated through deep learning approaches. Initially x-vectors were estimated with a Deep Neural Network (DNN) trained to recognise speakers' identities. In this deliverable, a variant is proposed by training a DNN to discriminate between accents.

For evaluating the proposed approaches, we have mainly used a speech corpus corresponding to spontaneous scheduling dialogues: the Verbmobil corpus (described in Section 4.1). This corpus contains a significant amount of non-native English speech, hence STT performance is evaluated on native and on non-native speech. Further experiments have also been conducted using accented English speech from the Voxforge project [Voxforge, 2020].

The deliverable is organised as follows. Section 2 explains where the computations take place in the global COMPRISE architecture. Section 3 details the scientific aspects. It positions the studied approaches with respect to the state-of-the-art, and then details the speaker embedding approaches, their integration with acoustic modeling, and presents a semi-supervised approach for training on more (unannotated) data. Section 4 presents the experimental setup, and then details and discusses the results. Some results are also given with respect to the application of the proposed approach on transformed speech data resulting from the application of the privacy-driven speech transformation presented in Deliverable D2.1. We describe the main functionalities of the software library in Section 5.

2. Architecture

Figure 1 positions STT training and decoding components in the overall COMPRISE architecture. In that figure, personalisation translates into the two "Speaker vector computation" blocks and their use by the subsequent "STT learning" and "STT" blocks.

Personalised features such as i-vectors or x-vectors (see Section 3) are computed from the user's speech signal. Such features, which provide a speaker embedding, have been initially proposed for speaker recognition, and hence carry speaker information. Taking such features into account in STT results in personalised computations.

To do so, the personalised features (i.e., the speaker embedding) are given as inputs to the DNN based acoustic model along with conventional spectral features. In the training branch, the model learns to map every observed sound to the corresponding phonetic class depending on the speaker embedding. The learning operates on data which have undergone a privacy-driven transformation. In practice, this transformation converts the voice of every original speaker into that of another random speaker called “target” (see Deliverable D2.2). Provided that the range of target speakers in the training data is large enough, this model is expected to generalize well to original speakers in the operating branch. Once this model has been trained, it can be downloaded on the user’s device.

At run time, the speaker’s embedding is also computed and used by the STT decoder. It is important to note that this embedding remains on the user device and is never sent to the Cloud Platform, since it would reveal personal features of the user. Note also that, for online decoding, speaker embeddings are computed on the fly and regularly updated (typically every 100 ms) during the decoding process.

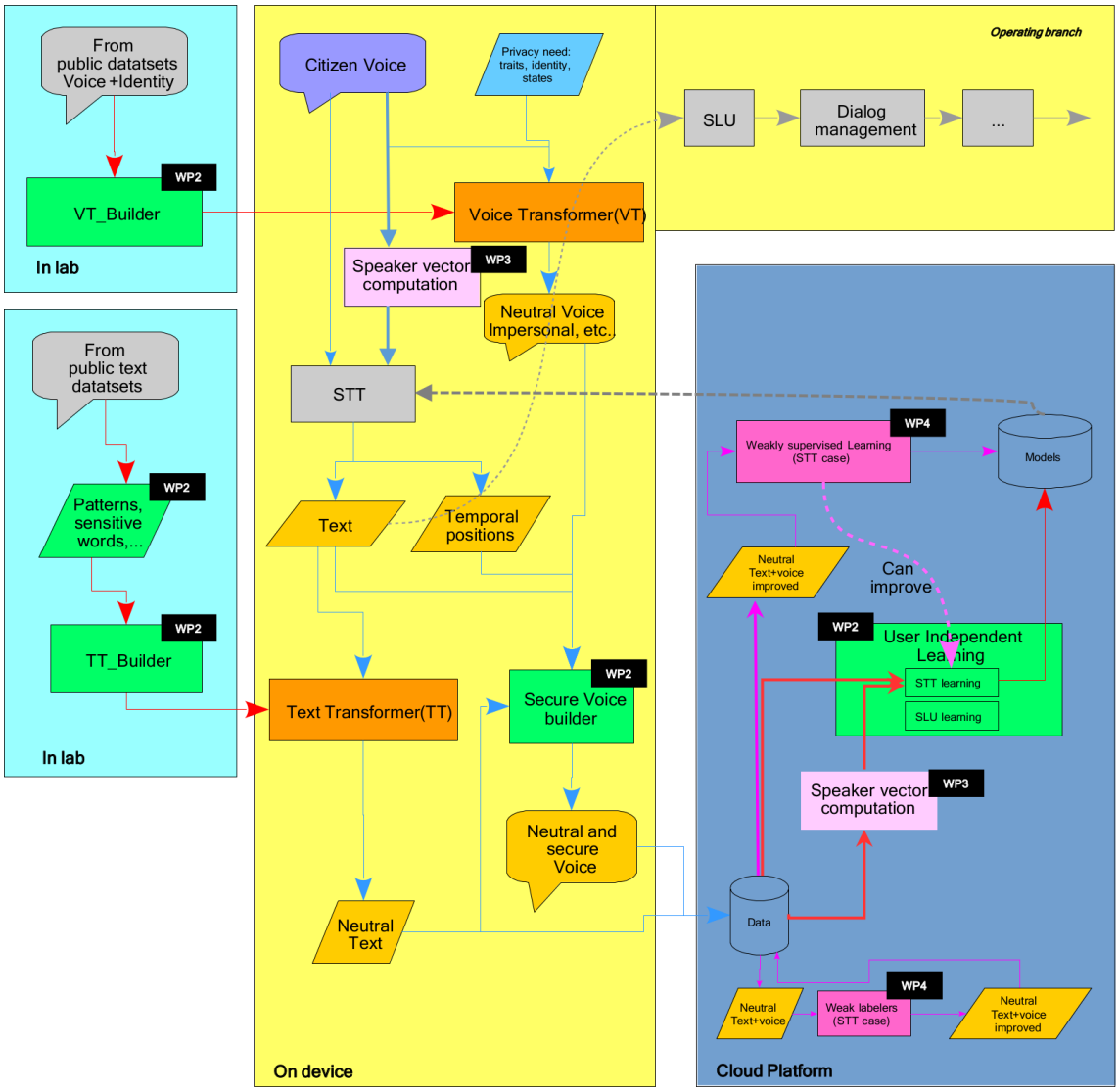

Figure 1: Overview of STT training and decoding in COMPRISE.

Compared to the original figure in Deliverable D2.1, the “speaker vector computation” block in Figure 1 has been moved to the center of the figure, and to the “Cloud Platform”

(lower right), since it computes the speaker embedding which is used both for training the acoustic model and for decoding at run time by the STT module on the user's device. Also, since personalisation is achieved through the additional use of speaker embedding information at the input of the acoustic model, there is no need for further adaptation of the acoustic model to each speaker. Hence, the personalization block, as mentioned in D2.1, is no longer needed, and has been removed from the figure.

3. Scientific approach

After a presentation of the state-of-the-art and related works in Section 3.1, the proposed approach for personalising STT is described in detail. Section 3.2 presents the speaker embeddings that are computed to represent the speaker information which is then exploited by the acoustic models, along with spectral features. Section 3.3 summarises design choices for acoustic modeling. Finally, Section 3.4 describes the semi-supervised process proposed for handling larger amounts of untranscribed non-native speech data. Figure 2 illustrates the overall approach.

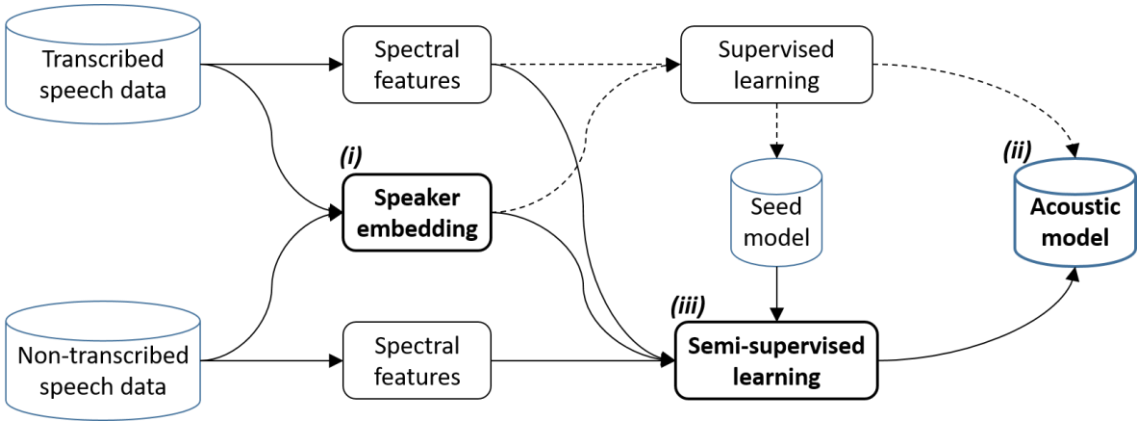

Figure 2: Training process for STT model personalisation involving three main blocks: (i) speaker embedding, (ii) acoustic modeling, (iii) semi-supervised learning.

Speaker adaptation of STT models can be achieved by modifying parameters of the acoustic models in order to match some adaptation utterances of the desired speaker. Another approach consists in making the acoustic model speaker-aware, by providing speaker-specific information at the input of the DNN, along with spectral features. This is achieved through speaker embeddings. In the following, we consider three types of speaker embeddings: conventional i-vectors, and two flavors of x-vectors obtained by training the x-vector extractor to classify either speakers or accents, from spectral input features. The last hidden layer of the extractor determines the x-vector embedding.

In such an approach, the acoustic model handles jointly two sets of input features: one that represents the spectral information, and one that corresponds to the speaker embedding.

The semi-supervised learning block is an additional module that is used during training to provide supervision for unlabeled accented speech. It is based on STT processing; the lattice that results from the decoding is used to determine different possible transcripts and the associated weights.

3.1. State-of-the-art and related work

STT performance degrades when there is a mismatch between the training and test speakers [Siniscalchi *et al.*, 2013]. Gender and accent are regarded as the most important factors of this mismatch [Yao *et al.*, 2014]. Current ASR models trained on native speech data often experience a dramatic loss of accuracy for speakers with strong accents [Viglino *et al.*, 2019]. However, there is relatively little research on accented STT, especially for speakers who have a different mother tongue.

3.1.1. Accent variability

With the advent of smart voice-assistants and dialogue systems, voice interfaces can perform daily tasks such as booking tickets, setting up calendar items or finding restaurants via spoken interactions [Celikyilmaz *et al.*, 2015]. Voice technology also encourages several applications about education, government, healthcare, and entertainment [Sarikaya *et al.*, 2016]. Non-native speakers in these systems have a very wide variety of accents usually influenced by their native language. Hence, accented speech introduces heterogeneous characteristics and affects the overall performance of any dialogue system.

Previous research on non-native or foreign-accented speaker adaptation mainly transforms the acoustic model, initially trained on unaccented speech. The simplest approach is merely the use of Maximum Likelihood Linear Regression (MLLR) on unseen speakers [Tomokiyo and Waibel, 2003]. A more effective approach is to train an accent-specific model for each accent [Elfeky *et al.*, 2016]. Although it performs well, a separate model per accent increases computational cost [Nguyen *et al.*, 2017]. This has motivated research on recognising speech from multiple accents with a single model. However, this approach can underperform compared to accent-specific models, especially when the accented data is scarce [Li *et al.*, 2018].

Collecting sufficient data may be infeasible due to the large number of possible accents. Recently, there have been many attempts to improve the single-model approach for accented STT. One such approach is based on multi-task learning, where the model is trained not only to predict phonemes but also to identify accents [Yang *et al.*, 2018]. Other approaches provide auxiliary input such as bottleneck features or dialect information to make more adaptive models [Jain *et al.*, 2018]. In another approach, an accent-specific model is obtained simply by training on all the available data and then fine-tuning on non-native samples [Huang *et al.*, 2014]. One of the main drawbacks is that such a method needs to maintain several steps for a given accent, which increases maintenance effort in an ASR system dealing with many accents or dialects [Yoo *et al.*, 2019].

3.1.2. Speaker variability

Many factors about speaker variability are complex in nature and there is no straightforward solution or model compensation for them. Hence, speaker adaptation methodologies either learn speaker transformations from target examples or require some additional data to adjust the large set of parameters in the ASR system [Zhang *et al.*, 2013].

Different approaches have been proposed to reduce the mismatch between training and test environments in traditional STT systems. For Gaussian Mixture Model (GMM) based acoustic models, speaker adaptation has proven to be effective for many years within

the transformation-based and Bayesian approaches. Methods like Maximum A Posteriori (MAP) adaptation [Tsao *et al.*, 2011], eigenvoice speaker adaptation [Weiss and Ellis, 2010] or MLLR [Povey and Yao, 2011] focus on transformations in the model space of either the Gaussian means or both the Gaussian means and variances. On the other hand, the feature-space MLLR (fMLLR)-based approach [LI *et al*., 2002] applies the MLLR transform on the features.

STT accuracies have improved substantially over the past years with the help of DNN-based acoustic models. It is reported that Word Error Rate (WER) has reduced between 10 to 32% in a wide variety of tasks compared with GMM-based systems [Hinton *et al.*, 2012]. Although displaying superior generalisation ability with respect to GMMs, DNN-based models still suffer from the mismatch between speakers. Several methods have been proposed for speaker adaptation in DNN-based models. The most straightforward approach is to augment Speaker-Independent (SI) models by adding some speaker-specific layers that are trained on the adaptation data [Kumar *et al.*, 2015; Zhang and Woodland, 2016]. Similarly, it is also possible to re-update certain layers of the SI model [Zhao *et al.*, 2018; Xie *et al.*, 2019]. However, these methods are very prone to over-fitting especially when adaptation data is coming from particular speakers [Kitza *et al.*, 2018]. I-vectors, which capture both speaker and environment specific information, have been shown to be useful for rapid adaptation [Karafiát *et al.*, 2011; Saon *et al*., 2013; Xue *et al*., 2014]. These vectors are used as an additional input to the feature layer of the DNN acoustic model. In initial versions, during training and recognition, one i-vector per utterance is computed and all the frames corresponding to a particular speaker have the same i-vector appended to them.

3.1.3. Proposed approach

Here, we focus on non-native and accent issues that are addressed in a general STT framework. The impact of non-native and accented speech is explored using the Kaldi toolkit [Povey *et al.*, 2011]. Cross-accent recognition experiments show that the error rates are increased up to 25-35% when the acoustic model and the test data are from different accents. To deal with accent variability, we trained a single model that includes speaker embeddings, employed as additional input features for the senone classifier. It is also possible to achieve better performance by adding a small amount of enrollment data from non-native and/or accented speech. Our STT systems rely on Time-Delayed Neural Network (TDNN) acoustic models [Peddinti *et al.*, 2015] trained using the Lattice-Free Maximum Mutual Information (LF-MMI) criterion [Povey *et al.*, 2016] and the cross-entropy criterion simultaneously. For the sake of simplifying our experiments, we focus here only on non-native and accented English. Nevertheless, we believe that our proposed approach will also be helpful for accented speech in other languages.

3.2. Speaker embedding

Three types of speaker embeddings are considered: i-vector, x-vector(speaker), and x-vector(accent). The last two are computed by means of a DNN which is either trained to recognise speakers for the x-vector(speaker) embedding, or trained to recognise accents for the x-vector(accent) embedding. We use the term ‘x-vector’ for both approaches because they are based on a similar deep learning formalism, but we indicate in parentheses the classification objective of the DNN.

3.2.1. i-vector embedding

Initially developed for speaker recognition [Dehak et al., 2011] i-vectors have been largely used in other speech processing tasks. Each speech segment is assumed to be represented by a GMM and principal component analysis is applied to the GMM supervectors[4]. The basic assumption is that all speech segment supervectors are confined to a low dimensional subspace of the GMM supervector space so that each utterance supervector is specified by a small number of coordinates. The supervector M of a GMM adapted on a speech segment can be expressed as

$$M = m + T\,w \tag{1}$$

where m is the supervector of the Universal Background Model (UBM), T is a low rank projection matrix, and w is the resulting i-vector which characterises the speech segment.

3.2.2. x-vector(speaker) embedding

'x-vector(speaker)' embedding was introduced in [Snyder *et al.*, 2018]. The embedding is based on a DNN model as the one represented in Figure 3, where a TDNN is used to handle a short-term temporal context. The lower layers of the network do frame-level processing. The upper layers do segment-level processing: after pooling the frame-level information, they provide a representation (last hidden layer) which is independent of the length of the speech segment. This representation is the x-vector speaker embedding.

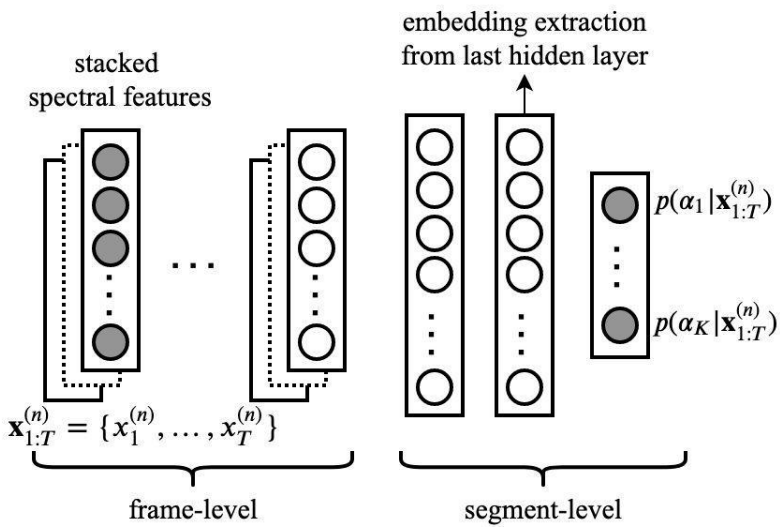

Figure 3: DNN model used for extracting x-vector embeddings. It combines frame-level processing (lower layers) and segment-level processing (higher layers). The segment-level embeddings are extracted from the last hidden layer.

For conventional x-vectors (here named 'x-vector(speaker)'), the network is trained to classify speakers using a multi-class cross entropy objective function defined in Eq. (2). Let K denote the number of speakers in the N training speech segments, and $p\left(\alpha_k \middle| x_{1:T}^{(n)}\right)$ the probability of speaker k given the T input frames $x_1^{(n)}, \ldots, x_T^{(n)}$ of the speech segment. The cross-entropy objective function is thus

[4] The GMM supervector is the concatenation of the mean vectors of all the Gaussian components.

$$E = -\sum_{n=1}^{N} \log\left(p\left(\alpha_{k(n)} \middle| x_{1:T}^{(n)}\right)\right) \tag{2}$$

where $\alpha_{k(n)}$ is the speaker ID of the speech segment $x_{1:T}^{(n)}$.

3.2.3. x-vector(accent) embedding

The proposed approach to derive an 'x-vector(accent)' embedding is very similar to the approach used to derive an 'x-vector(speaker)' embedding. It also relies on a DNN model, however, here, the output layer is used to recognise accents instead of speakers.

So, the formula of Section 3.2.3 still holds here, but now $\alpha_{k(n)}$ corresponds to the accent present in the speech segment $x_{1:T}^{(n)}$.

3.3. Acoustic modeling

We use TDNNs as acoustic models [Peddinti *et al.*, 2015] since they have been shown to learn effectively the temporal dynamics of the speech signal from short-term feature representations [Waibel *et al.*, 1989]. The model is used for representing rather long-term temporal dependencies from short-term spectral features.

The acoustic models are trained using the LF-MMI training criterion [Povey *et al.*, 2016] and the cross-entropy criterion simultaneously.

3.4. Semi-supervised learning

In some cases, for example when dealing with non-native or accented speech, obtaining large amounts of transcribed speech data is difficult, while the amount of untranscribed speech data may be larger by several orders of magnitude. In such cases, a semi-supervised approach might be useful.

We applied this idea for training acoustic models using the LF-MMI objective [Manohar *et al.*, 2018]. The main idea behind the approach is to apply a Finite-State Transducer (FST) based supervision. This provides a natural way to incorporate uncertainties when dealing with untranscribed speech data and allows the supervision of lattices obtained via decoding of untranscribed speech data.

In this setting, the minimisation of lattice entropy is the natural extension of the MMI objective to the semi-supervised setting. Given the transcriptions W and acoustic features X, the MMI estimation can be defined as

$$\mathcal{F}_{MMI}(\lambda) = \sum_{r} \log\big(p(W^r|X^r;\lambda)\big) \tag{3}$$

where the index r ranges over all training utterances, and λ represents the parameters of the model. In the semi-supervised learning approach, an average of this expression over all possible reference transcripts W^r is obtained taking into account the probabilities of the word sequences in the lattice

$$\mathcal{F}_{SSL}(\lambda) = \sum_{r}\sum_{W} p(W^r|X^r;\lambda)\log\big(p(W^r|X^r;\lambda)\big). \tag{4}$$

This criterion is also known as negative conditional entropy [Manohar *et al.*, 2015].

4. Scientific results

This section presents the experiments that have been conducted. Section 4.1. describes the experimental setup, and Section 4.2. presents and discusses the main obtained results.

4.1. Experimental setup

4.1.1. Dataset

A large part of the evaluations has been conducted on the Verbmobil corpus [Hess *et al.*, 1995] which contains spontaneous dialogue speech taken from human-human scheduling dialogues. In Verbmobil, we selected the volumes related to American English as native samples and accented English spoken by Germans as non-native speech.

For the initial set of experiments, the data used for training and test are as described in Table 1.

Table 1: Initial split of Verbmobil data used in the first set of STT personalisation experiments.

Training data			Test data		
Type of data	**Number of speakers**	**Duration (h)**	**Type of data**	**Number of speakers**	**Duration (h)**
Native (US)	260	24.5	Mainly non-native (German speakers)	34	3.8

A revised split of the Verbmobil data was later defined to better evaluate the performances on native speech and non-native speech, respectively; and to evaluate the benefit of adding a small amount of non-native speech data in the training set. Moreover, accented English speech data has been taken from VoxForge open corpora [VoxForge, 2020]; this includes Australian, Indian and British English. Table 2 provides a description of the speech datasets used in STT experiments.

Table 2: Revised split of Verbmobil and VoxForge data used in the second set of STT personalisation experiments.

Corpora	Type of data	Training data		Additional training data		Test data	
		Number of speakers	**Dur. (h)**	**Number of speakers**	**Dur. (h)**	**Number of speakers**	**Dur. (h)**
Verbmobil	Native (US)	235	25.4	---	---	25	1.1
	Non-native (DE)	---	---	25	1.0	25	1.1
VoxForge	British (UK)	---	---	28	1.0	25	1.2
	Australian (AU)	---	---	23	1.0	25	1.3
	Indian (IN)	---	---	21	1.0	25	1.4

Two types of STT experiments are conducted with these data. The first set of experiments is based on training acoustic models using only native speech data (column 'training data' of Table 2). The second set of experiments is conducted by including also non-native and/or accented data in the training process (cf. column 'additional training data' of Table 2). Note that non-native datasets are much smaller than the native one, which is a common setting in multi dialect STT [Yoo *et al.*, 2019].

In addition, the speech data described in Table 3 have been used for training the DNN model that provides the x-vector(accent) embedding. As above, the native (US) and non-native (DE - by German speakers) speech data come from the Verbmobil corpus, and the three other accented datasets (British, Australian, and Indian) come from VoxForge. Of course, the speakers are different from those selected in the test sets described in Table 2.

Table 3: Description of the speech datasets used to train the DNN model for x-vector(accent) embedding.

Corpora	Type of data	Number of speakers	Duration (h)
Verbmobil	Native (US)	235	25.4
	Non-native (DE)	54	3.4
Voxforge	British (UK)	148	5.7
	Australian (AU)	117	3.4
	Indian (IN)	79	3.1

In all experiments, we use Mel-Frequency Cepstral Coefficients (MFCCs) computed over 25 ms windows, with a 10 ms frame shift, using Kaldi scripts.

4.1.2. Acoustic model and lexicon

The TDNN acoustic model operates on 40-dimensional MFCC features and is similar to the model specified in [Peddinti *et al.*, 2015]. If t is the current time step, the first layer splices frames $\{t-2, t-1, t, t+1, t+2\}$ together at the input layer; then the following five hidden layers splice the previous layer output at the following offsets: $\{-1, +1\}$, $\{-1, +1\}$, $\{-3, +3\}$, $\{-3, +3\}$, $\{-6, +3\}$. As indicated in [Povey *et al.*, 2016], index differences that are multiples of 3 for most of the hidden layers helps reduce the amount of computations for the chain models that use a subsampling factor of 3.

The speed-perturbation technique [Ko et al., 2015] is used with a 3-fold augmentation where copies of training data are created according to factors of 0.9, 1.0 and 1.1.

For decoding, we use a lexicon containing 6945 words, and a 3-gram language model trained on only native data consisting of 214,036 word occurrences with a perplexity of 42.7. All experiments are conducted using the same language model and the same language model weight.

4.1.3. Extraction of speaker embeddings

The speaker embeddings are appended to the MFCC features. By default, and unless otherwise specified, the speaker embeddings are updated every 10 frames, i.e., every 100 ms. The speaker embeddings are extracted and handled using the Kaldi framework.

i-vector embeddings have a dimension of 100. They are extracted using a 512-Gaussian GMM applied on 40-dimensional acoustic features. The acoustic features are obtained by linear discriminant analysis applied on 10 stacked (i.e., 100 ms) MFCC frames of dimension 13.

The x-vector(speaker) embeddings are computed using a DNN trained on the Verbmobil data. We found that this leads to better STT performance than training on the bigger VoxCeleb corpus [Nagrani *et al*.]. The x-vector(speaker) embeddings have a dimension of 512, and are computed over 250 ms windows corresponding to 25 MFCC frames of dimension 13.

The x-vector(accent) embeddings are obtained in a similar way, except that the DNN is trained to recognise accents instead of speaker IDs. The embeddings are computed over 400 ms windows corresponding to 40 MFCC frames. The architecture of the DNN is described in Table 4. The frame level layers correspond to a TDNN approach. The pooling layer extracts information over the whole considered segment. The DNN used for extracting this embedding was trained on the speech data described in Table 3.

Table 4: Architecture of the DNN used to compute the x-vector(accent) embedding (extracted from layer segment7, before the non-linearity). The input layer accepts *F*-dimensional features and *K* corresponds to the number of accents.

Layer	Context	Frames	Inputs × Outputs
frame1	$\{t-2, t-1, t, t+1, t+2\}$	5	$(5 \times F) \times 512$
frame2	$\{t-2, t, t+2\}$	9	$(3 \times 512) \times 512$
frame3	$\{t-3, t, t+3\}$	15	$(3 \times 512) \times 512$
frame4	$\{t\}$	15	512×512
frame5	$\{t\}$	15	512×1500
pooling	Segment $[0, T)$	T	$(T \times 1500) \times 3000$
segment6	---	T	3000×512
segment7	---	T	512×512
softmax	---	T	$512 \times K$

4.2. Results and discussion

This section presents and discusses the results of the experiments. They are grouped in several subsections corresponding respectively to the impact of speaker embedding and the training data, the impact of the length of the window used to compute the speaker embedding, the application of the approach on privacy transformed speech signals, an analysis of speaker inclusiveness, and finally preliminary results achieved with semi-supervised learning.

4.2.1. Speaker embedding and training data

The first set of experiments have been conducted on the Verbmobil data using the initial split described in Table 1. The WERs achieved on the test set are reported in Table 5.

Table 5: WER (%) on the test set of the initial Verbmobil split with respect to the type of model and the type of speaker embedding used.

Model	Embedding	Test WER
GMM/HMM	none	34.7 %
GMM/HMM with fMLLR	none	28.9 %
Neural networks (Kaldi nnet3)	none	30,7 %
	i-vector	26.1 %
	x-vector(speaker)	25.7 %
Sequence trained neural network (Kaldi chain)	none	28.3 %
	i-vector	23.8 %
	x-vector(speaker)	23.1 %

Word error rates are quite high because the training is carried out on native data only, and 85% of the test data is non-native speech. The Kaldi chain model outperforms the other types of acoustic models. This type of acoustic model will be used in the remaining experiments. Adding speaker embeddings, whether i-vector or x-vector(speaker), improves the STT performance. The best results are achieved with the x-vector(speaker) embedding. This is consistent with the fact that x-vectors lead to better performance than i-vectors for speaker recognition.

The revised split of the Verbmobil data (cf. Table 2, top) is now used to analyze the behaviour with respect to native and non-native speech. Results are reported in Table 6. Here, the DNN model used to estimate the x-vector(speaker) embedding has been trained jointly on the VoxCeleb and Verbmobil corpora.

Table 6: WER (%) on the test set of the revised Verbmobil split with respect to the training data and the type of speaker embedding used.

Training data	Embedding	Test WER	
		Native (US)	Non-native (DE)
Native speech only	None	13.7 %	35.3 %
	i-vector	12.2 %	33.2 %
	x-vector(speaker)	12.2 %	31.7 %
Native speech + 1 hour of non-native speech	None	12.9 %	24.6 %
	i-vector	11.8 %	22.7 %
	x-vector(speaker)	11.5 %	25.4 %

The speaker embeddings provide a larger performance improvement for the acoustic models trained on native speech only. The behavior of speaker embeddings is less consistent on the non-native test data when a small amount of non-native speech is included in the training data. However, including a small amount of non-native speech in the training data leads to a large improvement of the STT performance on non-native test data.

Another set of experiments has been conducted using the data described in Table 2, which includes native and non-native data from Verbmobil, and several accented speech data from VoxForge. The results are reported in Table 7.

The results show that when dealing with native, non-native and accented speech, adding the x-vector(accent) embedding which characterises the accent present in the utterance leads to the best STT performance. Also, adding a small amount of non-native and accented speech in the training data leads to significant improvement in STT performance. Surprisingly, the STT performance is better on the English non-native speech (uttered by German speakers) than on the English accented speech (British, Indian and Australian). This is likely due to a difference in acquisition conditions (between Verbmobil data and Voxforge data), and a domain mismatch, hence a language model that better matches the Verbmobil data than the Voxforge data. Indeed, with respect to the language model used, the perplexities of the various test sets are 29.8 for non-native (DE) data, 117.3 for British accent (UK) data, 195.3 for Indian accent (IN) data, and 137.2 for Australian accent (AU) data.

Table 7: WER (%) achieved on native (US), non-native (DE) and accented speech (UK, IN, and AU), for different embeddings and training conditions.

Model	Embedding	Additional data	Native	Non-native or accented speech				
			US	all	DE	UK	IN	AU
GMM/HMM + fMLLR	None	none	19.5	52.9	38.6	46.2	65.8	53.9
DNN	None		13.6	45.3	35.2	43.4	56.9	45.1
DNN	i-vector		12.4	39.9	32.6	36.8	51.6	39.5
DNN	x-vector(accent)		12.1	38.5	31.5	38.3	45.2	38.1
GMM/HMM + fMLLR	None	1 hour per accent	20.6	45.5	29.2	42.8	61.4	46.5
DNN	None		13.5	40.4	24.1	40.6	55.4	39.3
DNN	i-vector		12.3	36.1	24.6	36.3	45.1	34.6
DNN	x-vector(accent)		11.8	34.5	23.3	36.1	43.7	36.5

4.2.2. Window length for speaker embedding

Figure 4 displays STT results on the test set of Verbmobil (revised split) with respect to the minimum length of the utterances and to the size of the windows used to compute the x-vector(speaker) embedding for training and for test. The horizontal axis refers to the minimum duration of the utterances. For example, 100 on the horizontal axis corresponds to considering all the utterances that are longer than 100 frames, that is longer than 1.0 s. With respect to the curves ‘train-10’, ‘train-40’, ‘train-100’, and ‘train-400’, the number specifies the length (in frames) of the window used to compute the embedding vector during training. For example ‘train-100’ means that, in the training data, the embedding is computed on 100 frames (i.e., 1.0 s), and updated every 100 frames. For these curves, the horizontal axis also specifies the length of the window used to compute the vector embedding on the test data. For example, for the horizontal mark ‘100’, the first embedding vector is computed on 100 frames, and then the embedding vector is updated every 100 frames.

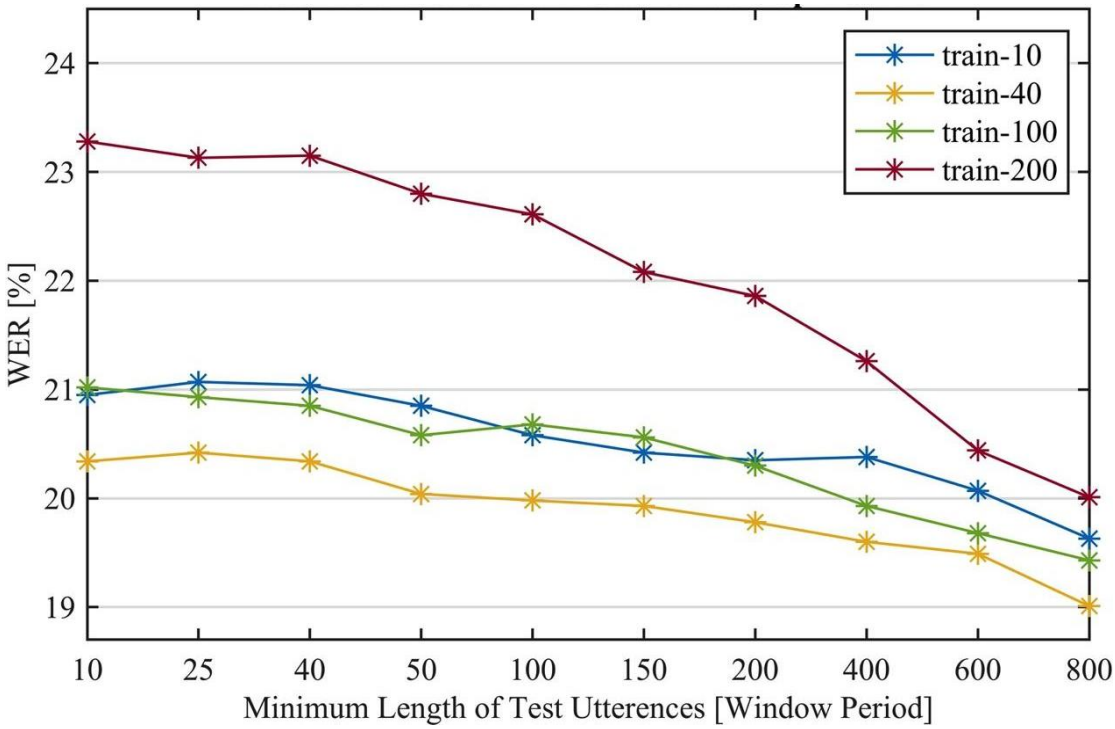

Figure 4: WER (%) with respect to the minimum length of the utterances (horizontal axis) and the window size used to compute the speaker embedding (curves).

Figure 4 shows that results improve for longer utterances and when the embedding is computed on longer time windows (the 95% confidence interval varies from ± 0.6% when considering utterances longer than 10 frames, to ± 0.8% when considering utterances longer than 800 frames). The improvement on longer utterances can be explained by the fact that, for long time windows, more speech data is available to estimate the speaker embedding. With respect to the curves, two facts have to be taken into account. When the window used to compute the speaker embedding gets longer, the embedding gets more reliable, and this should lead to better performance. However, it must be noted that the embedding is computed over a segment, and this embedding vector is associated to all the frames of the segment. In the train-100 case for instance, a first vector embedding is computed using the 100 first frames, and then associated with the spectral features of these frames. Then for the next 100 frames, a new embedding is computed using the 200 first frames, and associated with the spectral features of the frames 101 to 200, and so on. So using long windows for computing the speaker embeddings drastically reduces the number of speaker embedding vectors, and thus the variability of the training data. This explains why the best results are observed for intermediate durations of the window used to compute the embedding.

One should also keep in mind that using long time windows to compute the embedding introduces delays which may not be compatible with real-time interaction. Nevertheless, since this provides more reliable speaker information, it may be interesting to investigate this issue in dialog situations, for example by taking also into account previous utterances of the speaker.

4.2.3. Evaluation using privacy transformed speech data

COMPRISE has proposed a privacy-driven transformation to be applied to speech signals to hide the speaker identity. Here, we evaluate the performance on privacy transformed speech data using the Vocal Tract Length Normalisation (VTLN) based transformation that was described in Deliverable D2.1. Among the target speaker selection strategies proposed, we use strategy 2, which selects a target speaker at random for each source speaker. Results are presented in Table 8. VTLN transformation is applied to both training and test data.

Table 8: Performance on VTLN transformed data of the test set of the revised Verbmobil split with respect to the training data and the type of speaker embedding used.

Training data	Embedding	WER on test data	
		Native (US)	Non-native (DE)
Native speech only	None	17.9 %	46.8 %
	i-vector	15.7 %	41.2 %
	x-vector(speaker)	16.2 %	39.9 %
Native speech + 1 hour of non-native speech	None	19.0 %	37.4 %
	i-vector	16.7 %	29.4 %
	x-vector(speaker)	16.1 %	33.4 %

We observe the same behavior on privacy transformed speech that on original speech (cf. Table 6). Adding speaker embeddings improves the STT performance. In addition, including a small amount of non-native speech in the training data provides a large reduction of the WER on non-native speech.

4.2.4. Inclusiveness

We now investigate the benefit of personalisation on easy-to-recognise and difficult-to-recognise speakers. To do this, we rank the 25 speakers of each test subset according to the corresponding STT performance with the DNN model that does not use speaker embeddings. The 5 speakers that have the best score constitute group G1, the next 5 speakers constitute group G2, and so on. We then observe the impact of personalisation through the usage of x-vector(speaker) embedding on each subset of speakers. Table 9 reports these results for both models trained on native speech only (middle part of the table), and for models trained on native and non-native data (right part of the table).

Table 9: WER (%) and relative WER reduction with respect to speaker groups. The 25 speakers of each test subset are split into 5 groups according to the WER obtained with the DNN model without speaker embedding.

Model	Group of speakers	Training on native data only		Training on native and non-native data	
		Native (US)	Non-native (DE)	Native (US)	Non-native (DE)
DNN no-embedding	G1	6.1	22.6	5.9	16.7
	G2	9.8	30.1	10.2	19.4
	G3	12.6	32.5	11.3	24.8
	G4	15.2	42.9	14.5	27.5
	G5	26.3	47.2	23.7	36.7
DNN x-vector(speaker) embedding	G1	6.3	21.7	6.1	17.2
	G2	9.1	28.2	7.5	21.9
	G3	10.8	30.3	10.2	25.7
	G4	14.7	36.5	14.3	27.4
	G5	23.5	43.6	21.7	33.1
Relative reduction of the WER	G1	-3%	4%	-3%	-3%
	G2	7%	6%	26%	-13%
	G3	14%	7%	10%	-4%
	G4	3%	15%	1%	0%
	G5	11%	8%	8%	10%

In Table 9, the first set of rows reports the word error rates for each group of speakers with the model that does not include speaker embedding. The second set of rows reports the word error rates with x-vector(speaker) embedding. The last set of rows indicates the reduction in word error rates due to the introduction of the speaker embedding. It is interesting to see that there is almost no modification of the word error rates for the first groups of speakers (row G1); these speakers are well recognised whether the embedding is used or not. Another interesting point is the improvement observed for the last groups of speakers (row G5). On average a 10% relative WER reduction is observed on both native (US) and non-native (DE) test data, whether non-native speech is included or not in the training data.

4.2.5. Semi-supervised learning

The last set of experiments reports the performance achieved in the semi-supervised setting, i.e., when the training data includes both transcribed native speech and non-native speech without any transcription. Table 10 reports our preliminary experiments on this setting, using only one hour of non-native speech data.

The first and last rows report the performance resulting from the use of the conventional supervised training procedure. In the top row, supervised training is applied on native speech only: this is our baseline. The bottom row reports the results when supervised learning is applied to both native and non-native speech data: this is our topline, i.e., the best result that could possibly be achieved if the non-native data were transcribed. The

middle row uses the same training data, but the non-native speech material (about 1 h of speech) is used without its transcriptions. It is interesting to note that semi-supervised learning further improves the WER on top of x-vector(speaker) embedding, and the performance in this setting is between the baseline performance (first row) and the target performance (last row).

Table 10: Preliminary experiments with semi-supervised learning. All models include x-vector(speaker) embedding.

Training process	WER on US (native)	WER on DE (non-native)
Training on native data only (supervised)	12.2%	31.7%
Training on native and non-native data (semi-supervised setting)	11.7%	30.2%
Training on native and non-native data (supervised)	11.5%	25.4%

This semi-supervised approach could be applied to much larger training datasets, as no transcriptions are required. This will be the goal of future evaluations.

5. Software library

This package defines an initial version of COMPRISE's personalised learning library. All the components are consistent with Kaldi tools and binaries. The initial library focuses on learning embeddings for different accents which can be used as additional inputs to the acoustic model. For this purpose, we use the TDNN library for acoustic modeling and the SRILM toolkit for language modeling. Both of them can easily be accessed inside the Kaldi ecosystem. Installation and usage instructions are given below.

5.1. Initial Steps

The software has been tested and verified with Kaldi patch number 2b30a1e43. Please check your version using hash files in your Kaldi installation by `git log -1 --format="%H"`. You can also check the versioning scheme from the official documentation. Make sure that you are using a version `>=5.5.594` due to the non-backward compatible changes.

5.1.1. Prerequisites

- The personalised learning scripts employ both the Verbmobil and VoxForge corpora. It is also possible to use other datasets after arranging them as in Kaldi preparation style.
- We use the Verbmobil corpus for native American English and German accented English. British, Australian and Indian accented speakers are extracted from VoxForge corpora.
- English dialogs can be accessible from Verbmobil-I and Verbmobil-II. This implies CDs 6, 8 and 13 for Verbmobil-I, and CDs 23, 28, 31, 32, 42, 43, 47, 50, 51, 52, 55, 56 for Verbmobil-II.
- Data from VoxForge is freely-available on their official project page. It is possible to use your own implementation for fetching the data or extract them using another Kaldi script.

- To install the SRILM toolkit, run the `tools/extras/install_srilm.sh` script located under your Kaldi folder.
- A python script is used for merging x-vector embeddings after extraction. For this purpose, a module for reading and writing Kaldi ark files is required. Install it by `pip install kaldiio`. You can also download it directly from its source.

5.1.2. Setup

- Modify the symbolic links of the *steps* and *utils* folders in the main directory to point to `egs/wsj/s5/steps` and `egs/wsj/s5/utils` respectively, under your Kaldi installation.
- If you use data other than 16 kHz sampling rate, modify relevant files in `conf/` accordingly.
- Prepare the `path.sh` file in this folder which points explicitly to `KALDI_ROOT` and `SRILM` installations as well as other common paths.
- Create a `cmd.sh` file in this folder based on your running queue for Kaldi. If you have GridEngine installed, you should also create the `queue.pl` file with arguments specifying where GridEngine resides.

5.2. Run the Library

(1) Data Preparation: `run_1_data_prep.sh`

This script creates directories in `data` which will store training and test sets, and language model files. The following files should be present for each accented speech: `text`, `spk2utt`, `wav.scp`, `spk2gender` (see the details here). Additionally, `extra_questions`, `nonsilence_phones`, `optional_silence`,`silence_phones` and `lexicon.txt` should be prepared under `data/local/dict`. If you have non-verbal lexical entities like `[noise]` or `[laughter]`, don't forget to specify them in `lexicon.txt` and `nonsilence_phones`. Language model parameters are automatically selected with respect to smallest perplexity in the test set.

(2) Train GMM-HMM models: `run_2_train_hmm.sh`

This is a required step for further TDNN training. At the end of this script, alignments of triphone models are calculated.

(3) Train TDNN models without embeddings: `run_3_mfcc.sh`

This is the baseline model for personalisation experiments. Both training and decoding is done without any speaker embedding. Instead only spectral features (MFCCs) are utilised for acoustic model training. Note that GPU usage is required for this script. In other words, it should be configured on a machine where `nvcc` is installed.

(4) Evaluate i-vector based personalisation: `run_4_mfcc_ivec.sh`

This model is used as comparison to our proposel model. It consists of two stages, namely `train_extract_ivec.sh` and `chain_mfcc_ivec.sh`. The former prepares and i-vector system and extracts embeddings for the training and test folders. Then, the latter script is about training and decoding *TDNN* models with *i-vector* embeddings.

(5) Evaluate x-vector based personalisation: `run_5_mfcc_xvec.sh`

The proposed accent adaptation scheme is specified inside this script. Again GPU usage is required. There are three main stages, namely `prepare_data_and_train_xvec.sh`, `run_xvec.sh` and `chain_mfcc_xvec.sh`.

1. `prepare_data_and_train_xvec.sh`: This script first arranges the data according to their accent types and then prepares neural network training examples using `get_xvector_egs.sh` script. Finally, it executes the training model of x-vector embeddings using the standard `nnet3` library of Kaldi.
2. `run_xvec.sh`: This script performs 512-dimensional x-vector extraction for the training and test splits. This is done by giving a segment file which identifies speech segments in the recordings. This operation is similar to the standard Kaldi binary `ivector-extract-online.cc` where x-vectors for every *n* frames are extracted by including all frames up to that point in the utterance. This is designed to correspond with what will happen in a real-time streaming decoding scenario. The x-vectors are output as an archive of matrices, indexed by utterance-id; each row corresponds to an x-vector. Finally, `merge_arks.py` file merges multiple x-vector frames defined in the segment file of an utterance into a single vector.
3. `chain_mfcc_xvec.sh`: This is similar to the i-vector system except for the embedding type. Once we have arranged x-vector frames just like i-vectors, it is possible to provide them as an additional input inside the acoustic model by modifying the `online-ivector-dir` option of Kaldi's training script, `steps/nnet3/chain/train.py`.

6. Conclusion

This deliverable has presented current developments and results with respect to the personalised learning library for STT processing. The developments are based on the Kaldi toolkit. Personalisation is achieved thanks to the introduction of speaker information at the input of a DNN-based acoustic model. Several types of embeddings have been investigated for the speaker information: i-vector, initially proposed in the context of GMM-based speaker recognition, and x-vector, that has also been proposed for speaker recognition but relies on a DNN. In its initial version, the DNN x-vector extractor is trained to recognise speakers. Here, we have proposed and evaluated a variant using a DNN trained to identify non-native and accented speech. Finally, a semi-supervised learning process is presented with the goal of exploiting larger sets of untranscribed data in the training process.

The proposed approaches are evaluated mainly on the Verbmobil data. Results are presented and discussed with respect to different aspects. The presented results show the benefit speaker embedding in acoustic modeling, whether the models are trained only on native speech data, or trained on both native and non-native speech data. The introduction of speaker embedding vectors leads to 10% to 15% relative WER reduction; and in most cases, x-vector embeddings yield better performance than conventional i-vector embeddings. Evaluation on original speech and privacy transformed speech leads to a similar behavior. Experiments also show that including the speaker embedding leads to a significant improvement on speakers that are the most difficult to be recognised with the baseline acoustic model. Finally, preliminary results in the semi-supervised setting are promising, and open the way to including larger sets of untranscribed speech in the training data.

7. Bibliography

[Celikyilmaz *et al*., 2015] A. Celikyilmaz, Z. Feizollahi, D. Hakkani-Tur and R. Sarikaya, “A universal model for flexible item selection in conversational dialogs,” in *Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2015.

[Dehak *et al.*, 2011] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel and P. Ouellet, “Front-End Factor Analysis for Speaker Verification”, in *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4), pages 788–798, 2011.

[Elfeky *et al.*, 2016] M. Elfeky, M. Bastani, X. Velez, P. Moreno and A. Waters, “Towards acoustic model unification across dialects,” in *Spoken Language Technology Workshop (SLT)*. IEEE, 2016.

[Hess *et al.*, 1995] W. Hess, K.J. Kohler and H.-G. Tillmann, “The Phondat-Verbmobil speech corpus,” in *European Conference on Speech Communication and Technology*, 1995.

[Hinton *et al.*, 2012] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath and B. Kingsbury, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[Huang *et al.*, 2014] Y. Huang, D. Yu, C. Liu and Y. Gong, “Multi-accent deep neural network acoustic model with accent-specific top layer using the KLD-regularized model adaptation,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014.

[Jain *et al.*, 2018] A. Jain, M. Upreti and P. Jyothi, “Improved accented speech recognition using accent embeddings and multi-task learning,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2018.

[Karafiát *et al*., 2011] M. Karafiát, L. Burget, P. Matějka, O. Glembek and J. Černocký, "iVector-based discriminative adaptation for automatic speech recognition." in *Automatic Speech Recognition and Understanding Workshop* (ASRU), IEEE, 2011.

[Kitza *et al.*, 2018] M. Kitza, R. Schluter and H. Ney, “Comparison of BLSTM layer specific affine transformations for speaker adaptation,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2018.

[Ko *et al.*, 2015] T. Ko, V. Peddinti, D. Povey and S. Khudanpur, “Audio augmentation for speech recognition,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2015.

[Kumar *et al.*, 2015] K. Kumar, C. Liu, K. Yao and Y. Gong, “Intermediate-layer DNN adaptation for offline and session-based iterative speaker adaptation,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2015.

[LI *et al*., 2002] Y. Li, H. Erdogan, Y. Gao and E. Marcheret. « Incremental on-line feature space MLLR adaptation for telephony speech recognition.” In *Seventh International Conference on Spoken Language Processing (ICSLP)*, 2002.

[Li *et al.*, 2018] B. Li, T. N. Sainath, K. C. Sim, M. Bacchiani, E. Weinstein, P. Nguyen, Z. Chen, Y. Wu and K. Rao, “Multi-dialect speech recognition with a single sequence-to-

sequence model," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2018.

[Manohar *et al.*, 2015] V. Manohar, D. Povey, and S. Khudanpur, "Semi-supervised maximum mutual information training of deep neural network acoustic models," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2015.

[Manohar *et al.*, 2018] V. Manohar, H. Hadian, D. Povey and S. Khudanpur, "Semi-supervised training of acoustic models using lattice-free MMI," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4844–4848.

[Nagrani *et al*.] A. Nagrani, J. S. Chung, W. Xie and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild." *Computer Speech & Language*, vol. 60, 2020.

[Nguyen *et al.*, 2017] T. S. Nguyen, K. Kilgour, M. Sperber and A. Waibel, "Improved speaker adaptation by combining i-vector and fMLLR with deep bottleneck networks," in *International Conference on Speech and Computer*. Springer, 2017.

[Peddinti *et al.*, 2015] V. Peddinti, D. Povey and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2015.

[Povey and Yao, 2011] D. Povey and K. Yao, "A basis method for robust estimation of constrained MLLR," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2011.

[Povey *et al.*, 2011] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer and K. Vesely, "The Kaldi speech recognition toolkit," in *Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2011.

[Povey *et al.*, 2016] D. Povey, V Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2016.

[Saon *et al*., 2013] G. Saon, H. Soltau, D. Nahamoo and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors." in *Automatic Speech Recognition and Understanding Workshop (ASRU)*, IEEE, 2013.

[Sarikaya *et al.*, 2016] R. Sarikaya, P. A. Crook, A. Marin, M. Jeong, J.P. Robichaud, A. Celikyilmaz, Y.B. Kim, A. Rochette, O. Z. Khan, X. Liu, D. Boies, T. Anastasakos, Z. Feizollahi, N. Ramesh, H. Suzuki, R. Holenstein, E. Krawczyk and V. Radostev, "An overview of end-to-end language understanding and dialog management for personal digital assistants," in *Spoken Language Technology Workshop (SLT)*. IEEE, 2016.

[Siniscalchi *et al.*, 2013] S. M. Siniscalchi, J. Li and C.-H. Lee, "Hermitian polynomial for speaker adaptation of connectionist speech recognition," in *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, pp. 2152–2161, 2013.

[Snyder *et al.*, 2018] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *International*

Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 5329–5333.

[Tomokiyo and Waibel, 2003] L. Tomokiyo and A. Waibel, “Adaptation methods for non-native speech,” in *Multilingual Speech and Language Processing*, 2003.

[Tsao *et al.*, 2011] Y. Tsao, R. Isotani, H. Kawai and S. Nakamura, “Increasing discriminative capability on MAP-based mapping function estimation for acoustic model adaptation,” in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2011.

[Viglino *et al.*, 2019] T. Viglino, P. Motlicek and M. Cernak, “End-to-end accented speech recognition,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2019.

[Voxforge, 2020] “Voxforge: an open and free speech corpus for speaker recognition,” http://www.voxforge.org, accessed: 2020-03-12.

[Waibel *et al.*, 1989] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano and K. J. Lang, “Phoneme recognition using time-delay neural networks,” in *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989.

[Weiss and Ellis, 2010] R. Weiss and D. Ellis, “Speech separation using speaker-adapted eigenvoice speech models,” *Computer Speech & Language*, vol. 24, no. 1, pp. 16–29, 2010.

[Xie *et al.*, 2019] X. Xie, X. Liu, T. Lee and L. Wang, “Fast DNN acoustic model adaptation by learning hidden unit contribution features,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2019.

[Xue *et al.*, 2014] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai and Q. Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition." in *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 22, pp. 1713-1725, 2014."

[Yang *et al.*, 2018] X. Yang, K. Audhkhasi, A. Rosenberg, S. Thomas, B. Ramabhadran and M. Hasegawa-Johnson, “Joint modeling of accents and acoustics for multi-accent speech recognition,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[Yao *et al.*, 2014] K. Yao, D. Yu, L. Deng and Y. Gong, “A fast maximum likelihood feature transformation method for GMM–HMM speaker adaptation,” *Neurocomputing*, vol. 128, pp. 145–152, 2014.

[Yoo *et al.*, 2019] S. Yoo, I. Song and Y. Bengio, “A highly adaptive acoustic model for accurate multi-dialect speech recognition,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019.

[Zhang *et al.*, 2013] X. Zhang, K. Demuynck and H. Van Hamme, “Rapid speaker adaptation in latent speaker space with non-negative matrix factorization,” *Speech Communication*, vol. 55, pp. 893–908, 2013.

[Zhang and Woodland, 2016] C. Zhang and P. Woodland, “DNN speaker adaptation using parameterised sigmoid and relu hidden activation functions,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016.

[Zhao *et al.*, 2018] Y. Zhao, J. Li, S. Zhang, L. Chen and Y. Gong, “Domain and speaker adaptation for Cortana speech recognition,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.