Deliverable D4.1

Learning from user-provided data

Original title: Online learning for improving pronunciation modelling

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 287678.

Participant no.	Participant organisation name	Part. short name	Country
1 (Coordinator)	University of Edinburgh	UEDIN	UK
2	Aalto University	AALTO	Finland
3	University of Helsinki	UH	Finland
4	Universidad Politécnica de Madrid	UPM	Spain
5	Technical University of Cluj-Napoca	UTCN	Romania

Project reference number	FP7-287678
Proposal acronym	SIMPLE4ALL
Status and Version	Complete, proofread, ready for delivery: version 3
Deliverable title	Learning from user-provided data
Nature of the Deliverable	Report (R)
Dissemination Level	Public (PU)
This document is available from	`http://simple4all.org/publications/`
WP contributing to the deliverable	WP4
WP / Task responsible	WP4 / Task T4.1
Editor	Mikko Kurimo (AALTO)
Editor address	mikko.kurimo@aalto.fi
Author(s), in alphabetical order	Paavo Alku, Stig-Arne Grönroos, Reima Karhila, Mikko Kurimo, Juan Manuel Montero Martínez, Tuomo Raitio, Peter Smit, Adriana Stan, Oliver Watts
EC Project Officer	Pierre Paul Sondag

Abstract

The goal of the work package WP4 is to improve a TTS system by learning from user feedback. This is the first deliverable of WP4 and it describes the work done in 2013 for the first task T4.1. In the description of work the scope of this deliverable was rather limited and, finally, not very central in this task. Thus, we extended the scope of this deliverable to cover the entire task T4.1 and our initial progress towards the main goal of WP4. The main results within this scope are active learning, speech diarization, morphological text analysis, text polarity prediction, and improvements against noise.

Contents

1 Introduction

The goal of the project is to provide methods and open source tools to simplify and automate the training of new TTS voices and languages. Learning from user feedback will be a very important part of the training process, because the training data available for new voices and languages are seldom sufficient for high quality results.

This is the first deliverable (Deliverable 4.1) in WP4 and describes the work done in the first task (Task 4.1). The work was started at the WP4 kick-off session in the general project meeting in Helsinki, November 2012. Each partner was well represented in the meeting and each partner agreed to participate in the work. The outcome of the meeting was a list of various topics, which form the starting points of our exploration of learning from user feedback in this project.

The final goal of WP4 is very ambitious, because very few studies exist on TTS systems that learn from user feedback. The idea is promising, because user feedback is an as-yet untapped resource, however it is still unclear what the system can learn and whether the learning will produce any useful results. Thus, it was decided that the best way to conduct research on this topic would be to advance on multiple fronts at the same time and to explore via experimentation how to best use different forms of feedback. In other words, we made a plan to explore a wide variety of new ideas, many of which might fail, and from these initial experiments to re-plan and find the most effective directions to pursue. During the coming (and final) project year, these most promising directions will be continued and the resulting methods will be implemented in the final system.

During 2013, many discussions were held on what user feedback means for TTS, what kind of users there can be, and what can be learned from their feedback. Some new promising directions that extend the concept of user feedback were added to our initial ideas, including active learning and new ways to re-purpose existing results from subjective and objective evaluations. As a consequence of this strategy, when reporting the work done in WP4 and Task 4.1, the contents of this deliverable differ somewhat compared to the specifications in the Description of Work: we have extended the scope of this deliverable and it now represents a wider range or work aiming towards the general goals of WP4.

First, we will provide a revised specification of the scope of the work done in WP4 during the period M13 to M24, and then detail the topics covered in ongoing and planned work. In the original Description of Work, deliverable D4.1 was expected to describe only the use of feedback to improve the pronunciation of words. However, while considering this topic in more detail, this was found to be rather narrow, and unlikely to generalise beyond the particular words in question. Thus, that idea was quickly superceded and we have mainly worked directly towards the central challenge of improving the system by learning from user feedback.

The original description of T4.1 suggested two potential cases. The first one corresponds to the ‘Malaysian case’ which we have worked on and is described in Section 3.1. The second idea was to apply forced alignment on spoken feedback to determine a corrected phone transcription – this remains part of our plans and described in Section 4.1. The other cases where we tested learning from the user feedback to improve the TTS system are described in sections: 3.2 – 3.6.

The work done in this task is, naturally, not independent from the work in the other workpackages. As described in D1.5, the work on imperfect data from WP1 is somewhat overlapped with the active learning used in WP4 to select training samples (Section 3.2), improving the diarization (Section 3.3), and adaptation by noisy sentences given by the user (Section 4.1). In deliverable D2.2, the work on the semi-supervised front-end improvement is relevant to Section 3.4 of the current deliverable. The work done in WP3 on the flexible vocoder and the work here on defining user feedback and what can be learned from it to produce the proper voice (Section 3.6), have a natural connection to each other. Finally, the plans described here are a component of the project-level planning reported in D7.3.

Despite these extensions and our more developed specifications of the concepts of the user, feedback, and learning, the principal distinct feature of all work in WP4 remains: a TTS system is not trained and then becomes a static object, but instead there is an existing initial system that improves through further learning.

2 Scope of this task

In the Description of Work the goals of WP4 were ambitious and deliberately underspecified, since this is one of the most novel aspects of the project. The plans made at the end of year 1, the work actually done as part of T4.1 during the second year, and the plan for the work to be done in year 3 therefore all differ in detail from what was envisaged in the DoW, but are nonetheless at least as ambitious and our plans are of course now much more well-specified.

In this section we define the scope of what we mean by "learning from user feedback". When developing methods for using user feedback to improve a text-to-speech synthesis system, there are several things to consider. Firstly, the **scope of learning** must be identified, where either only some components or the complete system are improved by feedback. Two other aspects of learning that need to be specified are the **type of users** that give feedback, and the **mode of learning** that is done based on the feedback.

2.1 Scope of the learning

When examining methods for learning from user feedback, we must first think about which components will be updated as a result of learning. Using feedback to improve only a single component or to improve the entire integrated system, present different challenges.

2.1.1 User feedback on a single component

For most components in a TTS system, it is probably simplest and most convenient to start by using user feedback method to improve just that one current component, independently of the rest of the system. In many cases, a component performs a clear, measurable task that might benefit from direct user feedback in the form of annotations or direct, targetted evaluations.

Since most components in TTS systems are dependent on the previous components in the pipeline, e.g. a word classifier is dependent on a word being correctly tokenized, improvements in one component will still propagate through and should in general improve the whole system. Because even small errors in earlier components can result in significant errors when propagated through subsequent components where the error is compounded, even apparently small local improvements have the potential to make substantial improvements to overall performance.

However, optimizing a single component at a time also has disadvantages. The first disadvantage is the need for a great amount of user resources to optimize a complete system. As each component requires its own type of feedback, which is not shared between components, any other (possibly implicit) information which might have been useful to another component will be wasted. In the scenario of a single developer creating a synthesis system for a particular language, the amount of feedback needed to improve all components might be very substantial. One solution to that might be to crowd-source the feedback.

The second disadvantage is that some components can only be improved if there is access to expert knowledge. For example, a prosody labelling component can certainly benefit from correct annotations for some training sentences, but the task of annotating might be out of reach for non-experts. In this case, soliciting a less direct (i.e., more implicit) form of user feedback might be more appropriate, such as asking non-expert native speakers which of two sentences, with different prosodic patterns, sounds the best.

2.1.2 Integrated system feedback

In an ideal (from the users' point of view), albeit possibly unrealistic, situation is that the system learns from users' feedback which they provide about the overall end-to-end performance of the system. Because the user will not generally understand the functioning of individual components, this type of feedback is implicit, may take various forms, and will not be directed at a specific component. So, it may be hard to identify what useful information is contained in the feedback, and how to optimize individual components from it. This process is hard but we think not impossible.

A practical operating point in between the per-component and end-to-end situations is to engage in some form of interaction with the user to allow the system to narrow down the component(s) causing the error, and act appropriately.

Besides the problems in identifying the component that needs improvement, there is another practical aspect to using integrated system feedback. Because modifying earlier components changes the input of later components in the pipeline, it may be necessary to retrain large parts of the system, which could be resource-intensive. For example, modifying any part of the text processor will change the contextual labels of the acoustic models, and necessitate acoustic model re-training.

2.2 Types of user

2.2.1 Expert vs naive user

The user's expertise in linguistics and experience in using speech tools are invaluable for getting explicit feedback about specific system components. An expert user could even give intermediate feedback without hearing the end result and make complicated decisions like defining phone sets, editing pronunciation dictionaries and tuning vocoder parameters by hand.

However, a naive user can also be useful. In line with the goal of the project, to make it simple for everyone to build a TTS system, we will try to make it possible also for a naive user to give useful feedback. Often this will have to be implicit feedback based on overall system performance, tying in with Section 2.1 and Section 2.3.2.

2.2.2 Real vs simulated users

As real users' time is expensive and limited in quantity, this could limit the speed at which we are able to develop our methods. Therefore, we will investigate ways to simulate users, for the purposes of our own development cycle. Likewise, because subjective measures of synthesised sentences can only be obtained via listening tests, objective measures would be an attractive form of "simulated user"; as envisioned in the Description of Work, we will consider the use of such measures, possibly tuned to maximise correlation with the subjective scores.

2.3 Mode of learning

2.3.1 Offline vs online

By *offline learning* we mean that users do not expect an immediate result and are willing to provide substantial feedback (e.g., annotation) that will be then used in the learning process to improve the system. In contrast, *online learning* implies that the user expects the system to adapt quickly to her or his feedback and for the output synthetic speech to change in response to the feedback.

2.3.2 Explicit vs implicit knowledge

As already described in Section 2.2.1, users having different levels of expertise and knowledge of the TTS training process can provide different kinds of feedback to the system. Explicit feedback includes anything that is currently provided by linguistically-expert experts in the construction of current state-of-the art TTS systems. For example, phone lists, pronunciation and prosodic annotations and other rules needed by the system. However, non-experts can also provide very specific knowledge about their native language. For example, finding word boundaries, translating written numbers into sequences of word, and detecting foreign words are tasks that can be performed by any speaker of a language.

Providing implicit feedback is a task that can be done by even the most naive user. A user who needs a system to read audiobooks could make the decision to use audiobooks as training data, implicitly making a key decision. Also, most listeners will be able to detect incorrectly spoken utterances, and in that way provide annotations of both good and bad output, from which the system can learn.

2.3.3 Supervised vs semi-supervised vs unsupervised learning

A fundamental property of a machine learning algorithm is the level of annotation needed on its input data. A completely unsupervised system uses data without any annotation to perform its tasks. Supervised systems only take well-labeled data in order to train models that can predict the label of future unlabelled instances.

In semi-supervised learning, both labeled and unlabeled data are used. The most basic example is a clustering algorithm that needs a label for one data point in each cluster in order to label the clusters it created. One would predict that the more supervised data is provided, the better the semi-supervised system performs.

An example of a system developed in SIMPLE4ALL that can benefit from semi-supervised learning is Morfessor FlatCat. The user can provide correct morphological segmentations of some words in order to guide the automatic word analysis towards more linguistically-correct decompositions.

2.3.4 Active learning vs passive learning

In *active learning*, an intelligent selection is made of which data needs to be annotated, in order to minimize the amount of annotation needed. *Passive learning* corresponds the traditional annotation process, where neither the system nor the user will select the training data that is being annotated and used for the learning process: simply all data will be annotated.

3 Ongoing work

3.1 Interactive construction of letter-to-sound rules for Malay using active learning

The active learning tool developed for the work described is made as general as possible and is described further in Section 3.2. The extraction of features from possible training examples in the example-pool is task-specific, but the code by which the user interacts with the system is as general as possible. A preliminary experiment in applying it to a new task was carried out by a SIMPLE4ALL intern visiting UEDIN , Lau Chee Yong, from (and funded by) Universiti Teknologi Malaysia.

This experiment was designed to improve letter-based synthesis of Malay by exploiting some limited interaction with a native speaker, but without incurring the cost of compiling a full lexicon. The relation between graphemes and phonemes in Malay is generally straightforward, but there are a limited number of graphemes with ambiguous pronunciations. One example is the letter <e>, which can correspond either to /e/ or to schwa. The same approach to active learning was used as the one described in Section 3.2: several hundred examples of /e/ in different words were collected, and as predictor features neighbouring letters in a 7-letter window were collected. 150 randomly picked examples were initially hand-labelled (using the arbitrary symbols <e> and <é> to disambiguate the pronunciation). A further 200 examples were selected (one at a time) by active learning and presented to the user for labelling. The classifier resulting from the final iteration of active learning was then used to predict the pronunciation of new examples, including in the transcript of speech data for acoustic model training. Systems were built using the naive approach detailed in [1], both with and without the newly acquired classifier for <e>.

The learned classifier achieved an accuracy of 89.83% on held-out, semantically unpredictable sentences. Use of the classifier reduced the WER of 28 listeners' transcriptions of synthetic speech from 41.67% to 15.69% when acoustic models were trained with studio-recorded speech, and from 54.65% to 40.42% where automatically-aligned 'found' data was used. Further details of the Malay voices built are given in Deliverable 2.2.

As a form of learning from user feedback, this work can be categorized as a *naive* user giving *explicit* feedback for a *single component* of the system in an *active* learning fashion which is *supervised* and *offline*.

3.2 Lightly-supervised data selection

In [1] we outline an approach where active learning is used to get user feedback regarding data to be used to train TTS systems; details are repeated here for convenience. That paper describes systems built from training data which has been automatically harvested from audiobooks. The reason that audiobook data was used in the work described there is that it presents a source of 'found' data from which TTS training databases can be harvested without the need to acquire a purpose-made speech database. Although modelling the rich variety of speaking styles found in such data is a longer-term goal, we there followed an approach similar to the one presented in [2], which aims to select a neutral subset of a database containing *diverse* speech. In [2], 9 utterance-level acoustic features are used along with several textual cues to exclude diverse speech from the training set. Thresholds over these features are set manually by the system builder to exclude non-neutral utterances.

For the current work we perform utterance selection using an active learning approach, with uncertainty sampling [3]. Rather than being required to tune thresholds manually, the system builder is presented with example utterances and asked to indicate whether or not they are spoken in a neutral style. The interface therefore insulates the user from the details of the features used, and lets the user focus on what should be key: their intuitive response to hearing speech samples. The procedure we used is as follows:

1) **Feature extraction** First, frame-level features (F_0, energy and spectral tilt – approximated by 1st mel cepstral coefficient) are obtained, from which utterance-level features are computed. The fact that no thresholds need to be manually tuned means that we can afford to use a great many more features than the 9 employed in [2]. Our feature set is based on the one described in [4]: we compute mean, standard deviation, range, slope, minimum and maximum (6-level factor) for $F0$, spectral tilt, and energy (3-level factor) in the following sub-segments of each utterance: entire utterance, 1st and 2nd halfs, all 4 quarters, first and last 100ms, first and last 200ms (11-level factor), giving a total of 198 features.

2) **Initial labelling** The user is presented with the audio of s randomly-selected *seed utterances* from the whole corpus (via a text-based user interface) and asked to label them *keep* or *discard* – utterances are labelled with the user's decision.

3) **Classifier training** A classifier is trained on the labelled examples. Our choice of classifier is a bagged ensemble of decision trees [5] because it can be trained quickly (allowing online active learning in real time), is robust against noisy features and able to accept unnormalised input variables, and mixtures of discrete and continuous input variables (allowing a great many different acoustic features to be used, and different types of features), allows the space of utterances to be partitioned recursively (enabling complex interactions between features to be detected), and provides robust estimates of class probabilities (important for step 4).

4) **Uncertainty sampling** The set of u uncertain examples (utterances about which the classifier is most uncertain – in the present case, the utterances which have closest to 0.5 *keep* probability). The utterances in this set are presented to the user for labelling.

5) Steps 3 and 4 are repeated as many times as time allows.

6) The set of utterances either labelled *keep* by the user are kept for training, as well as enough of the utterances to which the trained classifier gives the highest *keep* probability to, to make up the desired quantity of training data.

For the work presented here, s was set to 15 and u was set to 1. That is, the user was asked to provide 15 labels at the outset, and presented with a single uncertain example at each iteration. The stopping criterion we used in this work was to limit the number of iterations to 15 – in the present, utterance selection time was limited to approximately 20 minutes per language, and 15 was found to be a reasonable number of iterations in that time. Informal comparison suggested the approach outlined is beneficial for this task, but in ongoing work we are testing this rigorously and comparing uncertainty sampling with random sampling, as well as applying our active learning tool to other TTS tasks.

This work can be categorized as a *naive* user giving *implicit* feedback about the *full system* leading to *active* learning which is *semi-supervised* and *online*.

3.3 Diarization

When dealing with imperfect data, a recorded file containing speech from several speakers must be processed automatically (for example, using a segmentation tool such a speaker diarizer, a style diarizer, a music detector or a text-to-speech aligner) before training a speech synthesiser based on that material (for example, cloning the voices of the speakers, or the available speaking styles).

It is common to run a battery of experiments to get the best possible output: the output with the most perfect data. Each experiment has a different set of configuration values, because, in some situations, there is a configuration which performs significantly better than the other ones. Sometimes, the best values for the recorded file currently being processed are unknown, so it is necessary to determine which of the available configurations is the best one.

The standard approach evaluates the performance of every configuration on a manually-labelled golden reference. However, if the full golden reference is created, the automatic tools are no longer needed, because the reference is already available. An alternative approach (semi-supervised or semi-evaluated) could be based on a partial evaluation of the output of the process: the best a-priori output (the best configuration according to experiments carried out on a previous independent set of recordings) is used for an iterative validation process, which builds an incomplete reference to select the best configuration. At each iteration, the outputs of all the configurations are ranked and the best one is selected for the next iteration, until a certain amount of speakers' material has been validated and the estimated evaluation on a partial reference is deemed sufficiently reliable.

Our user-feedback tool has been designed for improving the speaker diarizer and it tries to carry out that iterative process through a series of Yes/No questions, which will help the developer determine the best configuration to be used (that is, the best available output). It is designed to be used by people with no special expertise in speech science (e.g., the developer of a speech-enabled application), so the questions have to be simple: the user must simply decide whether the speakers of two audio segments are the same or not. The audio segments are selected from the output of the diarizer: each segment has a beginning time, an end time and a pseudo-speaker label; it is a pseudo-speaker because, being the output of an automatic process, the labels could be incorrectly assigned: the same pseudo-speaker label can be assigned to several speakers and vice versa.

The program will choose the best solution based on two overall phases: the pseudo-speaker confirmation phase (which of the automatically-detected pseudo-speakers are different from one another?), and the pseudo-speaker purity (have all the segments assigned to this pseudo-speaker been correctly assigned?). When the evaluation process starts, a default solution (the one that usually gives the best results) will be chosen, and some questions will be presented to the user in order to check that the different speakers present in that solution are, in fact, different. After that, the user will have to answer some questions about whether two segments of the same pseudo-speaker in the solution have been really spoken by the same person. Throughout the process, the answers given by the user are checked against all the solutions, so we can see which one suits best these answers. After a certain number of questions and answers, if there is a clear best solution, we may end the process and present it as the best diarization for this audio file.

3.3.1 Description of the program

Initially, the GUI program loads some files: the diarization outputs to be evaluated and configuration file of the GUI program. Then it activates the default preferred output; if there is no default, it activates the first available one.

In order to try to optimize the iterative process for a voice cloning task, the program filters out the shortest pseudo-speakers: diarized speakers with less than X seconds or without segments shorter than Y seconds, X and Y being defined in the configuration file. In the first iterative phase, the programme checks how many of detected pseudo-speakers are really different from one another: it chooses one representative segment from each pseudo speaker (the representative segment is the first one in the diarization output, but no special order is imposed to the output file: the first segment could be the longest one or the most prototypical one for that speaker, if the output of the diarizer has been properly reordered) The programme generates all the possible pseudo-speakers pairs to

be listened to by the user. Then, it iteratively shows a window with a pair of files to be listened to and a Yes/No question: "Are these two audio segments from the same speaker?". If the answer is 'No', the program iterates to the next pair of segments of pseudo-speakers. However, if the answer is 'Yes', the programs iterates through other segments assigned to the same pseudo-speaker to check if both pseudo-speakers are the same or not. An odd number of segments (3, currently) are checked, and the programme uses a majority voting scheme to decide whether to merge both pseudo-speakers or not (any case, the incorrectly assigned segments are properly labelled and be used to fix errors in the finally selected diarization output).

After this first phase, the programme has at least one segment from each pseudo-speaker which has been validated as different from the other pseudo-speakers' segments. This second phase iterates through the segments assigned to each validated pseudo-speaker to check the precision of that assignment. Once again the program shows the Yes/No question window with two audio files to be listened to, but this time, according to the diarizer output, both files are supposed to be from the same speaker. In this second phase, one of the files has been previously validated (in the first phase or in the second one) but the other segment is to be validated.

If the user validates that both speakers are the same (by answering 'Yes'), the program proposes another pair of segments of another pseudo-speaker: It tries to sweep many segments of all the validated pseudo- speakers. At any time, the user can stop the feedback process by closing the window and stopping the program execution.

If the user does not validate two segments as segments from the same speaker in this second phase, there is a new sub-phase to check if the non-validated segment could be assigned to any other of the validated speakers. If the segment can be attributed to another speaker, it is assigned to this speaker (partially fixing the diarization output); if the segment can be assigned to any of the speakers, it is marked as a failure.

After each iteration of this second phase, the program updates the reference solution that it is partially building, and checks which of the diarization outputs best matches this partial reference. If the previously-selected output is not the best one, the programme automatically switches from validating one solution (or diarization output) to validating another one.

To switch to a new solution, a new partial reference has to be created, taking information from the previous partial reference, of course, but also from the new proposed diarization solution: the names of pseudo-speaker can be quite different from one solution to another, and the names in the partial reference are based on the names on the previous solution, not on the new best one: the names have to be mapped from one solution to another and the segments have to be re-assigned. The script for computing the diarization error is able to map the pseudo-speakers automatically, and the validated segments are reassigned when the amount of overlapping between them exceeds a certain configurable threshold (for example, 70%) and the pseudo-speakers can be mapped.

If a segment was marked as a failure, but the mapped pseudo-speaker in the new solution is not the same as in the previous solution, we have detected a new possible pseudo-speaker to be checked in a new (but simplified) phase one: the new pseudo-speaker will be compared to the validated ones to verify it is really new. However, if a failed segment was assigned to the same mapped pseudo-speaker in the new solution, the segment has to be labelled as a failure again. All the other segments are marked as segments to be validated

In addition to build a partial golden reference file and selecting the best possible diarization taking into account that reference, the programme logs the actions and decisions of the user: the answer to the question, which segment the user has listened to first, how long was the user listening to each segment, the number of times the user was listening to each segment, in which phase the programme is now, from which solution these segments come from, etcetera. Finally, some statistics are computed.

In terms of learning from user feedback, this work can be categorized as a *naive* user giving *explicit* feedback about a *single component* via *active* learning which is *semi-supervised* and *offline*.

3.3.2 Evaluation

For this experiment we have run the style diarization tool on audio file corresponding to one chapter of the audiobook "A Tramp Abroad" from the Blizzard Challenge 2012. We have run the tool for all the possible values of weights from 0 to 1.0 in a 0.05 step. The objective of the user is to try to improve the diarization results, but

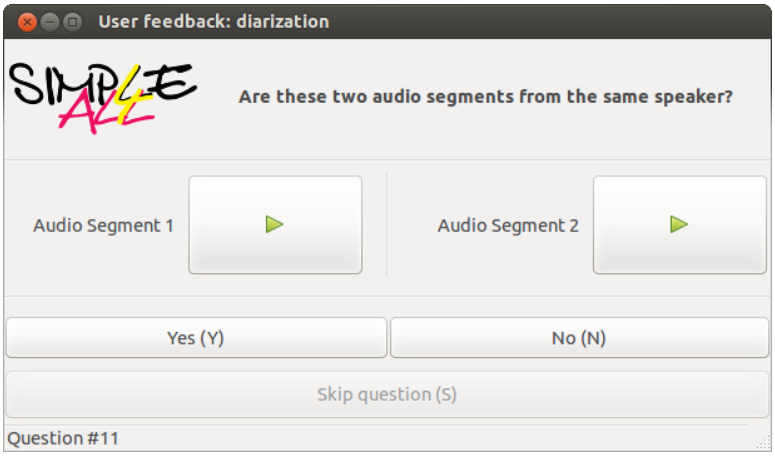

Figure 3.3a: The main window of the Graphical User Interface

focusing on speakers segments which are at least 3 seconds long. In this mono-speaker style diarization binary experiment, the styles the user is supposed to listened to are the narrator style and the expressive style of the characters the actor is imitating. The starting point of the feedback process is a mid-range solution with 3 detected speakers.

In the first phase of the feedback process the user listened to 6 pairs of audio diarized segments from chapter 2 (which is 1104 seconds long in total). The average time spent in this phase was 4.7 seconds per pair of files. As a result of this phase the 3 speakers were merged into just to 2. This led into 6% estimated error for the initial solution, therefore the user-feedback tool switched into a better solution according to the partial feedback provided by the user. The weight of the new solution was 0.65, very close to the optimum value 0.7. The estimated error was temporarily equal to 0%.

In the second phase, the system asked the user for feedback on different randomly-selected audio segments which were assigned to the same speaker by the 0.65 diarization process. After listening to a total of 112 seconds of speech (about 10% of the file), the 0.65 solution kept on being the best one, although the error estimated for other solutions is less than 5% greater. After listening to 219 seconds of speech, the estimated best solution was then 0.75 (the estimated error is 2.9% although there are 4 other solutions with an estimated error lower than 5%, including the optimal 0.7). After listening to 330 seconds (about 30% of the audio file), the best solution is now 0.9 (speaker error rate=4.4%) , and the best solution is ranked third (5.3%).

The listening time per segment was roughly similar for the narrator segments and for the expressive segments (around 4 seconds per segment). 72% of the times the user listened to narrator segments and 28% of the time the user listened to expressive segments.

3.4 Morfessor

3.4.1 Introduction

The Morfessor algorithm family is a group methods for segmentation of words into morphs using a probabilistic model. A morph is theoretically the smallest part of language carrying meaning. Although similar to syllables in appearance (a segmentation of a word into one or more sub strings), morphs decompose a word according to its information units instead of phonological ones.

Morfessor was originally developed at AALTO [6, 7]. During project year 2 it has been improved [8] and integrated into the new SIMPLE4ALL front-end. Also new methods for using annotated data have been implemented, which provides opportunities for improving existing unsupervised models with annotated data.

The cost function of Morfessor Baseline is derived using a MAP estimate:

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \boldsymbol{D}_W) = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}) p(\boldsymbol{D}_W \mid \boldsymbol{\theta}) \tag{1}$$

In order to include supervised annotations – such as user feedback – into the model, the likelihood of an annotated model $\boldsymbol{D}_{W \to A}$ is added with control parameters α and β to balance the cost of the unsupervised and supervised data likelihoods:

$$L(\boldsymbol{\theta}, \boldsymbol{D}_W) = -\log p(\boldsymbol{\theta}) - \alpha \log p(\boldsymbol{D}_W \mid \boldsymbol{\theta}) - \beta \log p(\boldsymbol{D}_{W \to A} \mid \boldsymbol{\theta}) \tag{2}$$

The annotated data can both be provided before training, or incrementally for an already-trained model. When data is provided incrementally, only a short re-training is needed to update the model. As annotated data can be provided at any time, Morfessor can be used for both offline and online learning from feedback. The type of feedback needed however, can only be provided by an expert in the target language, requiring explicit knowledge of its morphotactic structure.

The work on Morfessor 2.0 has been completed, but further study into additional active learning strategies is being considered.

This work can be categorized as an *expert* user giving *explicit* feedback about a *single component* of the system leading to *passive* learning which is *semi-supervised* and *online*.

3.4.2 FlatCat extension

A new method in the Morfessor family called Morfessor FlatCat has been developed in SIMPLE4ALL . FlatCat combines the morphotactic constraints from the Morfessor Cat-ML [9] and Cat-MAP [7] models, with the corpus likelihood weighting and semi-supervised learning introduced in Morfessor Baseline. FlatCat shares with Morfessor Baseline the use of MAP estimation for deriving the cost function. The way that the cost function is extended for likelihood weighted semi-supervised learning is identical to Morfessor Baseline: see Equation 2.

As FlatCat training is computationally more expensive than the training of Morfessor Baseline, an online training approach has been adopted for implementing an active learning strategy. In this online learning mode the user can input words either unlabeled as complete words or alternatively as a specified segmentation to be included in the labeled corpus. This allows the user to immediately correct detected errors in the segmentation during the training.

Currently, active training of FlatCat uses single component feedback, but it could be integrated as a part of full system feedback. The current feedback is in the form of explicit knowledge of the correct segmentation, which must be provided by an expert user. The required level of expertise could, in future work, be reduced by presenting *n*-best lists of segmentations that the user can select from. This would replace the need to specify the correct segmentation unaided.

3.4.3 Evaluation

To evaluate the incremental use of annotation data for Morfessor Baseline, three Finnish models with different amounts of data were created and trained until convergence. Every iteration, 20 annotated words were added and the F-measure on an independent test set is reported. The results can be seen in Figure 3.4a, which shows that the annotations give a significant boost in accuracy. In particular, the first 100 annotations lead to a boost in the F-measure for all three models.

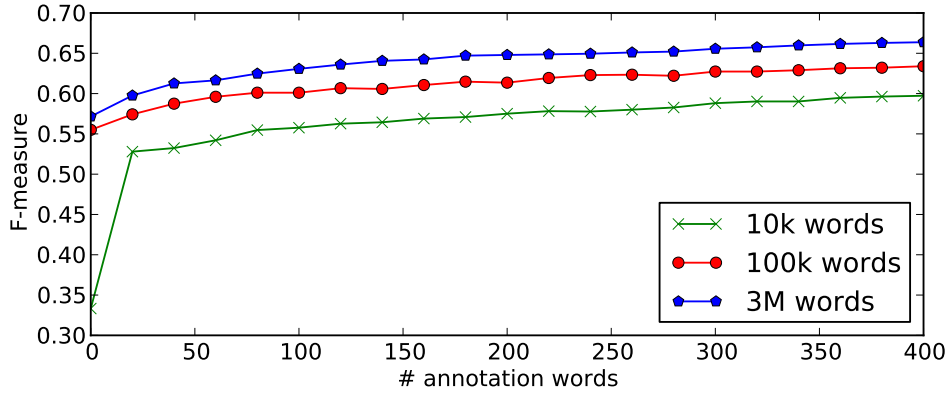

Figure 3.4a: Improvements in F-measure for Morfessor Baseline by incrementally increasing the amount of annotated data. The three different models differ in the amount of unannotated words available for initial training.

3.5 Polarity Prediction

Detecting the polarity of a text is, in most cases, a highly subjective task. Word meanings within full paragraphs are not always perceived in a similar manner by all individuals. In text-to-speech synthesis systems, for many applications, the text to be synthesised is limited to one phrase. This makes the task of determining and then inducing prosodic polarity for the output speech an rather daunting task.

There are, however, a number of recent studies which aim at using sentence-like chunks in a polarity prediction module. The work of [10] presents the entire workflow of a such module. One of its disadvantages in the context of SIMPLE4ALL is the fact that most of the pre-processing steps involved in the workflow are supervised, or make use of large amounts of annotated data. In [11] we tried to remove as many of these supervised steps as possible, and the results showed only a minor decrease in the algorithm's performance. But still there are some steps that are not fully unsupervised, the most important one being the use of the WordNet Affect database for the dimensional approach. Given the parallel work done in WP2, we have started to look at Vector Space Models (VSM) as a method for building an affective feature database. The idea is also based on the work of [12], where sentiment classification is achieved through the use of a single seed word. The additional development in our work will be the use of user feedback obtained via active learning.

Although we have not yet evaluated our methods rigorously, we will briefly summarize here our current solution: starting from a number of seed words considered to have a strong inherent positive or negative polarity, we ask the user to translate them into their own language. Then using a large text database and VSMs, we extract related words (that is, related in a distributional sense, but not necessarily in a linguistic sense) and ask the user to rate them as having a positive or negative polarity. This database can then be incrementally built not by just one, but by several users and can become a unsupervised replacement for WordNet Affect in that respective language. Then, the use of the dimensional approach presented in [11] will become feasible.

In terms of learning from user feedback, this work can be categorized as a *naive* user giving *implicit* feedback for a *single component* via *active* learning which is *semi-supervised* and *offline*.

3.5.1 Evaluation

The evaluation of the above described method is ongoing, and we detail here the framework in which it is included. The target language is Spanish, for which we also have a high-quality supervised sentence polarity detection algorithm, provided by UPM . From the English WordNet Affect we first selected the top 10 words best describing the negative and positive polarities. A native speaker then translated these 20 words, considered as *seeds* into

Spanish. In parallel, the Spanish Wikipedia dump[1] was used to create a VSM with 250 left and right context word co-occurrence matrix. Words which have fewer than 50 occurrences were discarded. Using the cosine similarity between each seed and all the tokens from the VSM, we then select the top 5 most similar words. These will then be rated by the users in a crowd sourcing manner. Newly added words will be used to select others, in an iterative manner.

After obtaining a sufficiently large number of positive and negative polarity words (we estimate that 1000 for each polarity should be enough for initial results), we plan on evaluating the lightly supervised Spanish WordNet Affect against the supervised method.

3.6 Adapting to different speaking styles

3.6.1 Introduction

Humans are capable of regulating the intensity of their speech extensively. When talking in environments with high levels of interfering noise, talkers typically raise their vocal effort in order to create sounds of larger intensity thereby improving the intelligibility of their vocal message. This phenomenon, the Lombard effect, is observed as changes in many acoustical features of speech such as an increase in fundamental frequency (F0) and vowel duration as well as a decrease in speaking rate and spectral tilt [13, 14, 15, 16]. In contrast when speaking in silence or in conditions with little noise, humans typically use low vocal effort in which a breathy type of phonation is used to create soft, low intensity speech. This variation in the use of vocal effort, from breathy to normal and then to Lombard speech, is called the vocal effort continuum in this report.

Since the vocal effort continuum is a natural part of human speech communication, it would be highly desirable to be able to utilize it also in TTS. Unfortunately, this has turned out to be difficult particularly in concatenative systems in which large amount of speech data is required to cover sufficient units along the continuum. In HMM-based synthesis, however, the construction of such a continuum is, in principle, feasible thanks to the recent progress in adaptation techniques [17]. In particular, combining these adaptation techniques with the physiologically oriented vocoder developed in the project was considered a research topic worth elaborating. Therefore, a study was conducted as a joint effort of WP3, WP2 and WP4 to investigate the following research questions: By using the physiologically oriented vocoder developed in WP3 together with an adaptation technique proposed in [17], would it be possible to synthesize high-quality speech along the vocal effort continuum and, particularly, what would be the perceptual user feedback to this kind of TTS when compared to natural speech. Vocal effort continuum was tested by simulating a speech-in-noise scenario, a situation where the phenomenon typically happens in real-life. It is worth noting that the utilization of user feedback in this context is limited to feedback which is obtained via subjective listening tests (e.g. possible intelligibility improvement due to successful synthesis of Lombard speech in noisy conditions). This feedback, however, can later be taken advantage in TTS applications where the synthesizer is adapted in terms of its speaking style on the vocal effort continuum based on automatically estimating the signal-to-noise (SNR) ratio of a listening environment (see, e.g., [18]).

3.6.2 Building synthetic voices

Synthetic speech was created to correspond to three points on the vocal effort continuum: breathy, normal, and Lombard speech. Corresponding synthetic voices were created by training and adapting the statistical parametric speech synthesis system GlottHMM. Speech material for the three speaking styles was recorded from two subjects (one female, one male). For the normal speaking style, approximately 2h of speech was recorded from both subjects. (For more details on recordings, see [19]). In creating the breathy and Lombard voices, the normal voice models were adapted with constrained structural a maximum a posteriori linear regression combined with maximum a posteriori (CSMAPLR + MAP) adaptation technique [17]. Adaptation was applied to all streams using

[1] `http://dumps.wikimedia.org/eswiki/`

pruned state-tying decision trees for regression classes. The amount of adaptation data was the same for both speakers, 300 sentences in Lombard style and 200 sentences in breathy style.

3.6.3 Evaluation

Various perceived characteristics of natural and synthetic were evaluated by subjective listening tests where listeners were played with speech at three points on the vocal effort continuum in different noise conditions. The intelligibility of speech was the main measure, but the importance of speech quality in silence and in noise was also addressed. Finally, the question of contextual appropriateness of speech with respect to the noise environment was considered. Additionally, the overall pleasantness of the voices was queried in order to find out whether the subjects would prefer to listen to low-effort speech, especially in silent conditions.

In order to simulate natural human speech communication, a realistic noise environment shown in 3.6a based on a multi-loudspeaker set-up was created [20, 21].Three different noise scenarios were selected: silence, moderate street noise, and extreme street noise. The averaged A-weighted SPLs were 63 dB and 70 dB for the moderate and extreme street noise cases, respectively. These noise levels were selected based on pre-listening of the speech samples with different noise levels. In this procedure, levels which resulted in the most prominent differences in intelligibility between the test cases were chosen. Thus, the average SNRs were -1 dB and -8 dB for moderate and extreme.

Figure 3.6a: A photograph of the listening test setup. The two rear loudspeakers at angles $\pm 160^{\circ}$ are not visible in the photo. The ceiling loudspeakers were not used in the test.

Two types of assessments were performed in order to evaluate the characteristics of the voices. First, an intelligibility test was performed, in which short Finnish sentences were presented to the listener either in silence or masked by the noise (63 dB or 70 dB). The listener was allowed to listen to the sample only once. The task

of the listener was to type in what she or he heard. Word error rates (WERs) of the answers were then evaluated. Second, the speech samples were subjectively rated according to three questions with continuous scales with five verbal attributes:

1. How would you rate the quality of the speech sample?
   (bad – poor – fair – good – excellent)
2. How suitable was the speaking style considering the sound environment?
   (bad – poor – fair – good – excellent)
3. How would you describe the speaking style?
   (very irritating – slightly irritating – neutral – slightly pleasant – very pleasant)

The continuous scale ratings, anchored by the five equally spaced verbal attributes, were then scaled to correspond to values from 0 to 100. Every listener rated two sentences of each of the six speech types in all three noise conditions for both genders. Thus each listener rated a total of 72 speech samples in both test types (intelligibility and subjective rating). The test took approximately 1h per listener. A total of 27 listeners (5 females and 22 males) with no reported hearing disorders took part in the test. The listeners were young university students, and they were paid for participating in the test.

3.6.4 Results

The results of the intelligibility test in all noise conditions are shown in 3.6b The WERs are shown for each case with 95% confidence intervals. In silence, all female voice types are equally intelligible with WERs of 1–3%, but for the male voice, the synthetic ones tend to have a slightly larger WER than the natural ones.

The differences in intelligibility with different vocal effort level start to show under the moderate noise condition. Female breathy voices are almost totally unintelligible with WERs around 80%, while normal and Lombard voices have significantly lower WERs, as expected. Male breathy voices do not have as high WERs (around 50%) as the female breathy voices, but they are still generally less intelligible than normal and Lombard voices. The synthetic voices in moderate noise tend to have slightly greater WERs than the natural ones, but the difference is not very large.

In extreme noise, both female and male breathy voices are almost totally unintelligible with WERs of 97–100%. Normal voices are slightly more intelligible with WERs of 74–93% whereas Lombard voices show the smallest WERs of 52–77%. The female Lombard voice is clearly more intelligible than the male Lombard voice. Interestingly, the synthetic male Lombard voice was more intelligible than the natural male Lombard voice. This is probably due to the effect of tuning the extrapolation coefficient in the adaptation, and in this case, the synthetic voice is slightly more “Lombard” than the original Lombard voice.

The results of the subjective evaluations in all noise conditions for female and male voices are shown in 3.6c. The subjective ratings according to the questions and continuous scales with five verbal descriptions are represented as values from 0 to 100 with 95% confidence intervals. The quality ratings show that the natural voices are rated higher than the synthetic voices in silence. The difference in quality ratings between natural and synthetic voices is smaller but still notable in moderated noise, but in extreme noise the differences almost completely disappear.

The suitability ratings of the voices in different sound environments show that breathy and normal speech are rated as very suitable in silence, whereas Lombard speech is rated as slightly less suitable, although still suitable rather than not suitable. In moderate noise, the ratings are reversed. Normal and especially breathy voices are rated low in suitability, indicating that the level of vocal effort was not adequate considering the noise. Lombard voices are rather suitable (fairly or well suitable) in moderate noise. No major differences can be found between natural and synthetic voices with regard to suitability, indicating that the reproduction of vocal effort was successful.

The impression rating measures whether the speaking style was irritating, neutral, or pleasant. The results show no clear evidence that decreased vocal effort would increase pleasantness. For both female and male natural voices

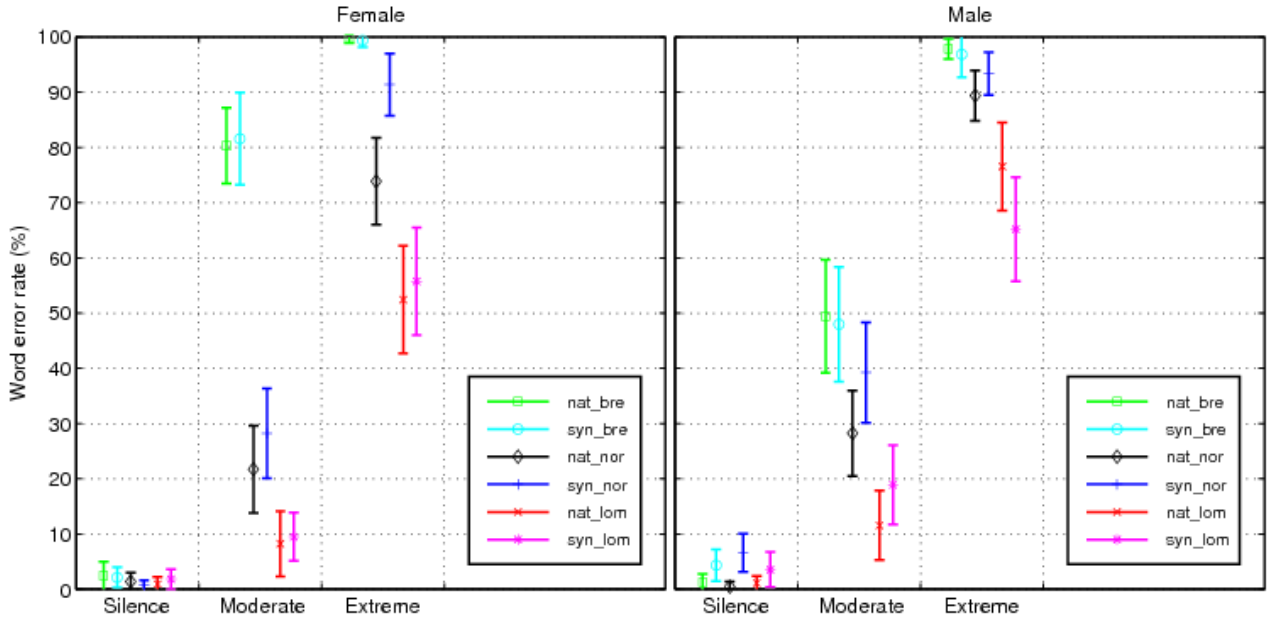

Figure 3.6b: Results of the intelligibility test for female (left) and male (right) voices in three noise conditions: silence, moderate street noise (63 dB, SNR = -1 dB), and extreme street noise (70 dB, SNR = -8 dB).

in silent conditions, normal style was rated as more pleasant than breathy style. For Lombard style, the results diverged.

3.6.5 Conclusions

In this study, the GlottHMM vocoder was used together with adaptation in order to synthesize speech at three points of a vocal effort continuum: breathy, normal and Lombard speech. The synthetic voices were evaluated together with natural speech in terms of their intelligibility, quality, and suitability in three different realistic multichannel noise conditions: silence, moderate street noise, and extreme street noise. The results of the evaluation show that increased vocal effort improves the intelligibility of synthetic voices. Although the synthetic voices have generally slightly higher WERs than natural speech, the reproduction of vocal effort in synthesis was successful. In the case of synthetic male Lombard speech, the WER was even lower than with natural Lombard speech.

The results of the subjective evaluation show that the synthesized voices with varying vocal effort are rated very similarly to their natural counterparts. Especially the suitability ratings of natural and synthetic voices have a very high correlation. Only the quality ratings show a strong separation between the natural and synthetic voices. The Lombard voices are generally more suitable to be heard in the presence of noise compared to conventional voices and they are rated very similar to natural Lombard speech, as was also reported in [20]. However, the breathy voice was not considered more appropriate nor more pleasant in silence than normal speech. This result is consistent with the results of [16]. They reported that speakers did not change their vocal effort substantially within the common communication distances in everyday conversations. Thus, breathy voice is probably a more socially dependent speaking style. On the other hand, breathy voice is also related to expression of emotions. These contexts were not intended to be reproduced in the current experiment and thus breathy voice was not rated more appropriate than normal voice. Nevertheless, breathy voice was rated as very suitable in silence.

The study also showed that the quality of speech does not seem to suffer significantly from adaptation; all the synthetic voices with different vocal effort levels were rated equal in quality in silence. Also an important outcome of the evaluation was that, in the presence of noise, the degradation of speech quality caused by statistical modeling and vocoding loses its significance, thus justifying the use of such synthetic voices in the presence of noise.

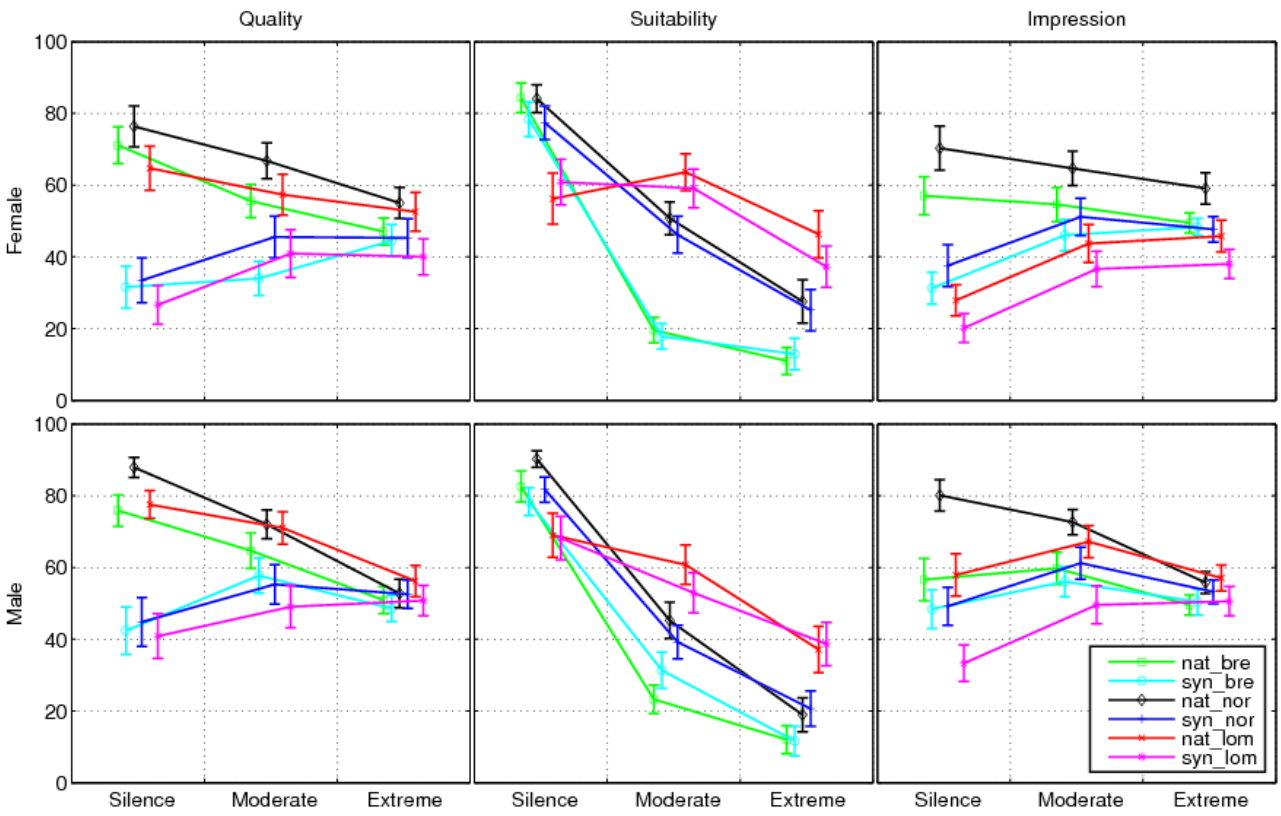

Figure 3.6c: Results of the subjective evaluation for female (upper) and male (lower) voices. The measured quantities are quality, suitability, and impression.

In summary, by collecting user feedback from subjective listening evaluations the study gave encouraging results indicating that our HMM-based speech synthesis technique based on the GlottHMM can be successfully used to create synthetic speech of different speaking styles. In particular, the intelligibility of the synthetic speech in noise can be improved by modeling the Lombard effect.

We consider our study on Lombard speech to be complete and do not plan to run any further experiments on the topic.

As a form of learning from user feedback, this work can be categorized as a *naive* user giving *implicit* feedback of the *full system* leading to *passive* learning which is *supervised* and offline.

4 Planned work

The above section has detailed the wide range of ongoing work – as noted at the start of this deliverable, our strategy has been to explore many different avenues in parallel, since it is not clear which of them will be most fruitful. Many of these avenues will continue to be explored during year 3, and in addition we plan to add a new topic, envisioned in the Description of Work: speech-based feedback.

4.1 Speech-based feedback

Speech is a natural way to communicate corrections to pronunciation or prosody that is perceived by the user to be wrong. For the user, this method of providing feedback in the form of a *correction* should be simple and intuitive. In this speech-based feedback scenario, when a new sentence is synthesised and is not found to be of satisfying

quality, the end user or system developer can speak a correct version of the sentence. This new utterance is then used to improve the system.

At this time, in order to make progress on this challenging task, our investigations are confined to updating the acoustic models (i.e., the context-dependent decision tree-clustered HMMs), leaving the text processing components unchanged. Even with this limitation, due to the statistical nature of the models, it is not obvious how to best use such correction data to improve them. So, the main task is to devise an effective mechanism for improving the models based on limited amounts of user-provided speech. Of course, in all HMM-based speech synthesisers, the models and the corresponding decision trees are always trained from highly-incomplete data, (i.e., only a tiny fraction of the possible models have examples in the training set) and bad pronunciation or prosody may be the result of using the 'wrong model' (because the 'correct model' could not be trained on the given data due to having no examples). The additional data must be somehow used to change this situation so that a 'more correct' model becomes available for use at synthesis time. In the ideal case, such changes to the model set will not only improve the problematic sentence noticed by the user, but also generalise to other utterances which use the same model(s).

4.1.1 Who gives the correction?

In the case where the system developer or user is the one whose speech data is used, or if the developer has access to the original speaker, it will be possible to record new training sentences and add those to the training data pool without any complications.

If the developer or user is not the original speaker, then there will be an acoustic mismatch between the original training data and the newly recorded sentences. It is therefore necessary to define very carefully what parts of the correction speech signal should be used to improve the system. Some parts of the correction data can be harmful and need to be discarded – we do not want the subset of models that have been updated to sound like a different speaker!

In terms of correction to rhythm, stress and other prosodic qualities, it might be possible to only use those acoustic features that most strongly correlate with: energy, F0 and duration information (each appropriately normalised). Spectral components of the correction signal can be discarded, in a fashion similar to what has been done for style transplantation [22, 23].

In the case of a pronunciation error, it will be necessary to use the spectral components for the corrections. This time, it must be ensured that the spectral parameters are transformed into the model space, perhaps by adaptation or voice conversion techniques.

4.1.2 Experiments so far: adaptation using noisy data

We have started by studying whether it is possible to personalize a high quality synthetic voice by providing low quality utterances of the target speaker. One reason for investigating this was that if correction utterances that were recorded by non-expert users in non-ideal conditions decreased the overall voice quality, then speech-based feedback would not be practical. In [24, 25] we showed that, after adaptation using utterances containing various background noises, the voice started to sound more like the target speaker without overly compromising the intelligibility or naturalness. Further details of adaptation using noisy data are given in D1.5.

This finding indicates that the voice of the synthesizer can be personalized by adding more utterances of the target speaker even in conditions where recording clean speech is not possible. In this way, the user can personalise his or her TTS system simply by speaking. However, the speaker adaptation method in current systems is not yet fast enough to allow online utterance-by-utterance evaluation and adaptation 'while-you-wait' so must be done in batch mode.

4.1.3 Planned experiments

The next set of experiments will aim to find out how the amount and quality of additional data affects the retraining of models. Speaker-dependent voices will be built in several languages from speech corpora using only a small amount (300-500 sentences) of the available data. The rest of the training sentences will be synthesised using the small model set, and the synthesised sentences will be evaluated by native listeners based on a simple choice of "adequate quality" and "needs improvement".

The original training sentences that correspond to the poor-quality synthesised sentences are then incrementally added to the training data pool and the voice is retrained. This simulates the situation where the system developer recorded the original training speech data. The quality of these voices will then be compared to voices trained on additional randomly-selected sentences. This should give an indication of how well a synthetic voice improves when users have selected the sentences that most need correction, and give an idea of how much additional data needs to be accumulated to gain an advantage by retraining.

Subsequently, we plan a second experiment to deal with the case where the original speaker and the system developer/user are not the same person. Voice conversion and adaptation techniques will be investigated, as well as using only certain selected acoustic features from the correction utterances.

Some of the many considerations that need to be taken into account when altering the acoustic model sets are:

- How far back do we have to go in re-training? Is it possible just to add extra splits and leaves to existing decision trees – thus adding new models – or is it necessary to retrain and recluster the whole model set?
- Are there cases where (MLLR and/or MAP) adaptation is enough and retraining won't be required at all?
- Are the changes additive or transformative? If the behaviour of the model set is to be changed fundamentally, some kind of 'forgetting' of parameters might be necessary during retraining so that the provided new data is weighted sufficiently highly.
- How much can the system be improved by only updating the the acoustic model, or will it be necessary to make changes to the text processor too?
- We suspect some users will over-emphasise the corrected part of their utterances, as for example some people do when instructing a foreign speaker in correct pronunciation. It must be ensured that users providing new data to a speech synthesis system do not overemphasise the corrections, otherwise the updated models will sound emphatic.

This work can be categorized as a *naive* user giving *implicit* feedback for the *full system*, followed by *supervised offline* learning. Our first study on adaptation to noisy data used *passive* learning, but the planned work of acquiring relevant data will be *active* learning.

5 Conclusion

A number of plans and ideas for the coming project year have been presented in this deliverable, notably Section 4.1, but also in Sections 3.4 and 3.5. A summary plan for the project as a whole is provided in D7.3.

References

[1] Oliver Watts, Adriana Stan, Rob Clark, Yoshitaka Mamiya, Mircea Giurgiu, Junichi Yamagishi, and Simon King. Unsupervised and lightly-supervised learning for rapid construction of TTS systems in multiple languages from 'found' data: evaluation and analysis. In *8th ISCA Workshop on Speech Synthesis*, pages 121–126, Barcelona, Spain, August 2013.

[2] Norbert Braunschweiler and Sabine Buchholz. Automatic sentence selection from speech corpora including diverse speech for improved HMM-TTS synthesis quality. In *Proc. Interspeech*, pages 1821–1824, Florence, Italy, August 2011.

[3] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[4] G. Murray, S. Renals, and M. Taboada. Prosodic correlates of rhetorical relations. In *Proceedings of HLT/NAACL ACTS Workshop, 2006, New York City, USA*, June 2006.

[5] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[6] Mathias Creutz and Krista Lagus. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology, 2005.

[7] Mathias Creutz and Krista Lagus. Inducing the morphological lexicon of a natural language from unannotated text. In Timo Honkela, Ville Könönen, Matti Pöllä, and Olli Simula, editors, *Proceedings of AKRR'05, International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 106–113, Espoo, Finland, June 2005. Helsinki University of Technology, Laboratory of Computer and Information Science.

[8] Sami Virpioja and Peter Smit. Morfessor 2.0: Python implementation and extensions for morfessor baseline. Technical report, Aalto University, 2013.

[9] Mathias Creutz and Krista Lagus. Induction of a simple morphology for highly-inflecting languages. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology*, pages 43–51, Barcelona, Spain, 2004. Association for Computational Linguistics.

[10] Alexandre TRILLA and Francesc ALIAS. Sentence-based sentiment analysis for expressive text-to-speech. *IEEE transactions on audio, speech, and language processing*, 21(1-2):223–233, 2013.

[11] I. Muresan, A. Stan, M. Giurgiu, and R. Potolea. Evaluation of sentiment polarity prediction using a dimensional and a categorical approach. In *Proc of the 7th Int Conf on Speech Technology and Human-Computer Dialogue, SpeD2013, ISBN: 978-1-4799-1063-2*, pages 23–28, 2013.

[12] Taras Zagibalov and John Carroll. Automatic seed word selection for unsupervised sentiment classification of chinese text. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 1073–1080. Association for Computational Linguistics, 2008.

[13] D. Rostolland. Acoustic features of shouted voice. *Acustica*, 50:118–125, 1982.

[14] W. Van Summers, D. Pisoni, R. Bernacki, R. Pedlow, and M. Stokes. Effects of noise on speech production: Acoustic and perceptual analyses. *Journal of the Acoustical Society of America*, 84(3):917–928, 1988.

[15] J.-C. Junqua. The lombard reflex and its role on human listeners and automatic speech recognizers. *Journal of the Acoustical Society of America*, 93(1):510–524, 1993.

[16] Hartmut Traunmüller and Anders Eriksson. Acoustic effects of variation in vocal effort by men, women, and children. *Journal of the Acoustical Society of America*, 107(6):3438–3451, 2000.

[17] J. Yamagishi, T. Kobayashi, Y. Nakano, K. Ogata, and J. Isogai. Analysis of speaker adaptation algorithms for hmm-based speech synthesis and a constrained smaplr adaptation algorithm. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(1):66–83, Jan. 2009.

[18] H. Pulakka, L. Laaksonen, S. Yrttiaho, V. Myllylä, and P. Alku. Conversational quality evaluation of artificial bandwidth extension of telephone speech. *Journal of the Acoustical Society of America*, 132(2):848–861, 2012.

[19] T. Raitio, A. Suni, M. Vainio, , and P. Alku. Synthesis and perception of breathy, normal, and lombard speech in the presence of noise. *Computer Speech and Language*, 2013. Available online 1 April 2013.

[20] T. Raitio, A. Suni, M. Vainio, and P. Alku. Analysis of HMM-based Lombard speech synthesis. In *Proc. Interspeech*, pages 2781–2784, 2011.

[21] T. Raitio, M. Takanen, O. Santala, A. Suni, M. Vainio, and P. Alku. On measuring the intelligibility of synthetic speech in noise – Do we need a realistic noise environment. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4025–4028, Mar. 2012.

[22] J. Lorenzo-Trueba, R. Barra-Chicote, J. Yamagishi, O. Watts, and J.M. Montero. Evaluation of a transplantation algorithm for expressive speech synthesis. In *workshop in Tecnologas Accesibles parte del CEDI2013*, Madrid, Spain, 2013.

[23] Jaime Lorenzo-Trueba, Roberto Barra-Chicote, Junichi Yamagishi, Oliver Watts, and Juan M. Montero. Towards speaking style transplantation in speech synthesis. In *8th ISCA Workshop on Speech Synthesis*, pages 179–183, Barcelona, Spain, August 2013.

[24] R. Karhila, U. Remes, and M. Kurimo. Noise in hmm-based speech synthesis adaptation: Analysis, evaluation methods and experiments. *Selected Topics in Signal Processing, IEEE Journal of*, PP(99):1–1, 2013.

[25] Reima Karhila, Ulpu Remes, and Mikko Kurimo. Hmm-based speech synthesis adaptation using noisy data: Analysis and evaluation methods. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE Internati onal Conference on*, pages 6930–6934, 2013.