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 Nº3.4: Final personalised learning library

Lead partner: INRIA

Version Nº: 1.0

Date: 26/02/2021

European Commission

Document information	
Deliverable Nº and title:	D3.4 – Final personalised learning library
Version Nº:	1.0
Lead beneficiary:	INRIA
Author(s):	Tuğtekin Turan (INRIA), Denis Jouvet (INRIA), Thomas Kleinbauer (USAAR), Matthew Kuhn (USAAR), David Ifeoluwa Adelani (USAAR)
Reviewers:	Gerrit Klasen (ASCO), Dietrich Klakow (USAAR)
Submission date:	26/02/2021
Due date:	28/02/2021
Type[1]:	OTHER
Dissemination level[2]:	PU

Document history			
Date	Version	Author(s)	Comments
10/02/2021	0.1	Tuğtekin Turan, Denis Jouvet, Thomas Kleinbauer, Matthew Kuhn, David Ifeoluwa Adelani	Draft version
23/02/2021	0.2	Gerrit Klasen & Dietrich Klakow	Revised version following the reviewers' comments
24/02/2021	1.0	Akira Campbell & Emmanuel Vincent	Final version reviewed by the Coordinator and the project manager

[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 presentation, evaluation and analysis of model personalisation strategies for Speech-to-Text, Spoken Language Understanding and Dialogue Management. The associated software components are available to the public on the COMPRISE git repository.[3] In this document, two different aspects are considered: one relates to the acoustic modelling for Speech-to-Text, and the other one concerns text-related processing.

The acoustic modelling is enhanced through the introduction of an accent embedding vector along with the conventional spectral features. This approach was detailed in Deliverable D3.2, and is briefly recalled here and evaluated on Latvian data.

With respect to text-related processing, it is important to recall that all text data which is collected on the COMPRISE Cloud Platform has been anonymized using the COMPRISE Text Transformer. Hence, this deliverable investigates the impact of training Speech-to-Text language models, Spoken Language Understanding models, and Dialogue Management models on such transformed data, compared to more classical training carried out using original data. Finally, approaches for improving these models through adaptation on a limited dataset of original text are presented, evaluated and discussed.

[3] https://gitlab.inria.fr/comprise/deliverables/deliverable-d3.4

Table of contents

1 Introduction

The objective of Work Package 3 is to enable any user to interact with dialogue 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 (TTS), on combining machine translation with the components of a dialogue system, and on adapting these models to a user. Deliverable D3.4 is concerned by this last point and deals with personalised learning approaches. More precisely, Deliverable D3.4 describes the personalised learning library for STT, SLU (Spoken Language Understanding), and DM (Dialog Management). Two types of personalisation processing have been investigated: one relates to the acoustic modelling for STT, and the other one relates to text processing, and stems from the fact that initial (user-independent) models are trained from privacy-preserving text transformed data.

With respect to the personalisation of the Acoustic Model (AM) for STT, Deliverable D3.4 keeps with the approach proposed in Deliverable D3.2 (Submitted to the European Commission on April 30, 2020 − Public).[4] As the STT in the COMPRISE library is based on Kaldi [Povey *et al*., 2011], the investigated personalised learning approach also uses the Kaldi toolkit and framework. The proposed approach relies on providing information on the speaker's accent, through the introduction of an x-vector accent embedding along with the usual spectral features. In Deliverable D3.2, the approach was evaluated on the Verbmobil corpus [Hess *et al*., 1995], and on accented English speech from the Voxforge project.[5] We evaluate it here on Latvian data.

The other personalisation aspects that we have investigated relate to text processing, and to the fact that the data collected by the COMPRISE Cloud Platform has been transformed via the COMPRISE Text Transformer to preserve privacy. This implies that user-independent models for STT Language Modeling (LM), SLU and DM are trained on transformed data rather than the original (untransformed) data. This deliverable investigates how much this impacts the model performance, and personalisation approaches are studied to investigate how the use of some original, typically user-specific data can help improve models trained on transformed data.

This deliverable is organised as follows. Section 2 recalls the text transformations that have been proposed and investigated in COMPRISE for preserving privacy, and introduces the resulting problem for LM, SLU and DM model training. Section 3 explains where the computations take place within the COMPRISE architecture. Section 4 presents and discusses the model personalisation strategy for STT, both for the AM and for the LM components. Section 5 presents and discusses the model personalisation strategy for SLU and for DM. Finally, the main functionalities of the software library are described in Section 6.

2 Privacy-preserving text transformations

The goal of Task 3.3 "Personalised privacy-preserving learning" is to compensate for the possible reduction in model quality incurred by the use of privacy-transformed training data. In COMPRISE, privacy-preserving text transformations have been developed as part of WP2 and implemented in the COMPRISE Text Transformer. They are briefly recalled here to remind readers of their specificities and to provide the relevant context.

[4] https://www.compriseh2020.eu/files/2020/05/D3.2.pdf
[5] http://www.voxforge.org

2.1 Named Entities

It is clear that any method to detect private information in text is prone to mistakes. This could lead either to uncritical information getting replaced, or worse, private information getting leaked. Knowing the limitations of privacy-preserving methods is thus crucial. In COMPRISE, we have addressed this problem in two ways: first, we focus on the expression of private information in the form of Named Entities; and second, we have developed a replacement strategy that can provide privacy even under circumstances where some of the private information has not been correctly identified by our recognisers.

Named Entities are expressions in text that refer to people, objects, locations, etc. with specific names, rather than generic words. For instance, a certain person could be referred to as "Jane" or as "the woman on the right", where the first case qualifies as a Named Entity, while the second case does not. It seems intuitive that Named Entities are more susceptible to carrying private information than the more generic variants, and therefore most state-of-the-art tools for detecting private information in text use automatic Named Entity Recognition (NER). However, it must also be noted that there are ongoing efforts to detect descriptive, non-Named Entity, references in text as well, for instance in the Horizon 2020 project MAPA.[6]

Despite the formal guarantees we provide (see p.14 of Deliverable D2.2, submitted to the European Commission on April 30, 2020 − Public),[7] some application domains may require that absolutely no original piece of information relating to the identity of a person remains in the transformed document, for instance for confidentiality or legal reasons. This is often the case in highly critical domains, such as medical patient records. However, such an absolute standard is very difficult to achieve, since even the most thorough human editor is prone to occasionally missing an instance of personal information in a document. Automatic means, even the COMPRISE solution, cannot reach 100% accuracy either, and therefore serve best as pre-processing tools in situations with such strict requirements.

2.2 Replacement strategies

Once private information has been identified in a document, there are different ways to handle it. The most radical way, completely deleting such words, would lead to distorted, ungrammatical texts in most cases. Instead, in COMPRISE, we have considered different ways to replace the identified private words with uncritical alternatives as listed in the following replacement strategies. In all cases we assume that it is non-trivial for an attacker to infer the original tokens that were replaced.

Redact. In this strategy, the private tokens are replaced with a non-word placeholder that is typically not part of the vocabulary of the source text, e.g., ■■■■■. From a privacy perspective, what perhaps seems to be the strictest form of protection might actually be less desirable because of the less-than-perfect recognition rates of automatic systems: an attacker can decide with certainty which of the tokens were part of the original text, since the replacements stand out clearly.

Typed placeholder a.k.a. value-class membership [Tang *et al*., 2004]. This strategy uses private category markers like LOCATION as the replacement token. It is similar to Redact, and provides the same level of privacy. However, it also provides additional

[6] https://mapa-project.eu/
[7] https://www.compriseh2020.eu/files/2020/05/D2.2.pdf

information about a replaced token's category and might thus be more useful than Redact for certain Natural Language Processing tasks for which the transformed data is used.

Named placeholder. A fixed category exemplar is used to replace all private tokens of that category [Pestian *et al*., 2007], e.g., all locations are replaced by "London". This strategy makes it slightly more difficult to judge which sentence was transformed and which was not, but for all instances that differ from the exemplar, it is clear that they must have been part of the source.

Word-by-word replacement. Here, the words in questions are replaced with other words, randomly chosen from a set of words of the same category. We can distinguish between *value distortion* if the replacement tokens are from an external source, and *value dissociation* when the surrogate tokens are from the same corpus. The latter keeps the distribution of tokens in the resulting document unchanged, which might be relevant for some tasks. Both variants make it hard to identify transformed sentences, which is reflected in lower values.

Full entity replacement. Text coherence could be improved if source tokens were consistently replaced by the same surrogates. One downside of the word-by-word strategy is that multi-word expressions could lead to nonsensical replacements when each contained word is replaced individually, e.g., "Frankfurt Airport" could be transformed to "New Francisco". A variant is thus to replace full entities instead of single words. This strategy does not necessarily lead to a higher level of privacy but the expected gain in coherence might benefit downstream tasks.

The following table illustrates the result of applying each of the outlined strategies.

Table 1: Illustration of various privacy-preserving text transformations.

Original	Hi, Mister Miller, the Lufthansa flight from Frankfurt Airport to Rome is leaving at six pm.
Redact	Hi, Mister ■■■■■, the ■■■■■ flight from ■■■■■ ■■■■■ to ■■■■■ is leaving at ■■■■■ ■■■■■.
Typed placeholder	Hi, Mister PER, the ORG flight from LOC to LOC is leaving at TIME.
Named placeholder	Hi, Mister Smith, the SAP flight from London to London is leaving at night.
Word-by-word replacement	Hi, Mister John, the Bosch flight from New Boston to Berlin is leaving at eleven morning.
Full entity replacement	Hi, Mister John, the Bosch flight from New York to Berlin is leaving at twelve pm.

2.3 Impact on LM, SLU and DM models

In the operating branch of COMPRISE, STT is followed by one or more text processing steps. In a full dialogue system, these can be grouped into SLU, which includes tasks such as intent classification, and DM, in which the system decides which actions to take in reaction to the speaker input.

Some of the challenges faced by STT and text processing are similar to each other albeit manifested in different forms, while other aspects are typical only for one of the two processing parts. For instance, regional dialects provide a challenge for STT components mostly through variations in the pronunciation. Other differences exist, too, such as grammar or choice of words, but the challenges brought about by these effects are comparatively minor. The main burden for dealing with dialects and accents thus lies on the shoulders of the STT component.

On the other hand, like STT, all text processing components in COMPRISE must be ready to work with models trained on privacy-preserving transformed data. Since the training branch is separate from the operating branch, and happens first and foremost in the COMPRISE Cloud Platform, GDPR requirements demand that all data is anonymised before it can be made available for cloud-based learning algorithms. Consequently, since the models are trained on data that differ from actual usage data, there is a risk that they perform below their potential when deployed in the actual application.

Therefore, we are interested in studying the magnitude of this performance degradation for LM, SLU and DM tasks, and investigating what countermeasures can be provided to compensate for it. The core idea is that, besides the privacy-transformed data available in the COMPRISE Cloud Platform, each user has their untransformed text data available that could be processed locally on the user's device or their instance of the COMPRISE Personal Server (see Deliverable D4.3, submitted to the European Commission on August 31, 2020 − Public)[8] to boost the performance of the component models.

3 Architectural context

An overview of the COMPRISE architecture is provided in Figure 1, with the overall positions of the STT, SLU and DM components.

The personalisation of STT AMs translates into the two "Speaker vector computation" blocks, and their use by the subsequent "STT learning" and "STT" blocks. The "Speaker vector computation" module computes an x-vector accent embedding which is handled by the AM along with the conventional spectral features and helps personalise the AM. At run time, the x-vector accent embedding is computed on the user's device or COMPRISE Personal Server and is never sent to the COMPRISE Cloud Platform.

Concerning the text processing for LM, SLU and DM, an initial "generic" user-independent model is first trained using text that has been transformed for privacy preservation. This deliverable investigates how using some original text data (i.e., untransformed text) can help improve the corresponding models.

[8] https://www.compriseh2020.eu/files/2020/08/D4.3.pdf

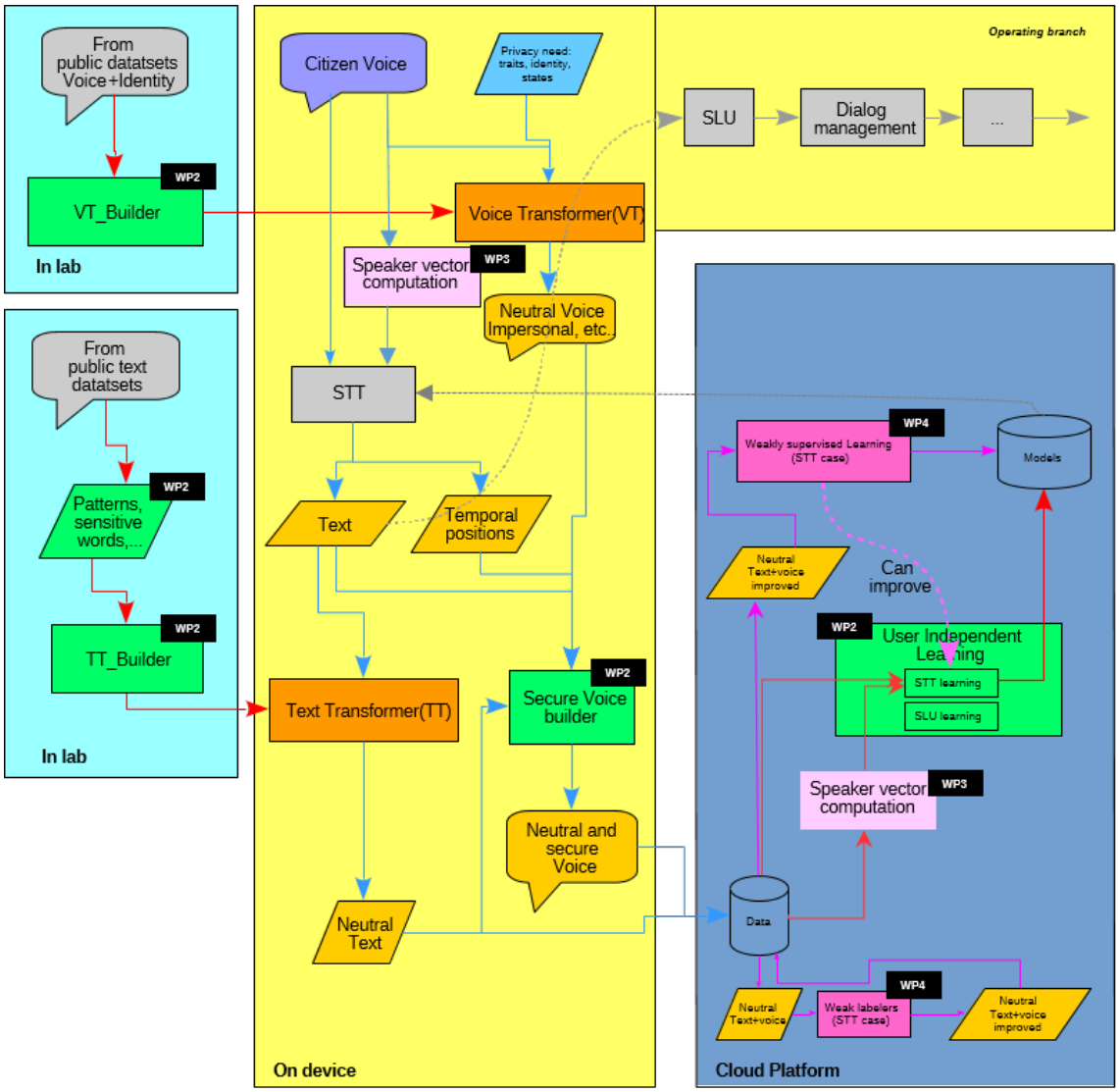


Figure 1: Overview of STT training and decoding in COMPRISE.

4 Model personalisation strategy for STT

This section presents the model personalisation strategy for STT. Two aspects are considered: AM personalisation and LM adaptation. Section 4.1 provides additional evaluation of the personalisation of the AM that has been proposed and described in Deliverable D3.2. The personalisation is achieved through the use of an x-vector accent embedding, that characterises the speaker's accent, along with the conventional spectral features. The additional evaluation is carried out on Latvian data. Regarding LMs, the initial models are trained on privacy-preserving transformed text data, so Section 4.2 investigates the adaptation of such LMs using a limited amount of original text data.

4.1 Acoustic model personalisation

Current STT systems trained on native speech often perform poorly when applied to non-native or accented speech. In our previous work (cf. Deliverable D3.2), we proposed to compute x-vector-like accent embeddings and use them as auxiliary inputs for the AM in order to improve the recognition of multi-accented English data containing native, non-

native, and accented speech. In this section, we review this idea and report new experiments using accented Latvian speech data.

4.1.1 Speaker embedding

In Deliverable D3.2 we investigated AM personalisation approaches that are based on making the AM speaker-aware. This was achieved by providing speaker specific information at the input of the AM neural network, along with conventional spectral features. To do this, we considered various types of speaker embeddings.

I-vectors, initially developed for speaker recognition, have since been largely used in other speech processing tasks, including speech recognition [Saon *et al*., 2013].

X-vector speaker embeddings have been introduced in [Snyder *et al*., 2018]. They are computed using a neural network trained to recognise speakers from a speech segment. The lower layers correspond to a time delay neural network that processes short-term frame contexts. Then a pooling layer computes statistics over the whole speech segment, which are processed by a few fully connected layers to classify speakers. The embedding is obtained from the last hidden layer.

X-vector accent embedding is another approach that we have proposed in Deliverable D3.2 and [Turan *et al.*, 2020]. It is very similar to the x-vector speaker embedding approach, except for the last layer that classifies (recognises) accents, instead of speakers. The training of this model requires accented data with accent labels for training, but the data doesn't need to be transcribed. The STT AM is then trained on transcribed data, but this data doesn't need to have accent labels.

4.1.2 Evaluation on non-native and accented English data

We have evaluated our method on native, non-native, and accented English speech. For the native and non-native data, we used the Verbmobil corpus which contains spontaneous speech from meeting scheduling dialogues [Burger *et al*., 2000]. In Verbmobil, we selected American English dialogues as native data and English dialogues from German speakers as non-native data. We also gathered British, Indian, and Australian non-professional accented speech recordings from the VoxForge project. We created training, test, and adaptation sets for AM training using disjoint sets of speakers (cf. Table 2). For decoding, we use a 3-gram LM trained over the native data where decoding parameters are kept fixed for all experiments. In particular, we just focus on the AM adaptation and do not perform lexicon or LM improvements.

Table 2: Statistics of the speech data from Verbmobil and Voxforge corpora, used for training and evaluating the AMs.

Corpus	***Data Type***	***Training Set***		***Test Set***	
		# of Spk.	***Dur. (h)***	***# of Spk.***	***Dur. (h)***
Verbmobil	Native (US)	235	25.4	25	1.1
	Non-Native (DE)	25	1.0	25	1.1
Voxforge	British (UK)	25	1.0	25	1.2
	Australian (AU)	25	1.0	25	1.3
	Indian (IN)	25	1.0	25	1.4

We used Mel-frequency cepstral coefficients as inputs for the accent embedding network. A 512-dimensional accent embedding was extracted from the last hidden layer before the nonlinearity. Non-overlapping chunks of 0.5 s duration were utilised with an online extraction scheme. In other words, we extracted the accent embedding for each 0.5 s segment by combining all frames from the beginning of the utterance up to that point. The embedding network was trained on utterances of native (US) and accented (UK, IN, AU, DE) speakers in Verbmobil and VoxForge, excluding those belonging to the AM test set.

Table 3: WERs (%) achieved with different embeddings on native, accented, and non-native English data.

Embedding model	***British (UK)***	***Indian (IN)***	***Austr. (AU)***	***Non-native (DE)***	***Average accent and non-native***	***Native (US)***
No embedding	27.9	35.6	28.1	23.8	28.4	14.0
i-vector (speaker)	23.7	31.8	24.3	21.9	24.8	12.8
x-vector speaker	24.4	32.1	23.6	20.3	24.5	12.5
x-vector accent	22.2	30.3	21.2	20.1	23.1	12.4

Table 3 presents the word error rates (WERs) obtained on native, non-native, and accented English[9] speech when the data used for training the AMs include one hour of data per accent (cf. Table 2). We observe that introducing additional information through a speaker or accent embedding improves the performance of the model. On average, x-vector embeddings lead to a smaller WER than i-vector embeddings, and the best results are obtained with x-vector accent embeddings. The better speech recognition performance observed on non-native English data (uttered by German speakers) compared to English accented data (British, Indian, and Australian accents) is likely due to a difference in acquisition conditions between Verbmobil and Voxforge, and a domain mismatch with respect to the LM that better matches the Verbmobil data than the Voxforge data.

4.1.3 Evaluation on Latvian data

Latvian, also known as Lettish, is the official language of Latvia as well as one of the official languages of the European Union. There are about 1.3 million native Latvian speakers in Latvia and 100,000 abroad. Altogether, about 2 million people speak Latvian. There are three main dialects in Latvian: the Livonian dialect, High Latvian and the Middle dialect. Also, the history of the Latvian language has placed it in a peculiar position as it is spoken by a large number of non-native speakers as compared to native speakers.

We use the Latvian speech recognition corpus [Pinnis *et al*., 2014] which contains about 100 h of speech from 1,500 speakers (each having an accent label). For this evaluation we selected the accents that had the most examples (Russian, High Latvian and English). Labels for remaining accents were ambiguous and suffered from sparsity, therefore

[9] The results reported here (Table 3) are extracted from [Turan *et al*., 2020]. Due to a different split of the speech data, they differ from those reported in D3.2, but the behaviour of the various approaches is similar.

we combined them into a single label "*Other*". We created training and test sets for AM training using disjoint sets of speakers (cf. Table 4).

Table 4: Statistics of the data from TILDE's Latvian speech recognition corpus used for training and evaluating the AMs.

Accent	***Training Split***		***Test Split***	
	# of Spk.	***Dur. (h)***	***# of Spk.***	***Dur. (h)***
Native	1280	82.9	9	1.0
Russian	75	6.4	5	0.6
High Latvian	66	4.7	10	0.6
English	7	0.6	5	0.6
Other	57	1.6	4	0.1

Table 5 presents the WER obtained on native, non-native, and accented Latvian data. The three types of embeddings (i-vector, x-vector speaker, and x-vector accent) lead to similar performance, and the improvement with respect to the baseline (i.e., no embedding) is negligible compared to what was observed on the English data in the previous experiment. It is worth mentioning that the error rates observed on non-native Russian speech samples are a bit smaller than those observed on native speech samples. Also, the error rates on the accented data (High Latvian), and on the non-native data (English) are not very far from those observed on native Latvian speech. This is a good point for speech-based applications as it seems to indicate that all categories of speakers are well recognised. Further investigation of the Latvian data, as well as analysis of these approaches on some other languages, would be useful. However, there are very few accented and non-native speech corpora available.

Table 5: WERs (%) achieved with different embeddings on native, accented, and non-native Latvian data.

Embedding model	***Russian***	***High Latvian***	***English***	***Other***	***Average Test***	***Native***
No embedding	28.6	32.8	31.5	36.8	31.0	30.3
i-vector (speaker)	27.9	32.2	31.2	36.0	30.0	28.8
x-vector speaker	27.8	32.7	31.6	36.5	30.5	29.5
x-vector accent	28.5	32.7	31.8	35.6	30.5	29.1

4.2 Language model adaptation

STT LMs are typically trained on large text corpora from domains that are as similar as possible with the target domain. When there is a domain mismatch between the training data and the actual application domain, this results in lower STT performance. LM adaptation aims to adjust the parameters of any LM, using a limited amount of adaptation

data that represents the target domain, so that it performs better on that target domain [Bellegarda, 2004; McGraw *et al.*, 2016].

In the COMPRISE framework (cf. Figure 1), the text data that is collected on the COMPRISE Cloud Platform has been anonymised via a text transformation. Consequently, in this deliverable we focus on adapting an initial LM trained on anonymised data (cf. Figure 2, Stage - 2) using a limited amount of original (non-anonymised) data, that would be typically stored on the user's device or their COMPRISE Personal Server.

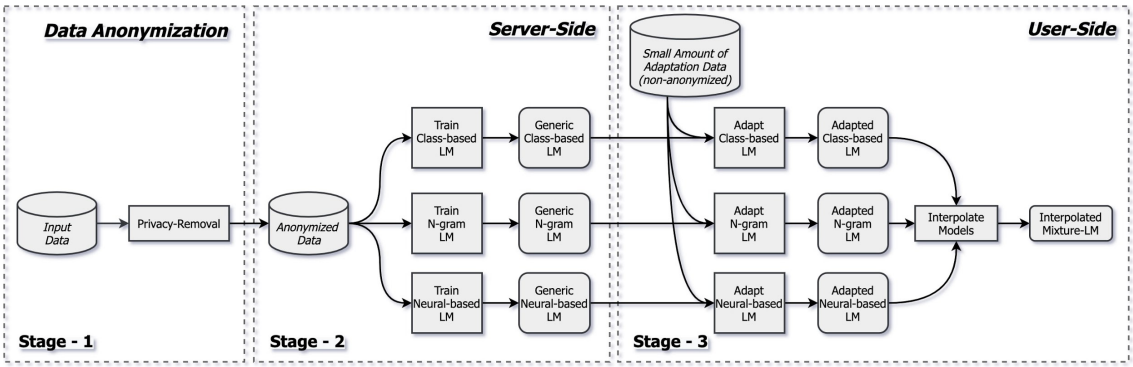


Figure 2: Privacy-preserving text transformation, training of initial LMs on anonymised data, and adaptation using limited amounts of non-anonymised data.

4.2.1 Adapting language models initially trained on anonymised data

An LM provides an estimate of the probability $P(w_k|h_k)$ of a word w_k knowing the history (previous words) h_k. These estimates allow the probability of a given sequence of K words, w_k $(1 \leq k \leq K)$ to be computed via the general expression

$$P(w_1, \ldots, w_K) = \prod_{k=1}^{K} P(w_k|h_k).$$

This expression can be approximated using a Markovian assumption, i.e., $h_k = w_{k-N+1}, \ldots, w_{k-1}$, to obtain an N-gram LM.

LMs are typically estimated on a large text corpus. Then, if needed, they are adapted to the target domain using a limited amount of adaptation data from the target domain. The specific settings within COMPRISE will be explained in the following section.

Figure 2 illustrates the processing of the text data, the training of initial (*generic*) LMs and their adaptation to the target domain. Stage 1 corresponds to the privacy-preserving text transformation which is applied on the text data before sending it to the COMPRISE Cloud Platform. Stage 2 corresponds to the training of the initial generic LMs using the transformed (i.e., anonymised) text data. This training will typically be carried out on the COMPRISE Cloud Platform. Stage 3 corresponds to the adaptation to the target domain using a limited amount of original (non-anonymised) text data.

For more details concerning privacy-preserving text transformations, refer to Section 2 above and to Deliverable D2.1 (Submitted to the European Commission on August 31, 2019 − Public).[10] In the experiments below, we have considered four categories of

[10] https://www.compriseh2020.eu/files/2019/08/D2.1.pdf

Named Entities, namely persons (PER), organisations (ORG), locations (LOC), and miscellaneous information such as date or time (MISC), and anonymisation has been performed using the word-by-word replacement approach.

Several types of LMs are investigated and combined: a conventional N-gram word-based model, an N-gram class-based model, and a neural network-based model. These models are first trained on the anonymised data, then adapted using a limited amount of non-anonymised data, and finally combined.

Conventional N-gram word-based language model

N-gram LMs look at the previous $N-1$ words to predict the *N*-th word in a sequence, based on (smoothed) counts of N-grams collected from training data. In the case of bigrams (i.e., $N=2$), the probability of a sequence of words is given by

$$P(w) = P(w_1, \dots, w_K) = P(w_1) \prod_{k=2}^{K} P(w_k | w_{k-1}).$$

Such LMs are rather simple, but very effective for STT applications.

In the experiments reported below (in section 4.2.2), we will use trigrams (i.e., $N=3$) and directly train the generic N-gram word-based model from the anonymised data.

N-gram class-based language model

Class-based N-gram models rely on an N-gram LM over word classes, and on the probability of individual words inside each class. If we consider non-overlapping classes and a bigram LM over the classes, the probability of a sequence of words is given by

$$P(w) = P(c_1)\, P(w_1 \mid c_1) \prod_{k=2}^{K} P(w_k | c_k)\, P(c_k | c_{k-1})$$

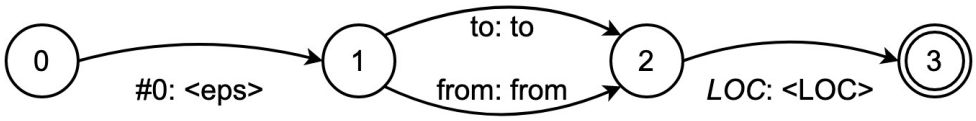
where $P(w_k \mid c_k)$ is the probability of word w_k inside class c_k, and $P(c_k \mid c_{k-1})$ is the probability of class c_k given the preceding class c_{k-1}.

In the experiments reported below (in Section 4.2.2), we consider a few "real" classes corresponding to Named Entities, such as persons (PER), organisations (ORG), and locations (LOC). The remaining words that are not associated with Named Entities are treated as degenerated classes, i.e., classes reduced to a single word.

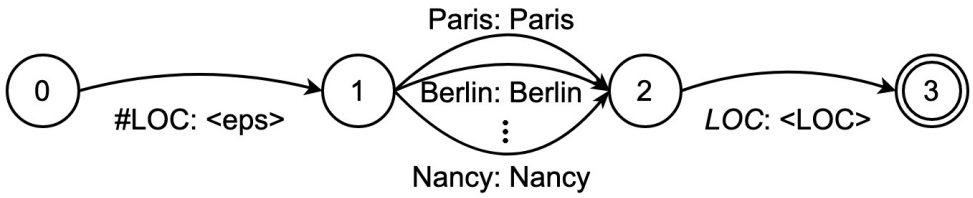
Since the Kaldi speech recognition toolkit[11] used for the STT experiments relies on Finite State Transducers (FSTs), a class-based LM can be built and used in the first decoding pass via FST composition. Four different FSTs are involved: H which defines the hidden Markov models through a mapping of transition-ids into context-dependent phones; C which maps context-dependent phones into context-independent phones; L which specifies the lexicon through a mapping of phone sequences into words; and G which is an acceptor (input and output symbols are the same) for encoding the LM. The final decoding graph is obtained after determinisation and minimisation of the combined FST denoted as HCLG. A class-based LM is obtained by composing the FST representing the N-gram over the classes with the FSTs representing the words inside individual classes (i.e., one FST per Named Entity class).

[11] https://kaldi-asr.org

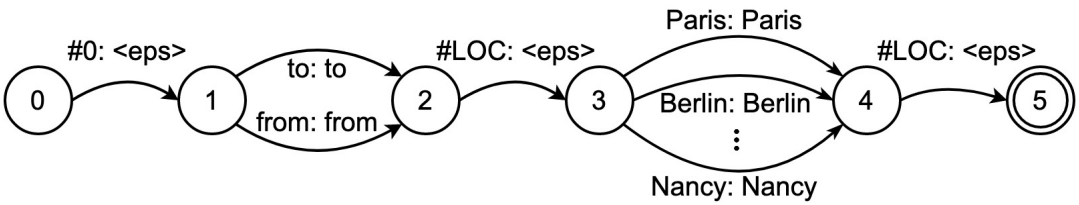
The process consists in first creating the FST corresponding to the N-gram of classes. The following graph[12] shows a toy example with two words ("to" and "from") and a class ('LOC'):



Then a simple FST is created for each class. The followings graph shows an example for the class "LOC" (location):



The composition of the FST representing the N-gram of classes with the FST associated with each class leads to the G graph that will be used by Kaldi:



Neural network-based language model

Neural network based LMs are now popular, and lead to efficient modelling when trained on large datasets. The recurrence of neural network models allows to take into account contexts that are longer than the $N-1$ words of the conventional N-gram models.

In the experiments reported below (in Section 4.2.2), we will use long short-term memory (LSTM) based LMs [Sundermeyer *et al*., 2012] that have proved to be efficient for such tasks.

Adaptation of language models

LMs can be adapted using a small amount of target domain data, as indicated in Figure 2, Stage-3. In the reported experiments, adaptation is catered by using original (non-anonymised) text data.

For what concerns the N-gram models, we follow the idea of marginal adaptation that was proposed in [Klakow, 2006] and outperformed linear interpolation. This combines the unigram and trigram information based on the fast marginal adaptation idea [Kneser *et al*.,1997]. The adapted probability values $P_{\text{adapted}}(w|h)$ are obtained by scaling the probability values $P_{\text{generic}}(w|h)$ of the generic model (trained from anonymised data) as

[12] '<eps>' is not a real word. It actually means "no symbol here", i.e., no output for the corresponding arc.

$$P_{\text{adapted}}(w|h) = \frac{1}{Z(h)}\left(\frac{P_{\text{adapt}}(w)}{P_{\text{generic}}(w)}\right)^{\beta} P_{\text{generic}}(w|h)$$

where β is a weighting parameter, $Z(h)$ is a normalisation term, $P_{\text{generic}}(w)$ is the unigram probability of the generic model, and $P_{\text{adapt}}(w)$ is the unigram probability estimated on the adaptation data. Therefore, this adaptation scales the probability of certain words up or down depending on whether they are more or less frequent in the adaptation data than in the initial training data (generic model).

The neural network based LM trained on the anonymised data is adapted through fine-tuning [Ma *et al*., 2017]. As transfer learning becomes easier and more effective with high-level abstract features [Wang and Zheng, 2015], only the last layer (softmax layer) of the neural network model is fine-tuned on the adaptation data in the experiments reported below.

Interpolation of language models

LMs can be used individually, or combined as indicated in Figure 2, Stage-3. The combination of several LMs is based on the conventional linear interpolation:

$$p(w|h) = \sum_{q} \lambda_q p_q(w|h).$$

N-gram approximation of neural network language models

The current version of Kaldi cannot use a neural network LM in the first decoding path. Hence, we have created an N-gram approximation of the neural network LM to be able to use it in the first pass decoding. In [Adel *et al*., 2014], several approximation techniques are presented. In the reported experiments, we used the probability-based conversion technique which provided good performance in [Singh *et al*., 2017]. For every word w_i of the training corpus (here, anonymised dataset), and associated history h corresponding to unigram, bigram, and trigram, we extract the neural-based LM probability $y(w_i|h)$. Then, these values are averaged (if multiple occurrences of $(w_i|h)$) and normalised to obtain a probability distribution:

$$p(w_i|h) = \frac{y(w_i|h)}{\sum_{w_k} y(w_k|h)}.$$

4.2.2 Experimental setup

In this deliverable, we evaluate LM adaptation schemes using the Augmented Multiparty Interaction (AMI) corpus containing multi-hour meeting recordings. These meetings were recorded as part of the AMI/AMIDA projects[13] by the University of Edinburgh and Idiap. The AMI Meeting Corpus is a collection of data captured in specially instrumented meeting rooms, which record the multimodal signals (audio and video) for each participant. This corpus contains both scenario and non-scenario meetings. In the scenario meetings, four participants play the role of a design team composed of a project manager, a marketing expert, a user interface designer, and an industrial designer. The meeting is supervised by the manager who follows an agenda with several items to be discussed with the other speakers. The non-scenario part corresponds to about two-thirds of the

[13] http://www.amiproject.org

data, and consists of samples of "real" meetings. The corpus is manually transcribed at different levels and we use the NITE XML Toolkit[14] to manage the annotations.

We partition the AMI data into training, adaptation, and test sets ensuring that no speaker appears in more than one set. Also, we only use speech data recorded with individual headsets. The following table presents some statistics of the data. The average length of the utterances is 7.5 words.

Table 6: Some statistics of the training, adaptation and test sets from the AMI corpus.

Set	Duration (min)	Utterances	Number of words	
			Unique	Occurrences
Train	4,880	108,221	11,882	802,604
Adaptation	580	13,059	4,145	94,914
Test	531	12,612	3,913	89,635
All	5,991	133,892	13,079	987,153

In this split the adaptation set represents 12% of the training data. Other splits representing 15% and 20% of the training data have been considered to investigate the impact of larger adaptation sets. In all cases, the test set is the same.

We use the provided Named Entity annotations whenever available. These annotations adhere to AMI's Named Entity instructions,[15] which mainly follow the hierarchical structure of the NIST task definition [Chinchor *et al*., 1999], but they are not available for all subjects or meetings. Therefore, the Named Entities in the remaining subjects or meetings are found using the open-source software spaCy,[16] which comes with pre-trained pipelines. After combining the pre-annotated AMI entities and spaCy extracted ones, we obtained 2,167 unique Named Entity tags including 226 LOC, 489 PER, 515 MISC, and 937 ORG.

During the experimental evaluation, we use WER and perplexity (PPL) as objective metrics. We also present statistical tests for deciding whether differences in error rates compared to the baseline model are significant. The "sc_stats" tool from NIST[17] is used to compute the matched pairs sentence-segment word error (MAPSSWE) test. This is a parametric test that looks at the numbers of errors occurring in units (segments of utterances) of varying sizes. MAPSSWE is essentially a t-test for estimating the mean difference of normal distributions with unknown variances [Gillick and Cox, 1989].

For the reported experiments, the AM was based on Kaldi's time-delay neural network (TDNN) chain architecture.[18] The TDNN-based AM operates on 40-dimensional Mel frequency cepstral coefficient (MFCC) features extracted from frames of 25 ms length and 10 ms stride and is similar to the model specified in [Peddinti *et al*., 2015]. The speed-

[14] http://groups.inf.ed.ac.uk/nxt
[15] http://groups.inf.ed.ac.uk/ami/corpus/Guidelines/NamedEntityInstructions.pdf
[16] https://github.com/explosion/spaCy
[17] https://github.com/usnistgov/SCTK
[18] https://kaldi-asr.org/doc/chain.html

perturbation technique of [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.

4.2.3 Results

The individual LMs used are the following: a 3-gram word-based model, a 3-gram class-based model, a 3-gram approximation of the neural network based model, and finally rescoring of the word lattice hypotheses resulting from first-pass decoding using a 3-gram word-based LM with the neural network LM.

Baseline results

We first evaluated the performance of the individual generic **LMs trained over the anonymised input text** at Stage-2 of Figure 2. These results can be regarded as our baseline performance. Table 7 shows the corresponding WER and PPL over the (untransformed) test data. For these WERs, the 95% confidence interval is equal to ±0.3%.

Table 7: Baseline performance obtained with individual LMs trained on anonymised data.

Model	*WER [%]*	*PPL*
3-gram word-based LM	32.3	121
3-gram class-based LM	**30.2**	103
3-gram approx. neural-network LM (1st-pass dec.)	32.9	137
Neural-network LM (2nd-pass rescoring)	30.5	103

The best results are obtained using either the class-based LM, or through rescoring with the neural-network LM.

Topline results

Table 8 presents the results using **LMs trained on the original data**, i.e., before applying the anonymisation process. These results can be regarded as our topline, and help understand the impact of anonymisation and how much of it is recovered through the proposed adaptation approaches.

Table 8: Topline performance obtained with individual LMs trained on original data.

Model	*WER [%]*	*PPL*
3-gram word-based LM	28.8	82
3-gram class-based LM	29.3	74
3-gram approx. neural-network LM (1st-pass dec.)	29.1	88
Neural-network LM (2nd-pass rescoring)	**27.6**	73

Compared to Table 7, we see a large difference in the WERs achieved with the 3-gram word-based models. Estimating the 3-gram word-based model on anonymised data leads to a 10% relative WER degradation, compared to training it on original data. The

best results are obtained when a second pass is applied for rescoring hypotheses with a neural network LM, but this increases the computational requirements, and induces an extra delay before getting the STT output.

Adding limited amounts of original data to the anonymised training data

We now measure the benefit of **adding limited amounts of original data** to the large set of anonymised data for training the LMs. Results are reported in Table 9 and show improvements for each type of modelling, and obviously, compared to the baseline models (Table 7), the WER gets smaller when the amount of additional non-anonymised data gets larger. The improvements are usually considered as statistically significant when the p-value (that results from a statistical test) is below 0.05.

Table 9: Performance obtained when adding limited amounts of original data to the anonymised training set for training the LMs.

Model	***12% Adapt. Data***			***15% Adapt. Data***			***20% Adapt. Data***		
	WER	***PPL***	***p-val***	***WER***	***PPL***	***p-val***	***WER***	***PPL***	***p-val***
3-gram word-based LM	31.9	116	*.008*	31.4	107	*.003*	29.6	90	*.014*
3-gram class-based LM	**30.2**	96	*.026*	**29.8**	93	*.012*	**29.3**	84	*.019*
NN LM (3-gram approx.)	31.4	106	*.034*	30.5	101	*.062*	29.9	92	*.027*
NN LM (2nd-pass)	30.3	98	*.019*	30.0	94	*.041*	29.5	81	*.058*

Adaptation of language models using limited amounts of original data

We also evaluate the benefit of **adapting LMs** initially trained on a large set of anonymised data by **using a limited amount of original data**. Results are reported in Table 10. The larger the adaptation set, the better the performance. For the smallest size considered here, the WERs achieved with the LM adaptation procedure are smaller than those achieved when the same amount of non-anonymised data is directly used in addition to the anonymised training set in a conventional LM training procedure (cf. Table 9).

Table 10: Performance obtained when adapting the LMs with limited amounts of original data.

Model	***12% Adapt. Data***			***15% Adapt. Data***			***20% Adapt. Data***		
	WER	***PPL***	***p-val***	***WER***	***PPL***	***p-val***	***WER***	***PPL***	***p-val***
3-gram word-based LM	31.5	109	*.004*	31.2	98	*.022*	31.0	93	*.031*
3-gram class-based LM	**29.9**	94	*.017*	**29.8**	91	*.044*	**29.7**	86	*.012*
NN LM (3-gram approx.)	30.8	101	*.038*	30.6	94	*.025*	30.3	90	*.057*
NN LM (2nd-pass)	30.1	95	*.042*	29.9	91	*.011*	29.8	85	*.009*

Interpolation of adapted language models

Finally, we evaluate the benefit of interpolating several adapted LMs obtained using 12% of adaptation data. The results are reported in Table 11. We first merge the 3-gram class-based and the 3-gram word-based LMs with the best weight combination: $\lambda_{\text{wordLM}} = 0.3$ and $\lambda_{\text{classLM}} = 0.7$. Then, we interpolate with the neural network LM. The final mixture LM is obtained as $0.6 \times (\lambda_{\text{wordLM}} \times P_{\text{wordLM}} + \lambda_{\text{classLM}} \times P_{\text{classLM}}) + 0.4 \times P_{\text{NeuralLM}}$.

Table 11: Performance obtained when linearly interpolating the adapted LMs.

Model	***WER***	***PPL***	***p-value***
3-gram word based + 3 gram class-based	29.7	95	*.006*
+ NN LM (3-gram approx. in 1st pass)	29.6	92	*.037*
+ NN LM (2nd-pass)	29.4	86	*.021*

4.3 Discussion and conclusion

For what concerns AM personalisation for STT, the introduction of speaker or accent embeddings along with the conventional spectral features proved to be efficient for dealing with accented and non-native English data, and the best performance was obtained using the proposed x-vector accent embedding. The new results reported in this deliverable on accented and non-native Latvian data show very small improvements when adding speaker embeddings, much smaller than the ones observed on English data. Further experiments on other languages or datasets would be useful to draw conclusions.

The second topic investigated for STT concerns the LM adaptation. Indeed, in the COMPRISE framework the initial LMs are trained from text data that has been anonymised. The experiments reported above show that using some original data helps improve the quality of these LMs, which translates into a smaller WER. Neural network LMs lead to good performance when used in a second pass decoding for rescoring lattices of word hypotheses. Using a 3-gram approximation of the neural network LM allows for its use in the first pass decoding, but loses the benefit of handling long histories. Combining several LMs leads to a further improvement of the speech recognition performance. The fact that the neural network LMs (used for lattice rescoring) yield perplexities that are not much better than those of the class-based LMs might be due to the limited amount of data available for training the models.

From a practical point of view, the 3-gram class-based LM yields good speech recognition performance, even for the initial model trained from anonymised data. Note that this class-based LM leads to a baseline WER of 30.2%, which is more than halfway between the baseline WER yielded by the 3-gram word-based model trained on anonymised data (32.3%) and the topline WER yielded by the same model trained on original data (28.8%). Few classes of words are used, which corresponds to the Named Entities that have been modified in the anonymisation process. A small improvement can be achieved by adapting this model using some original data, however the gain is limited and the amount of data needed is rather large for being provided by a single user. Although the WER reduction is statistically significant (p-value: 0.017), this limited improvement might not be noticed by the user of a voice-enabled application.

In addition, when we examine the proposed LMs in terms of computational complexity, we see that class-based LMs are an acceptable option. Although neural-based LMs can

outperform them in terms of WER and PPL, they require two-pass decoding and a bigger computational effort and bigger data for an effective solution. The proposed class-based LMs attain decent performance and are not more complex than the traditional N-gram LM training. Moreover, they can be used directly in single-pass decoding.

5 Model personalisation strategy for SLU and DM

In this section, we consider the SLU and DM components of COMPRISE. Similarly to Section 4 for STT, we investigate whether small amounts of original, user-specific data can be leveraged to improve SLU and DM models trained on privacy-transformed data. The intuition is that such transformations will likely have a negative impact on the performance of the resulting models since the similarity between the training data and the test data reduces after it was transformed.

We examine two representative tasks: Named Entity Recognition (NER) for SLU and intent detection for DM. These tasks were chosen because, while they both operate on sentences as input, one is a sequence labelling task and the other, a sentence classification task. We present an extensive number of experiments using multiple datasets.

5.1 Context: Privacy-preserving text transformations

Given the different anonymisation strategies listed in Section 2, we are interested in how transforming text data prior to their use for training impacts the performance of the resulting machine learning models. For SLU, we focus on the NER task because of its immediate relevance to privacy preservation.

To measure the impact of a small amount of original, user-specific data, we start out with an idealised setting in which there are no computational constraints on the user's side to train models. To a certain degree, this is an unrealistic assumption: today, many state-of-the-art Natural Language Processing models have extensive requirements in terms of computational resources (e.g., GPUs and memory) as well as for training time and power consumption. While such requirements will certainly be within reach for the COMPRISE Personal Server setup in the near future, they are currently prohibitive as on-device solutions for current mobile devices. However, as a lab condition, this setup provides us with an upper bound in terms of what could be achieved.

Our experimental setup is thus as follows. We combine transformed and original (a.k.a., personal) data at various ratios in order to derive a better understanding of the effect that original data can have on a model. This raises a practical issue, though: in a real-life situation, all original data would be from the same user (hence "personalisation") but this is not easy to replicate using available datasets, since user information is not always provided. It is therefore interesting to study whether small amounts of original data from *different* users lead to similar outcomes as small amounts of original data from *a single* user. This motivated the inclusion of the Verbmobil dataset into the SLU experiments, since it provides user information unlike the other datasets.

5.2 Personalisation for spoken language understanding

5.2.1 Datasets

For SLU, we evaluate the NER performance using two different corpora: Verbmobil and WNUT2016 [Strauss *et al.*, 2016]. The Verbmobil corpus contains dialogues in different languages. Here, we only use the English portion of the dataset, which contains 726

dialogues between two speakers each. The same speakers may partake in multiple dialogues, and so the number of unique speakers is 302. As Verbmobil does not come pre-annotated with Named Entity labels, about 20% of the dataset was annotated via crowdsourcing while the remaining 80% was first labelled automatically using spaCy and then post-corrected manually. The labels used are DATE, LOC, ORG, PER, TIME. After pre-processing and filtering, there are 28,052 annotated utterances in the dataset.

The WNUT2016 corpus is a collection of 7,251 tweets collected from Twitter.[19] The data is split in training/development/test splits of sizes 2,395 / 1,000 / 3,856, respectively. Its annotation is based on the following labels: company, facility, geo-loc, movie, musicartist, other, person, product, sportsteam, tv show.

5.2.2 Experiments

Because of its versatility and ease of use, even for non-expert users, we base our experiments on BERT [Devlin *et al*., 2019]. Different versions were used for the two sets of experiments: we used BERT-base-cased for the Verbmobil experiments, and BERT-large-cased for WNUT2016. For each of the described experimental conditions, the training data was used to fine-tune the model.

For the Verbmobil experiments, we set aside as our test set 400 utterances by the 5 users with the largest number of utterances in the dataset. The remaining utterances of these users form the pool from which we select different amounts of samples as the original (untransformed) portion of the training set, depending on the specific experimental condition. The utterances of all other users are processed using the COMPRISE Text Transformer and serve as the transformed portion of the training set. The transformation strategies considered here are Redact, Word-by-word replacement, and Full entity replacement.

Figure 3 displays the classification F1-scores for three speakers of the test set; the general trend is the same for all. The transformed portion of the training set remains the same for each experimental condition, the different bar plots for each speaker differ from each other in the number of original per-speaker samples that were added to the transformed portion for training. Each block of bars includes a topline (displayed in purple) for which the full training set was used without any transformation; in COMPRISE, this condition would not normally be available as all collected data undergoes a transformation before leaving the user's device or the COMPRISE Personal Server. In addition, we also show in blue how training only on the small amount of per-user original data (without using any transformed data) fares for the different amounts of original data considered. The leftmost block of bars for each speaker shows the outcome of using *only* transformed data for training.

Across all considered experiments, two aspects stand out. First, the Full entity replacement strategy in red is only minimally affected by the absence of user-specific data (see the left-most block of bars for each speaker), as the red bar is almost on par with the topline. As a matter of fact, adding a limited amount of original data does not lead to an improvement for that strategy. A similar behaviour albeit with a slightly lower F1-score can be observed for the Word-by-word replacement strategy in green.

[19] https://twitter.com

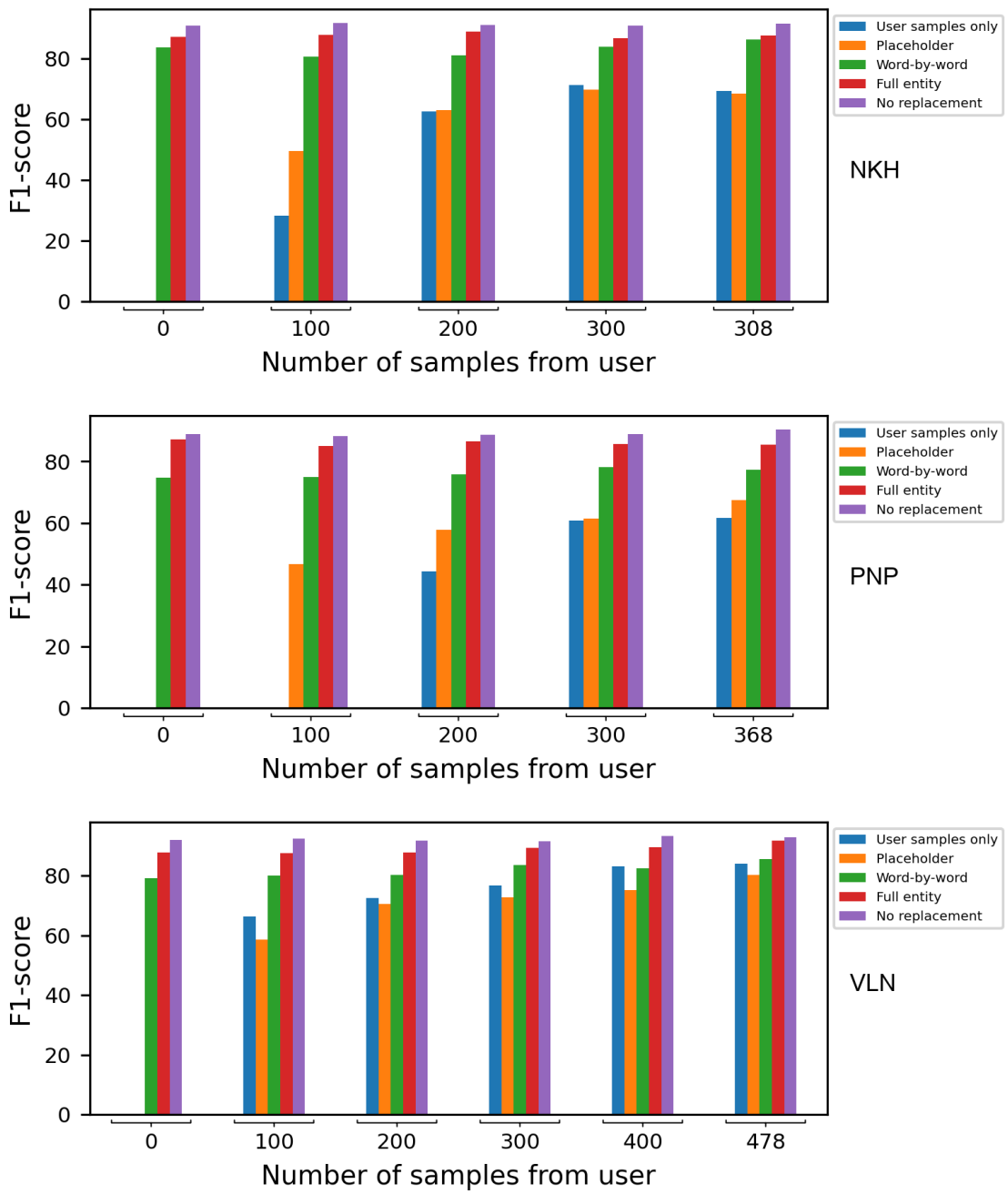


Figure 3: NER results achieved on the Verbmobil corpus by adding a limited amount of original data from one of three speakers (NKH, PNP, or VLM) to the transformed training data. For comparison, the topline performance achieved by training on original data only is shown in purple.

In contrast, the Redact condition fails with a F1-score of zero when only transformed data are in the training set (note the missing yellow bar in the left-most blocks of bars: the F1-score for the Redact strategy is 0 in that condition for all speakers). Recall that this strategy replaces all occurrences of private information with the same replacement token, independent of the Named Entity type, basically removing the most important information for the classification. At the same time, since the test set is untransformed, the replacement token never appears at test time, and so the model fails to recognise not only the type of Named Entity, but the presence of any Named Entity at all.

The WNUT2016 corpus presents a much more challenging task for NER because of its Twitter origin. Given the strict character limit of tweets, users commonly resort to using non-standard language, especially abbreviations. State-of-the-art pre-trained LMs are typically trained on standard language data, and this data mismatch is known to be problematic. Figure 4 shows the results of a number of different experimental conditions over the transformation strategies: Redact, Typed placeholder, Named placeholder (Exemplar), and Full entity replacement. As a topline, two conditions using only original data are included as well.

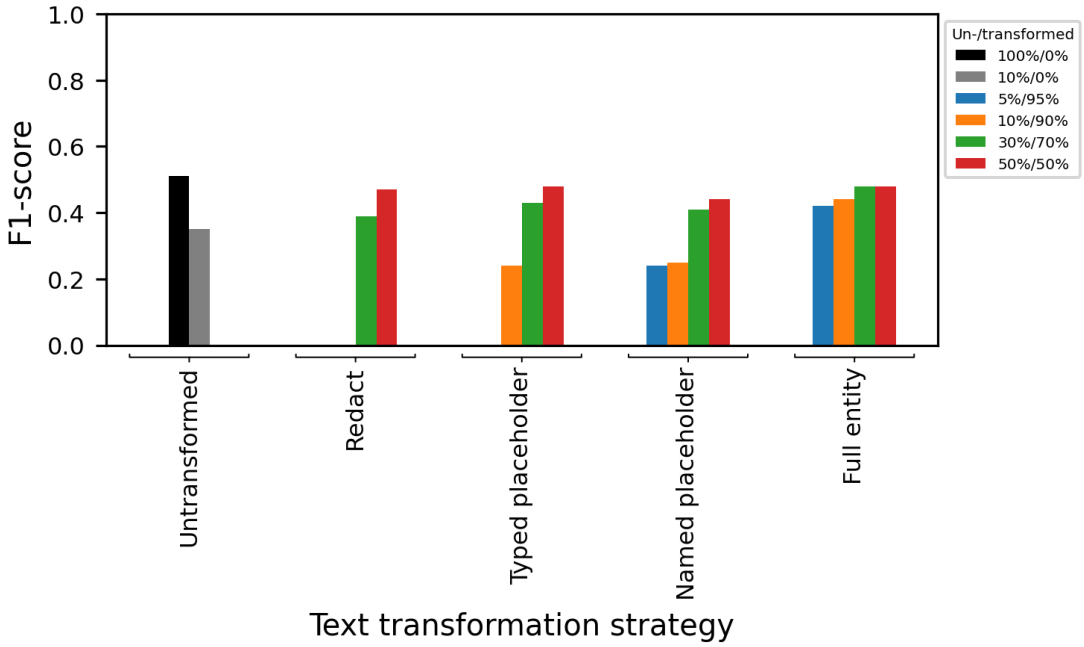


Figure 4: NER results (F1-score) achieved on the WNUT2016 corpus for different data splits.

The first observation is that the overall F1-scores achieved even in the topline conditions are much lower than for the Verbmobil dataset. As discussed above, this is likely due to the vocabulary and grammar mismatch between the data used to train BERT and the WNUT2016 data.

We note again that using only transformed data takes a large toll on the performance of the model, as witnessed by the 0% / 100% and 0% / 90% conditions. For the sampling condition, the former condition failed to train a useful model, while the latter did, likely hinting at a technical issue rather than a methodical one.

We also see that, like for Verbmobil, the Full entity replacement strategy proves to be the most robust against lack of original data. Adding personal data helps in all conditions, however, the amount of data required to reach acceptable performance is rather high: adding 10% original improves the performance only moderately in all conditions except Full entity replacement. This fact puts a question mark behind the general feasibility of personalisation based on single-user data alone.

5.3 Personalisation for dialogue management

5.3.1 Datasets

For DM, we evaluate the intent classification performance using two primary datasets: ATIS-2 [Hemphill *et al.*, 1990] and the SNIPS intent classification dataset [Coucke *et al.*, 2018]. Table 12 includes a break-down of the most relevant statistics of these datasets, divided between training, development, and test sets.

Table 12: Corpus statistics for ATIS-2 and SNIPS.

	ATIS-2			SNIPS		
	Train	Dev.	Test	Train	Dev.	Test
#Samples	4,478	500	893	13,084	700	700
#Labels	17	14	15	7	7	7
Vocabulary size	867	463	448	11,421	1,571	1,624
Avg. sentence length	11.28	11.41	10.26	9.00	9.12	9.08
Most frequent label	3,328	359	645	1914	100	124
Least frequent label	1	0	0	1818	100	80
Label counts (mean)	263.41	35.71	59.53	1,869.14	100	100
Label counts (std. dev.)	796.27	93.69	162.63	32.46	0.0	15.0

ATIS-2 presents several distinct advantages: it is an established and heavily researched dataset which makes it an excellent benchmark for performance tests. It does, however, have a few notable characteristics.

First, the items in the corpus were not obtained through interactions with an automated system but through a “Wizard of Oz” experiment in which human agents provided responses to the users. Although the users phrased their own queries (instead of reading from a script), the items in the corpus were transcribed by one of the “Wizards”, removing disfluencies and interpreting the query to provide a transcription of what the users meant to say.

Second, the dataset is relatively small; comprising only 5,871 items in the training, development, and test sets combined. The small number of samples is opposed by a relatively large number of class labels (17). The label distribution is also rather imbalanced: across all three sets, a single label, a request for flight information, makes up for the vast majority of the annotations (74% of the total size).

In contrast, the SNIPS dataset is larger, and it has fewer labels which are distributed more evenly across the data points. There is also more linguistic variation in the speaker turns. Since ATIS-2 and SNIPS contrast with one another in a number of ways, similar performance between the two datasets would suggest that the results are somewhat agnostic to the particular differences between them.

Unfortunately, both ATIS-2 and SNIPS do not provide user labels. For our experiments, we thus randomly selected a portion of the training set to remain untransformed, with the rest of the training set being transformed, in order to simulate the split between cloud data and local user data. These splits are identified by their proportion of original data to

transformed data. For instance, if an experiment uses as training data a mix of 10% original data with the remaining 90% being transformed, it is referred to as a 10% / 90% split.

5.3.2 Experiments

Our experiments for intent classification are based on a Bi-LSTM architecture, as displayed in Figure 5. The input is fed first into an embedding layer using GloVe 100d embeddings [Pennington *et al*., 2014] to encode the input into word vectors. The encoded vectors are then passed on to the Bi-LSTM layer. The outputs of the forward and backward passes of the Bi-LSTM are concatenated and passed to a fully-connected layer. The estimated label is the argmax over these outputs.

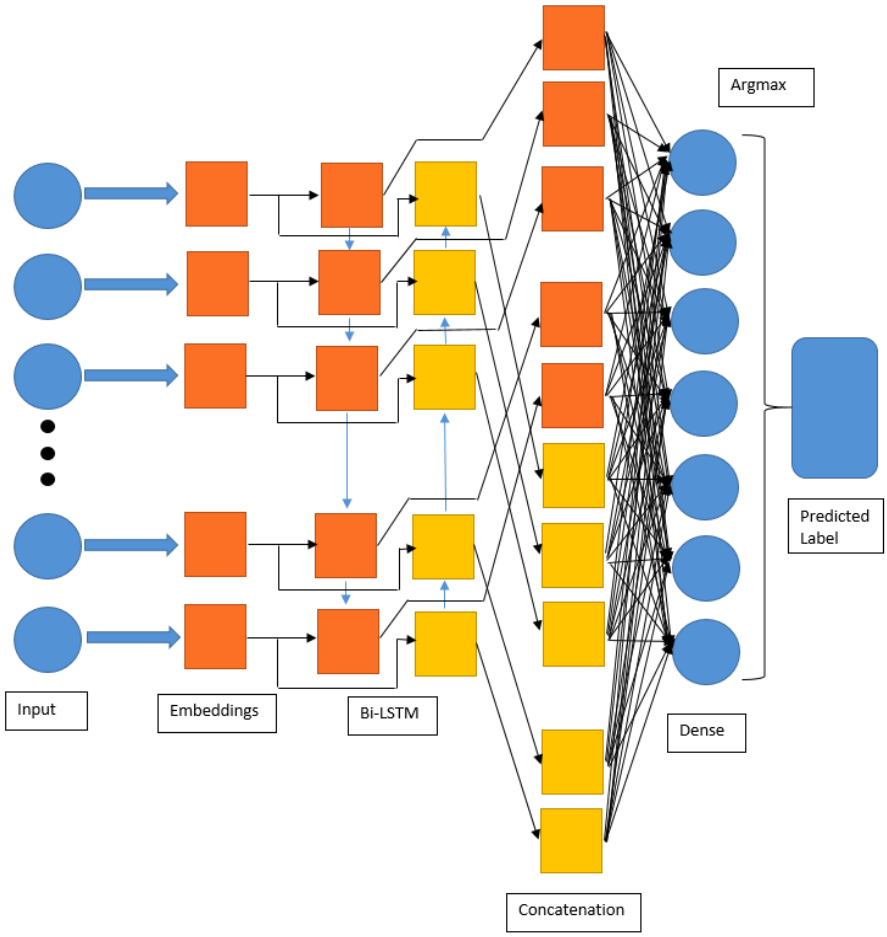


Figure 5: Neural network architecture used for the intent classification experiments.

The network was trained with a batch size of 32 for 8 epochs, using the Adam optimiser and the cross-entropy loss.

Figure 6 shows an overview of the performance of a number of different splits across four text transformations on ATIS-2 data. A topline on 100% original data is given in black at the top of the figure. A baseline in grey also uses only original data but utilising only 10% of the full training set. This is akin to using *only* personal data (at the same amount as the lowest actually mixed split), which of course does not constitute a realistic situation but serves solely as a comparison. Increasing the amount of original data beyond 20% does not result in further improvements.

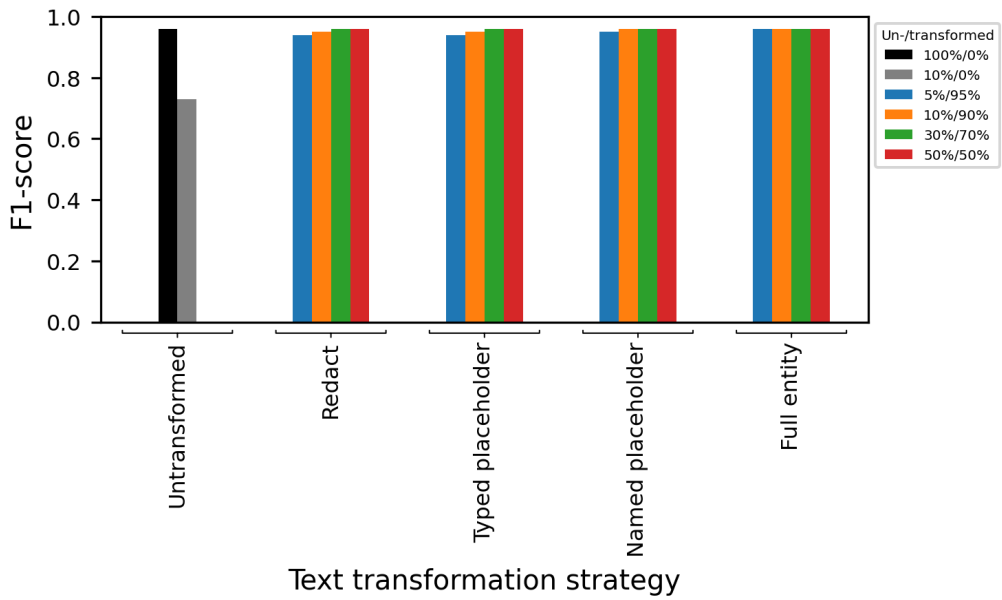


Figure 6: Intent classification results (F1-score) obtained using different original/transformed splits as training data, for a topline, a baseline, and four privacy-preserving text transformation strategies over the ATIS-2 dataset.

The Full-entity replacement condition performs almost equally well for all tested conditions, which can be explained by the fact that this text transformation results in the most natural transformed text, as private words are replaced by randomly sampled words of the same type. In fact, there is no significant difference between that condition and the topline, even when no personal data is used at all.

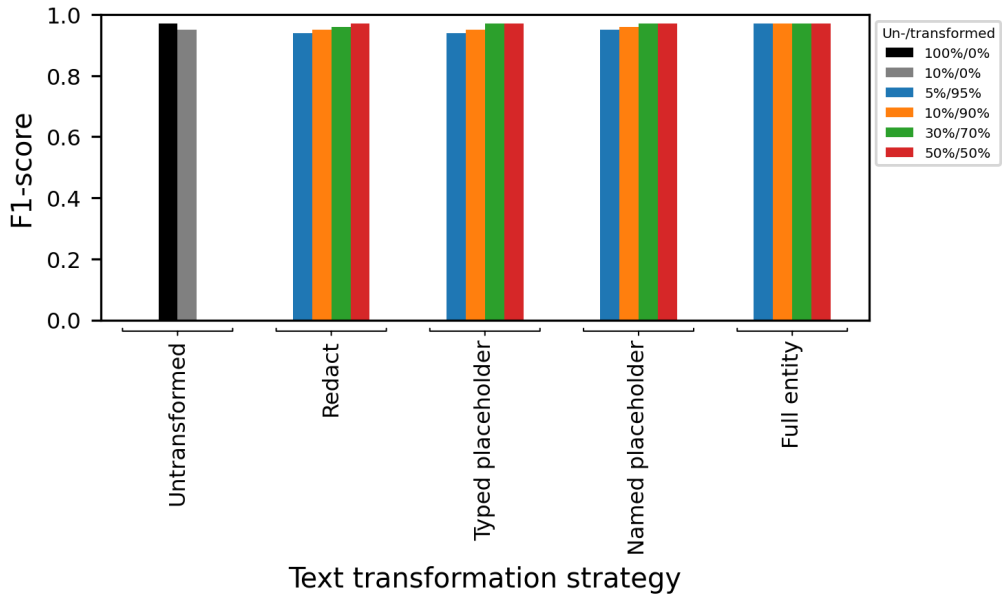


Figure 7: Intent classification results (F1-score) achieved using different original/transformed splits as training data, for a topline, a baseline, and four privacy-preserving text transformation strategies over the SNIPS dataset.

Results for experiments using the SNIPS dataset are included in Figure 7. In comparison to the ATIS-2 results, there are two notable differences. The first is that the performance drop from removing 90% of the data is less substantial, differing only by .020 points of F-score (from .972 to .952).

The second major difference is that the difference observed in the Redact, Typed placeholder, and Named placeholder conditions across the different splits is much more pronounced, with far higher variance than for ATIS-2. The cause of this phenomenon is still unclear, and further research will be required to investigate it.

However, there are also clear similarities between both sets of experiments. As for ATIS-2, we observe a significant performance boost when 10% of the training data are original in the Redact, Typed placeholder, and Named placeholder conditions, and additional original data does not continue this trend. More importantly, the Full-entity replacement condition performs, as before, robustly even when only transformed data are used for training.

5.4 Discussion and conclusion

The experiments for both SLU and DM show a similar trend: for the transformations Redact, Typed placeholder, and Named placeholder, including some original data in training improves the final result. However, to reach a performance as good as in the Word-by-word replacement or Full entity replacement conditions, the amount of additional data required might be too high to be realistically provided by a single user.

At the same time, these two sampling transformations do not even exhibit any relevant drop in performance, even when no personal data at all is provided during training. This observation is not unique to the experiments described above. As a matter of fact, we have shown that this effect holds for a number of additional tasks across multiple corpora in [Adelani *et al*., 2020].

There is no significant difference between using a small set of data only from a single user or from a mix of users. Where additional data can make a difference at all, it seems that it is more relevant for the additional data to be original.

The NER task on the WNUT2016 Twitter corpus has produced slightly worse results across all experimental conditions. This is likely due to the stylistic specificities of Twitter data. Unlike all other corpora, even the sampling conditions show a noticeable degradation when trained without or with only a small amount of personal data. However, since COMPRISE is targeting voice applications and not microblogging services, effects related to the typical grammar and vocabulary of the latter are unlikely to surface in the kind of applications COMPRISE seeks to support.

All of the experiments described here were done in “lab conditions” insofar as they combined transformed and original data and trained each model on the combined dataset. For the COMPRISE architecture, this setup is, of course, unrealistic as it would put a high burden on the user's device in terms of computational resources (e.g., GPU, memory and energy requirements). Alternatively, a common way to combine a large dataset (here: transformed data) with a small dataset that is more closely related to the target domain (here: personal data) is to train a base model on the former which is then fine-tuned on the latter. This would be a better match for personalisation in COMPRISE, where the base model could be trained on the COMPRISE Cloud Platform and only the fine-tuning would be done on the user's device or Personal Server. We therefore tested

this approach on the Verbmobil corpus, but found the results to be comparable to the simpler setup that was followed here.

In summary, our results give a clear recommendation for using one of the sampling-based text transformations, in which case additional personalisation is unnecessary for the SLU and DM tasks considered here.

6 Software libraries

6.1 Speech-to-Text

6.1.1 Acoustic model personalisation

The AM personalisation library is mainly identical to the one provided in Deliverable D3.2 except for the extraction of accent embeddings. In this version, the embeddings are obtained from all previous utterances of a given user. This version also provides a more refined code structure. Before starting, make sure that you have completed the following prerequisites:

- Modify the symbolic links *steps*, *utils*, and *sid* to point to the *egs/wsj/s5/steps*, *egs/wsj/s5/utils*, and *egs/sre08/v1/sid* folders, respectively, under your Kaldi installation.
- Also modify the content of *cmd.sh* and *path.sh* according to your needs and running environments. Make sure that the *KALDI_ROOT* and *SRILM* variables are pointing to the correct directory structure based on your Kaldi installation.

Step 1. Data Preparation

- In this first step, Kaldi-related files such as lexicon, spectral features, and LM are prepared. Check the documentation to learn more about data preparation in Kaldi: https://www.kaldi-asr.org/doc/data_prep.html. Examples can be found in the *AM_personalization/local/data_preparation.sh* script.

Step 2. Preparation for Embedding Training

- This step performs feature extraction, silence removal, mean-variance normalisation as well as speaker modification for accent embedding training. Some sample implementations can be found in the *AM_personalization/local/prepare_accent_data.sh* script.

Step 3. Train Embedding Model

- After the data preparation stages, we can start to train the embedding network. Note that this stage requires GPU usage. To enable multi-GPU settings, don't forget to configure the execution mode: *sudo nvidia-smi -c 3*. Then, run the training script *AM_personalization/local/train_xvector.sh* followed by the *AM_personalization/local/extract_xvector_embeddings.sh* script for extraction.

Step 3. Train and Decode Acoustic Model

- The extracted embeddings can now be used as auxiliary inputs for AM training. Run *AM_personalization/local/train_gmm_hmm.sh* first for traditional GMM/HMM training and alignment. Then, run *AM_personalization/local/train_chain_model.sh* for neural AM training, and *AM_personalization/local/decode_test_samples.sh* for

the corresponding decoding. Note that AM training also requires GPU usage, while decoding can be performed over the CPU.

6.1.2 Language model adaptation

The LM adaptation library implements the proposed personalisation strategies for LM adaptation. Its major components are designed to match built-in Kaldi tools and binaries. However, for the neural-based LM design, the open-source machine learning framework, PyTorch,[20] is utilised. Classical LM operations like N-gram training or linear interpolation are performed using the SRILM toolkit[21] which provides fast and lightweight calculations. The installation and usage instructions of this library are summarised below.

Prerequisites

- Ensure that you have a working Kaldi installation, if not refer to Kaldi's official repository: https://github.com/kaldi-asr/kaldi. Then, modify the *steps* and *utils* symbolic links inside the main directory to point to the *egs/wsj/s5/steps* and *egs/wsj/s5/utils* folders, respectively, under your Kaldi installation.
- In this library, the experimental analysis is performed over the AMI dialogue corpus. To use another dataset, pre-process your samples using the Kaldi preparation style.[22] A helper script for downloading the AMI speech samples and their corresponding annotations is given in *local/download_ami.sh*. It is also possible to download them directly from the AMI website.[23] Note that the annotations are in NITE XML toolkit (NXT) format[24] and require NXT version 1.4.4.
- To prepare the data for model training, the final dataset requires three splits:
    1. Training data with anonymised annotations. A domain/application specific corpus is preferred for more robust models.
    2. A small amount of adaptation data with similar characteristics to the test data. Like the test data, it should contain original annotations.
    3. Test data whose annotations should be original (non-anonymised) text.
- The PyTorch framework should be installed for neural LM manipulation. Note that scripts are organised to run individually for each component. Thus, if you are not going to use neural LMs, you do not need to install PyTorch.
- For entity recognition, the spaCy library is used with pre-trained pipelines which can be installed as Python packages. Please visit its official page for usage and installation: https://spacy.io/usage/models. For English, the *en_core_web_sm* module optimised for CPU can be used for NER: https://spacy.io/models/en#en_core_web_sm.

[20] https://pytorch.org
[21] http://www.speech.sri.com/projects/srilm
[22] https://kaldi-asr.org/doc/data_prep.html
[23] http://groups.inf.ed.ac.uk/ami/download
[24] http://groups.inf.ed.ac.uk/nxt

Setup

- For faster and efficient decoding, this library uses a "look-ahead" strategy to decode with runtime graph composition. This also makes it compatible with the VOSK library[25] which provides an offline speech recognition API for mobile deployment. The traditional approach for decoding is to create a huge graph from the LM, dictionary, and context-dependency graph and decode with a relatively simple decoder that just explores the best path. However, larger graphs may not fit the mobile platforms and limit the flexibility of the model. For this reason, our library rearranges the lexicon graph so that composition with the grammar can be done on-the-fly during decoding. This can be done by removing useless epsilon paths and pushing forward labels and weights along different paths. In order to take advantage of this cost-effective decoding, the OpenFST library must be compiled with the *--enable-lookahead-fsts* option. Download the latest version of OpenFST from http://www.openfst.org/twiki/bin/view/FST/FstDownload and install it: *./configure --prefix=/path/to/your/folder/openfst --enable-lookahead-fsts*.
- For main LM-related operations, the SRILM toolkit is required. Check the *tools/extras/install_srilm.sh* script for installation and other information located under the Kaldi folder.
- Create a *cmd.sh* file under the experiment folder based on the running queue for Kaldi. If you have GridEngine installed, you should also create the *queue.pl* file with arguments specifying where GridEngine resides.
- Prepare the *path.sh* file under the experiment folder which points explicitly to *KALDI_ROOT*, *SRILM* and OpenFST installations as well as other common paths.

Running the library

Step 1. Data Preparation

- This part includes extracting AMI data alongside their pre-annotated Named Entity annotations. Then, Kaldi processing files for each data split (train, test, adaptation) are created accordingly. To see the extracted meetings for each split, please refer to the *LM_adaptation/misc/meeting_ids* folder. Language and lexicon related folders are also obtained in this step. Finally, the training data is anonymised. The overall pipeline can be found in the *LM_adaptation/local/data_preparation.sh* script.

Step 2. Training TDNN-based AM

- In this step, MFCC and i-vector features are first extracted from the training, test, and adaptation splits in *LM_adaptation/local/feature_extraction.sh*. Then, GMM-HMM and TDNN-based AMs are trained in *LM_adaptation/local/gmm_hmm_training.sh* and *LM_adaptation/local/chain_training.sh*, respectively. Note that TDNN-based AM training requires GPU usage. To enable multi-GPU settings, don't forget to configure the execution mode: *sudo nvidia-smi -c 3*.

Step 3. Generating Individual LMs

- This step corresponds to Stage-2 in Figure 2. After getting anonymised training data, individual LM training schemes are evaluated to achieve generic LMs. The

[25] https://github.com/alphacep/vosk-api

different components of this stage are given in the script *LM_adaptation/local/train_generic_LMs.sh*.

Step 4. Adaptation and Interpolation of LMs

– After Stage-2, generic LMs are first adapted over the original adaptation data and then interpolated to achieve the final mixture LM corresponding to Stage-3 in Figure 2. For simplicity and effortless deployment, the SRILM toolkit is mainly employed over the ARPA-based LM representations. Inside the scrip *LM_adaptation/local/adapt_and_interpolate_LMs.sh*, major components of this step are presented.

Step 5. Decoding AM with Look-Ahead Composition

– To decode test samples with different LM settings, refer to the script *LM_adaptation/local/decode_with_lookahead.sh* which implements lookahead decoding. If you don't want to implement on-the-fly graph composition, *LM_adaptation/local/decode_standard.sh* performs standard Kaldi decoding with traditional HCLG graph generation.

6.2 Spoken Language Understanding and Dialogue Management

Our experiments on SLU and DM described in the previous sections have painted a mixed image. On the one hand, we were able to show that a modest amount of personal data can have a beneficial impact on the resulting model. However, it is questionable whether the contributions of a single user can be considered sufficient to have a relevant impact.

Despite extensive efforts to determine the decisive factors that determine the performance of the resulting models, our experiments eventually remained inconclusive. However, we have found out that using the Full-entity replacement strategy as the privacy-preserving text transformation already leads to model performances that are on par with those of models trained on fully original data.

The final personalised learning library therefore does not contain any special components for SLU and DM learning. Instead, we recommend the use of the Full-entity replacement strategy where possible.

7 Conclusion

This deliverable has presented, evaluated and analysed several model personalisation strategies for STT, SLU and DM. Two different aspects have been considered: one that relates to the AM for STT, and the other one that concerns text related processing. With respect to text related processing, it is important to recall that all text data which is collected on the COMPRISE Cloud Platform has been anonymised using privacy-preserving text transformations. This deliverable has investigated the impact of training models on such transformed data, compared to a more classical training carried out using original data (which is not feasible in the context of COMPRISE). Then, approaches for improving these models through adaptation using a limited data set of original text are presented, evaluated and discussed.

The AM is enhanced through the introduction of speaker embedding information along with the conventional spectral features. This approach was detailed and evaluated on

native and non-native English data in Deliverable D3.2 and in [Turan *et al*., 2020], and has been briefly recalled in this deliverable and evaluated on Latvian data. Several speaker embeddings can be used: i-vectors, initially proposed in the context of GMM-based speaker recognition; speaker x-vectors, that have also been proposed for speaker recognition but relies on a deep neural network; and accent x-vectors that we have proposed in Deliverable D3.2. On accented and non-native English data, accent x-vector embeddings lead to the best performance. On the Latvian data, the three types of embedding (i-vector, x-vector speaker, and x-vector accent) lead to similar performance, and the improvement with respect to the baseline (i.e., no embedding) is small. It is worth mentioning that the error rates observed on native, non-native and accented Latvian data are rather close to each other, even for the baseline model. Further investigation of these approaches on some other language datasets would be useful. However, there are very few accented and non-native speech corpora available.

The LM is the other component of STT systems where privacy is a concern. Several types of models can be used: N-gram word-based models, N-gram class-based models, and recurrent neural network based models that can handle longer histories that N-gram models. Experiments have shown that rescoring first-pass lattices using neural models in a second decoding pass yields the best results, especially when interpolated with 3-gram word-based and class-based models. However, from a practical point of view, the 3-gram class-based models that can directly be used for single-pass decoding are the best choice. Moreover, class-based models trained on anonymised data provide good speech recognition performance. Further improvement can be achieved by adapting the models with original data, but the improvement remains small with respect to the amount of adaptation data needed.

For SLU and DM we have conducted a series of exploratory experiments to sound out the potential for improvements when personal data is available in addition to the privacy-transformed data collected in the COMPRISE Cloud Platform. The results of these experiments are disappointing and encouraging at the same time: we found that, for most privacy-preserving text transformations, the amount of original data required to have a noticeable impact is rather large. Therefore, when these text transformations are used, achieving performance gains through personalisation may not always be realistic.

However, there is an exception. The two sampling conditions, Word-by-word replacement and especially Full entity replacement, while not benefitting from additional personal data, already perform very well out of the box, comparable almost to the (unrealistic) case where all training data is original.

In summary, the hopes of Task 3.3 to use personal data to improve models trained on transformed data have not been fulfilled at this point for SLU and DM. We therefore do not provide any particular software to that end. Rather, we recommend using one of the two sampling strategies for privacy-preserving text transformation.

Bibliography

Adel, Heike, Katrin Kirchhoff, Ngoc Thang Vu, Dominic Telaar and Tanja Schultz. "Comparing approaches to convert recurrent neural networks into backoff language models for efficient decoding." In *Proceedings of the 2014 Annual Conference of the International Speech Communication Association (Interspeech)*, Sep. 2014, Singapore, pp. 651-655.

Bellegarda, Jérôme. “Statistical language model adaptation: review and perspectives.” *Speech Communication*, vol. 42, n° 1, 2004, pp. 93-108.

Burger, Susanne, Karl Weilhammer, Florian Schiel, and Hans Tillmann. "Verbmobil data collection and annotation." *Springer Foundations of Speech-to-Speech Translation*, pp. 537-549, 2000.

Chinchor, Nancy, Erica Brown, Lisa Ferro, *et al*. "Named Entity Recognition task definition." *Mitre and SAIC*, 1999.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding." *In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Jun. 2019, Minneapolis, pp. 4171–4186.

Gillick, Laurence, and Stephen J. Cox. "Some statistical issues in the comparison of speech recognition algorithms." *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 1989, Glasgow, vol. 1, pp. 532-535.

Hemphill, Charles T., John J. Godfrey, and George R. Doddington," The ATIS spoken language systems pilot corpus." *In Proceedings of the DARPA Speech and Natural Language Workshop*, Jun. 1990, Hidden Valley, pp. 96-101.

Hess, K., K. J. Kohler, and H.-G. Tillmann, “Phondat-Verbmobil speech corpus.” *In Proceedings of the 4th European Conference on Speech Communication and Technology (EUROSPEECH)*, Sep. 1995, Madrid, pp. 863, 866.

Klakow, Dietrich. "Language model adaptation for tiny adaptation corpora." In *Proceedings of the 9th International Conference on Spoken Language Processing (ICSLP)*, Sep. 2006, Pittsburgh, pp. 2214-2217.

Kneser, Reinhard, Jochen Peters, and Dietrich Klakow. "Language model adaptation using dynamic marginals." In *Proceedings of the 5th European Conference on Speech Communication and Technology (Eurospeech)*, Sep. 1997, Rhodes, pp. 1971-1974.

Ko, Tom, Vijayaaditya Peddinti, Daniel Povey and Sanjeev Khudanpur. "Audio augmentation for speech recognition." In *Proceedings of the 2015 Annual Conference of the International Speech Communication Association (Interspeech)*, Sep. 2015, Dresden, pp. 3586-3589.

Ma, Min, Michael Nirschl, Fadi Biadsy and Shankar Kumar. "Approaches for neural-network language model adaptation." In *Proceedings of the 2017 Annual Conference of the International Speech Communication Association (Interspeech)*, Aug. 2017, Stockholm, pp. 259-263.

McGraw, Ian, Rohit Prabhavalkar, Raziel Alvarez, Montse Gonzalez Arenas, Kanishka Rao, David Rybach, Ouais Alsharif, Hasim Sak, Alexander Gruenstein, Françoise Beaufays, and Carolina Parada. “Personalized speech recognition on mobile devices.” In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, Shanghai, pp. 5955-5959.

Peddinti, Vijayaditya, Daniel Povey and Sanjeev Khudanpur. "A time delay neural network architecture for efficient modeling of long temporal contexts." In *Proceedings of the 2015 Annual Conference of the International Speech Communication Association (Interspeech),* Sep. 2015, Dresden, pp. 3214-3218.

Pestian, John P., Christopher Brew, Paweł Matykiewicz, D.J. Hovermale, Neil Johnson, K. Bretonnel Cohen, and Włodzisław Duch. “A shared task involving multi-label classification of clinical free text,” *in Proceedings of Biological, Translational, and Clinical Language Processing*, Jun. 2007, Prague, pp. 97–104.

Pinnis, Marcis, Ilze Auzina, and Karlis Goba. "Designing the Latvian speech recognition corpus." *In Proceedings of International Conference on Language Resources and Evaluation (LREC)*, May 2014, Reykjavik, pp. 1547-1553.

Povey, Daniel, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlıcek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. “The Kaldi speech recognition toolkit.” *In Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec. 2011, Waikokoa.

Saon, George, Hagen Soltau, David Nahamoo, and Michael Picheny. "Speaker adaptation of neural network acoustic models using i-vectors." In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Dec. 2013, Olomouc, pp. 55-59.

Singh, Mittul, Youssef Oualil, and Dietrich Klakow. "Approximated and domain-adapted LSTM language models for first-pass decoding in speech recognition." In *Proceedings of the 2017 Annual Conference of the International Speech Communication Association (Interspeech)*, Aug. 2017, Stockholm, pp. 2720-2724.

Strauss, Benjamin, Bethany Toma, Alan Ritter, Marie-Catherine de Marneffe, and Wei Xu, "Results of the WNUT16 Named Entity Recognition shared task," In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, Dec. 2016, Osaka, pp. 138–144.

Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney. “LSTM neural networks for language modeling.” In *Proceedings of the 2012 Annual Conference of the International Speech Communication Association (Interspeech)*, Sep. 2012, Portland, pp. 194-197.

Snyder, David, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. "X-vectors: Robust DNN embeddings for speaker recognition." In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018, Calgary, pp. 5329-5333.

Tang, Min, Dilek Hakkani-Tur, and Gokhan Tur, “Preserving privacy in spoken language databases,” In *Proceedings of the International Workshop on Privacy and Security Issues in Data Mining*, Sep. 2004.

Turan, M. A. Tuğtekin, Emmanuel Vincent, and Denis Jouvet. "Achieving multi-accent ASR via unsupervised acoustic model adaptation." In *Proceedings of the 2020 Annual Conference of the International Speech Communication Association (Interspeech)*, Oct. 2020, Shanghai, pp. 1286-1290.

Veselý, Karel, Arnab Ghoshal, Lukas Burget, and Daniel Povey. "Sequence-discriminative training of deep neural networks." In *Proceedings of the 2013 Annual Conference of the International Speech Communication Association (Interspeech)*, Sep. 2013, Lyon, pp. 2345-2349.

Wang, Dong, and Thomas Fang Zheng. "Transfer learning for speech and language processing." In *Proceedings of the IEEE Asia-Pacific Signal and Information Processing Annual Conference (APSIPA)*, Dec. 2015, Hong Kong, pp. 1225-1237.