The *Papageno* TTS System

Horst-Udo Hain, Jens Racky, Thomas Volk

Siemens AG, Corporate Technology, Munich, Germany
{horst-udo.hain,jens.racky,thomas.volk}@siemens.com

Abstract

One activity of Siemens in the TC-STAR project is to develop a high-quality text-to-speech (TTS) system for UK English. Our main focus is the improvement of the text preprocessing and the acoustic synthesis. Therefore in the second evaluation we took part with the text preprocessing module (task M1) and the whole system (tasks S1 and S2) for UK English. In this article the three modules text preprocessing, prosody generation and acoustic synthesis are described. The results we achieved in the second evaluation are investigated.

1. Introduction

Main focus of the recent development was a TTS system for embedded systems like mobile phones or PDAs. Restrictions like CPU bandwidth and memory consumption had to be considered during the design of that system. Several steps were taken to save memory, for instance the weights of the neural networks were stored as 8bit fixed-point values instead of 32bit floating-point values. Another reason to use fixed-point arithmetic was that most CPUs in embedded systems do not have a floating-point unit. The speaker database was compressed, and only small lexica were used. The result was a TTS system with a memory footprint of less than 500 kilobyte, and on an ARM-9 platform a CPU bandwidth of 50MHz was necessary to achieve speech output in real time. This system is available in seven languages.

The goal for the development within the TC-STAR project is a high-quality TTS system for UK English. This will be achieved by the improvement of all three modules. In the text preprocessing, more detailed rules or algorithms are applied, bigger neural networks (more input context) and huge lexica are used. The prosody generation is based on more information (e. g. POS tags) and also uses bigger neural networks. Main difference is the acoustic part. Here a non-uniform unit selection based on a speech corpus of about 10 hours yields to a more natural sounding voice.

2. Text Preprocessing

The task for the preprocessing module is to prepare the input text for the other modules. The text is split into paragraphs and sentences, numbers and abbreviations are converted, and for every word the part-of-speech tag and the phonetic transcription are determined.

2.1. Text Normalization

The end-of-sentence detection aims at breaking a text into single sentences. In most cases, this can be done by breaking at each potential end-of-sentence punctuation (”.”, ”?”, ”!”). However, sometimes, we face exceptions of this simple rule (at abbreviations, nested punctuations and the like). These are treated by means of word lists which are automatically learned from a training text and contain abbreviation suffixes, typical first words of sentences and abbreviations that are mostly located at the end of a sentence.

2.2. Phonetic Transcription

The grapheme-to-phoneme conversion uses a phonetic lexicon and two neural networks for the out-of-vocabulary handling (Hain, 2000). In a first step, the word is looked up in the lexicon. If it cannot be found there, the algorithm tries to find parts of the word in this lexicon. The transcriptions of these parts are then concatenated. If there are gaps between the parts or at the beginning or the end of the word, or if nothing can be found in the lexicon, then two neural networks are used. The first network generates the phoneme sequence including the syllable boundaries (cf. Figure 1), and the second network determines the position of the lexical stress within the phoneme sequence.

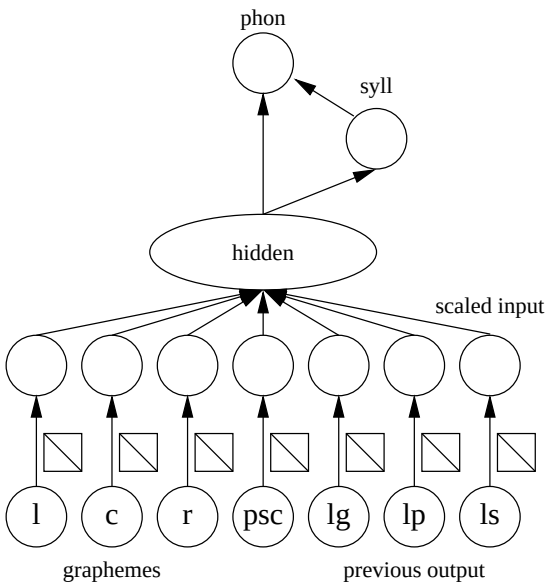

Figure 1: Neural network for phoneme string and syllable boundaries.

2.3. Number Handling

The number type recognition is performed by a rule based approach (Hain, 2005). The rules consist of two main parts: the syntax of the number string and the handling of each part of this number. For example, the US date 04/21/05 in the form `MM/DD/YY` is recognized by the rule

syntax: month { replace 1-2 1-12 } / { word } day { ord 1-2 1-31 } / { word } year { yspell 2-2 0-9 }

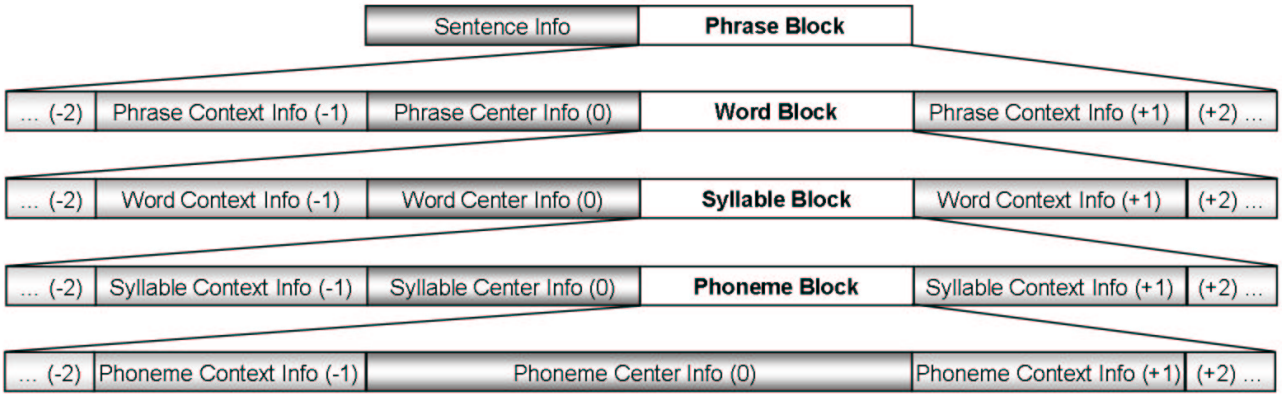

Figure 2: Sentence model for prosody generation.

order: 3 1 5
insert: 2 of
The month has to be a number that consists of 1 or 2 digits, and the number must have a value in the range from 1 to 12. The number will be replaced by the name of the month for the pronunciation (e. g. April instead of 04). The day must have one or two digits with a range of 1 to 31. It is pronounced as an ordinal number. The year must contain two digits, only one digit is not allowed here. The rule covers the special case that the year starts with 0. This is necessary because then the year is read digit by digit (therefore the type yspell and the 0 is read as *oh* instead of *zero*. The parts of the date are spoken in the order 3-1-5 (day-month-year), and the preposition of is inserted between day and month. This results in the pronunciation *twenty-first of April oh-five*.
The phonetic transcription of the numbers is determined by a graph based algorithm (Flach et al., 2000). The digit string is split into smaller parts for which the pronunciation is given in a lexicon.

2.4. Part-of-Speech Tagging

We use a statistical part-of-speech tagger that makes use of n-gram statistics extracted in a training phase. As training data, the Wall Street Journal corpus (Santorini, 1990) consisting of about one million running words was used. By means of an automatic conversion procedure, the original spelling convention (US English) was adapted to the target convention (British English) as required in the framework of TC-Star.We used an n-gram order of 5 and applied a probability smoothing technique based on linear interpolation with weights depending on the training data coverage as suggested in (Sündermann and Ney, 2003). Unknown words were also treated using a linear interpolation technique whose weights were estimated as described in (Samuelsson, 1996).

3. Prosody Generation

In the context of this project the training of the modules for the prosody generation and the preparation of the corresponding database (voice 'Laura', UK) was carried out. The module for the prosody generation, described in the following section, was brought in at the beginning of the TC-STAR project and can be considered as pre-existing.

The prosody generation covers the modules for estimating the prosodic parameters: fundamental frequency as well as duration and energy for each phoneme. The estimated prosody parameters are provided for the succeeding acoustic processing stage and used as target values for the signal manipulation.

3.1. Goal

The quality of a TTS system depends on signal quality on the one hand, and on naturalness of voice on the other hand. Naturalness of voice is reliant on the capability of prosody generation. The goal of our prosody generation module is to achieve a very natural prosody in general. Furthermore the prosody estimation should be expandable to be able to imitate the prosody outline of the original speaker, e.g. for corporate voices. Expandable means that performance and quality of the prosody generation is increasable by the use of additional information delivered from the text preprocessing module, e.g. part of speech tags (POS), and the processing of more context information. The prosody module is scalable regarding quality, computation effort and memory consumption in a wide range and can be adapted to the needs of the appropriate application.

3.2. Approach

We followed the common approach which divides the task of prosody generation into two components. Within the symbolic modeling component "symbolic prosody", the input phoneme stream is processed and enriched with tags that contain information about prosodic intent like accent peaks and boundaries. The acoustic modeling component "acoustic prosody" processes the linguistic representation and generates the prosody parameters.
The symbolic prosody is realized as a multi-layer perceptron (MLP) neural network (Müller and Zimmermann, 2001). The input information is processed on word level and applied from the text pre-processing module. It contains word positions and POS tags in a defined context range. As result of the symbolic prosody the positions of phrase boundaries (full prosodic phrases B3, ToBI labeling conventions) and phrase accents (primary accents PA, ToBI) are estimated.
For the acoustic prosody a new approach for prosody parameter estimation was developed (Hain et al., 2003). Our

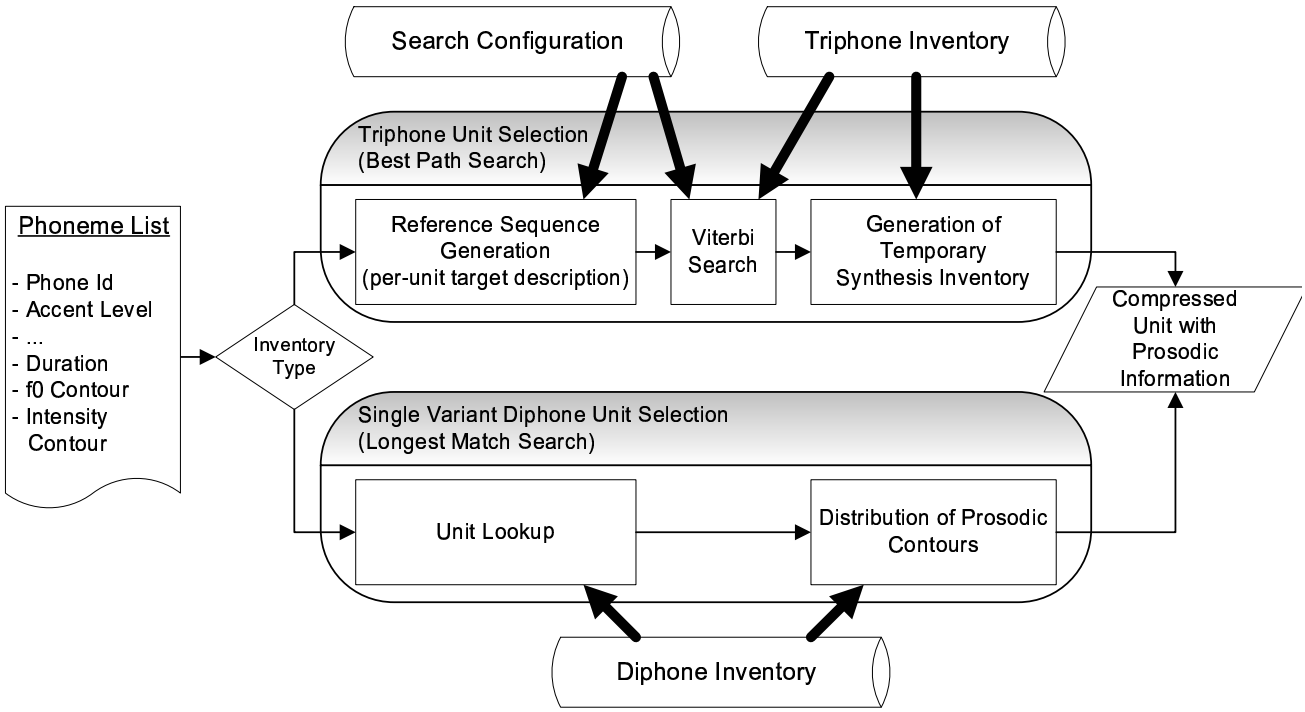

Figure 3: Data flow within the Papageno Unit-Selection Module. It switches between triphone unit selection (upper path) and diphone unit selection (lower path) depending on the active inventory.

approach implements the combination of symbolic and acoustic information within one feature vector for a neural network (NN), and estimates the acoustic prosody parameters (frequency, duration and energy) on phoneme level. The acoustic prosody parameter estimation is performed in the following four steps.

First the sentence structure analysis divides the given sentence, delivered from the preceding text processing stage, into parts like phrases, words and syllables and calculates the positions of these parts. Secondly for each phoneme a set of features is extracted, an input vector is generated, the output values via NN are computed and finally a post processing for outlier detection and correction is performed.

The neural network is realized as an MLP with an input layer, a scaled input layer (input vector scaled by weight-decay-matrix), a hidden layer and an output layer.

In this special approach the feature extraction is based on a new sentence model. Compared with standard procedures this approach offers a scaled view to the complete sentence within each input vector. Our sentence model was designed in the following manner 2: First the regarded phoneme, for which the parameters are to be estimated, is represented as central phoneme within the context phonemes and characterized by its index, type, accent, break level, and position. Then this center phoneme and context phonemes represent a center syllable, which is specified by its length and position. This center syllable and some neighbor syllables describe a center word by length, position, accent level, and linguistic category. This center word and context words describe a center phrase by position, length and accent information. Finally the sentence enfolds the center phrase and context phrases.

4. Acoustic Synthesis

The acoustic synthesis module produces audio data based on the specification (list of phonemes) given by the text preprocessing module (e.g. PhoneId, Accent Level) and enriched by the prosody module (phoneme duration, f_0 contour, intensity contour). As can be seen from fig. 3 the module chooses the unit selection method depending on the type of the active inventory. Within the TC-STAR, (only) the triphone setup has been evaluated, of which the unit selection sub-module is being developed as part of the project.

4.1. Triphone Unit Selection

The Papageno triphone unit selection was designed with the following major constraints kept in mind:

1. Frontend (text preprocessing and prosody) interface must be the same as for the diphone system.
2. It shall drive the existing synthesis backend.
3. Use a compatible structure of binary resource files.
4. It must coexist with the diphone synthesis (runtime switchable).
5. Resource usage (memory and CPU) must be acceptable for midrange (embedded) platforms.
6. Step based processing to keep environment working.

(1) is implemented by transforming the input sequence into a reference sequence (see fig. 3) whose member's data structure is optimized for efficient comparison with inventory entries. To drive the synthesis backend according to claim (2) from the selected inventory units a temporary inventory with entries of the same structure as for the diphone

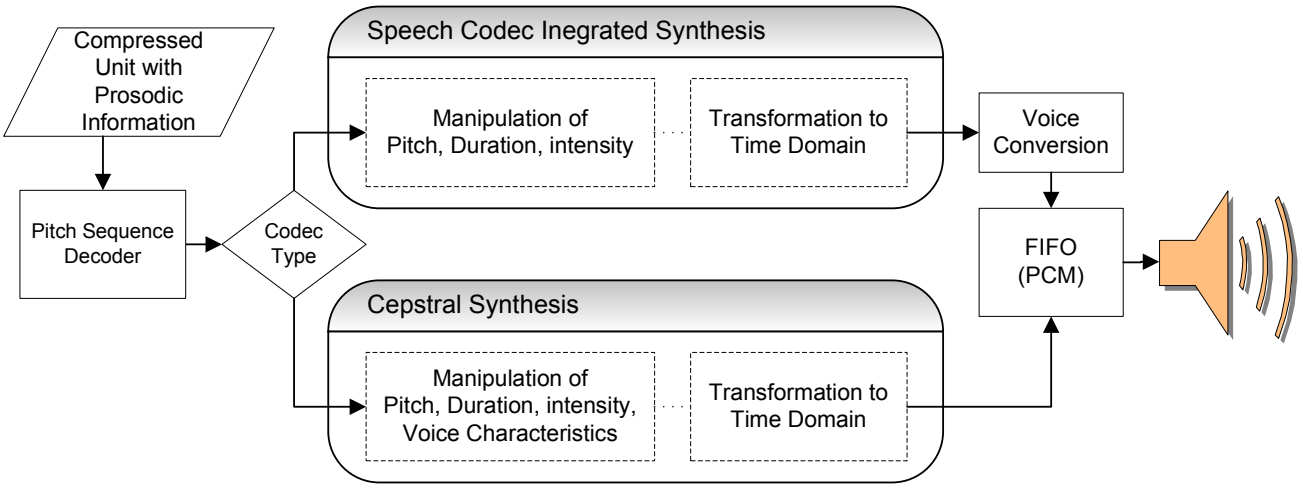

Figure 4: Structure of the audio processing sub-module. The division of the synthesis blocks (*Speech Codec Integrated Synthesis* and *Cepstral Synthesis*) summarize the purpose of the block. It is given as an analogous to a system based on time-domain based manipulation and concatenation.

synthesis is generated (memory intensive data like the audio parameters or the pitch sequence is just linked, though) and referenced by a backward compatible driving sequence. This corresponds with (3) which is based on a compatible inventory header and compatible subfields of the units (e.g. audio parameter storage, pitch sequence storage). Finally the main influence of goal (5) has been to concentrate on features which can be matched inexpensively, thus relying on a carefully selected database (i.e. w.r.t. segmentation and annotation).
The search itself is based on a (Viterbi) best-path search, where the list of the local candidates is ranked (and optionally pruned). Features are individually weighted, the weighting vector depends on the phoneme class for local features, or the pair of phoneme classes for transition features.
Within the inventory there exist units of two kinds:

- Base (Context) Unit:
  Links to 'Real' preceeding and next neighbor (if present), stress level, mean T0, phone ID
- Full Unit (inherited from Basic Unit):
  T0 contour description, class–relative intensity, coded pitch sequence, samples/codec parameters

The search configuration is supplied in a format similar to `.ini` files, with extensions to allow simple specification of groups which share some properties (phoneme classes may be clustered and values may be given by reference instead of direct values). The configuration can be reloaded before each search, in total or in parts. (Selected) Groups are

- Phone Groups (set of phonemes to be treated in the same way, e.g. to which the same configuration):
  e.g. with `phones/unvoiced= C f h k p S s t x` all unvoiced phones can be referenced by writing `phones/unvoiced`.
- Local Fitness Configuration (allows for sharing of configurations between similar phone classes, e.g. using the same configuration for all unvoiced phonemes):
  - `classes` to which to apply to (e.g. `phones/unvoiced`):
  - feature weights (state probability is computed from weighted features), e.g.
    `duration`, `intensity`, `stress`, `pitch/mean`, `pitch/contour`, ...
- Transition Cluster (short–hand notation for all transitions using the same configuration/weights):
  `classes/from`, `classes/to`, `config`,
  e.g. `classes/from=phones/unvoiced`,
  `classes/to=phones/unvoiced`,
  `config=transconf/unvoiced` state that for all transitions between unvoiced phonemes the same configuration (`transconf/unvoiced`) shall be used.
- Transition Configuration (weights of the transitional features):
  `weight/original` (peer is original neighbor), `weight/pitch/level` (pitch transition)

4.2. Unit Selection Analyzer

For comprehensive analysis of the search process the development of an interactive analyzing tool has recently been started(fig. 5).
It's major purposes are to

- visualize the search (i.e. target specification, features values of candidates, features of a specific transition, n-best paths) allowing to quickly find out why which units have been selected
- efficiently compare alternatives (for a given step, for a sub-sequence)
- link to the database to easily
  - show a given (suspect) unit within the original database context

– disable/remove bad units discovered during an analysis session

Figure 5: Interactive search analyzer. The trellis is visualized, nodes show the feature values, transitions the transition score. Alternatives for sub-sequences can easily be selected to allow for efficient comparison “by ear”.

4.3. Audio Processing

The audio processing sub-module of the Acoustic Synthesis Module is the same for both the Triphone Synthesis and the Diphone synthesis mode. It’s structure is shown in fig. 4. Generally this module is configured at compile-time for a specific synthesis method, e.g. a Speech Codec Integrated Synthesis (e.g. AMR, SPEEX) or Cepstral Synthesis. In either case first the pitch sequence has to be decompressed, then the units are manipulated w.r.t. the given prosodic objectives (for special cases these are, however, modified right before, especially plosives need special treatment). With the exception of the cepstral synthesis (which directly includes this feature) the concatenated to-time-domain-converted signal modified by the (optional) voice conversion before it is written to the output FIFO.

5. Evaluation Results

5.1. Text Preprocessing

Text Normalization There are currently no results available.

End-of-Sentence Detection There were 9 errors out of 500 sentences. All errors occured because our system treated every colon as an end-of-sentence, which was not the case in the reference sentences. For example
But let us be honest with ourselves: it would not allow us to meet all of the goals that we have set.
was split into two sentences by our system.

Part-of-Speech Tagging The task here was to tag a text of 100.000 words, but only about 10 percent (10.862 words) were checked. The error rate was 4.7 percent. Table 1 depicts the most common confusions between the reference category and the output of the POS tagger.
One mistake here is that in our system every “to” is tagged as “TO”, but this tag should only be used if “to” precedes a verb (VER or AUX).

reference	hypothesis	percentage
CON	ADP	19.96
DET	PRO	12.87
ADP	TO	12.16
PRO	DET	11.55

Table 1: Confusion of POS tags.

The error rate is higher than in our internal tests. One reason therefore is a mismatch between the knowledge bases that were used to generate the reference on the one hand and for the training on the other hand. The reference was created using the LC-STAR tags including features (case, gender, number), but the system was trained using the tag set of the Penn Treebank Project. These tag sets were then converted by a script to be comparable, but the conversion is error prone and cannot be performed automatically for all cases. This results in a number of mismatches between reference and tagger output, which increases the error rate.

Grapheme-to-Phoneme Conversion Three categories were used in this test: common words (CW), geographic locations (GL), and proper names (PN). Table 2 depicts the phoneme and word error rates for these subtasks.

task	PER (%)	WER (%)
CW	5.6	25.0
GL	19.1	60.3
PN	18.2	55.9

Table 2: Phoneme (PER) and word error rate (WER) for the three subtasks of the grapheme-to-phoneme conversion.

The results for the common words are worse than expected, because in the runtime system a phonetic lexicon with about 60000 entries is used. A more detailed analysis of the errors showed that there are again (as in the first evaluation) several mismatches between the reference and lexicon transcriptions. For instance for the word *advisor*

```
id: (test_0026)
Scores: (#C #S #D #I) 9 0 0 1
REF:  4 d - v " ai - z 4 *
HYP:  4 d - v " ai - z 4 R
Eval:                  I
```

the reference is `/ @ d - v "aI - z @ /`, but the lexicon entry is `/ @ d - v "aI - z @ r /`. This mismatch occurs 23 times, which is about 10 percent of all phoneme errors. Therefore, in the next evaluation a common lexicon should be used for both the runtime system (training of the OOV routines) and the generation of the reference transcriptions.

5.2. Prosody

The prosody generation module was not evaluated by the 2006 evaluation campaign.

5.3. Acoustic Synthesis

The overall synthesis result was dominated by major errors in the database. Thus in this discussion we focus on typical

errors of the database and how they affected the synthesis results. Algorithmic deficiencies, as far as the influence is already known are also mentioned.
As a consequence from these results current work concentrates on the (1) improvement of the database segmentation and annotation process, (2) refinement of the selection method as w.r.t. sensible deficiencies, (3) development/integration of features for rating concatenation eligibility with show a high discriminative performance but which are not necessarily suitable for (runtime) usage in the target environment.

5.3.1. Impact of the Database

The major database related problems where

1. Wrong phonetic transcription, which leads to a wrong segmentation (also of several neighboring units) and in turn to unsuitable audio data.

2. Inadequate transcription, especially in the novel's part, where speech was extremely expressive (e.g. extreme range of f_0, slurring of phonemes, contraction of words), leading to the same consequences as (1).

3. Bad segment bounds positioning leads to wrong feature values and (after selection) to an odd audio signal. Especially boundaries of plosives appeared to be problematic (in the 1st place if no initial silence was present)

4. Wrong/incomplete pitch-marking leads to wrong f_0 features and –in some cases– to wrong resulting f_0 after manipulation.

5. SAMPA phoneme set without extension was used, resulting in segmentation errors in case of syllabic consonants (i.e. `l=` and `n=` cannot be replaced, as it was done, by `@ l` and `@ n`)

Several efforts are now undertaken to mature to manually corrected db part and to improve the automatic consistency and plausibility checking.

5.3.2. Impact of Unit Selection

Several weak areas of the runtime-unit-selection are known, partially the implementation was just missing due to lack of time (checking of the database consumed much more time than planned), partially suitable methods still need to be developed.

1. Selection (ranking) of alternatives for missing triphones: There is currently no ranking of replacement candidates (e.g. an `A:` context may be better substituted by an `O:` context than an `i:` context)

2. Several improvements dealing with plosives are necessary (currently plosives are used without manipulation due to the following problems):

   (a) Intensity needs to be normalized (within the burst) to allow a correct selection and manipulation,

   (b) Duration cannot be reasonably changed.

   Both topics can only be solved by dividing plosives into an initial silence and a burst part, which then can be handled separately (TC–STAR did not allow this so far, though)

3. To better deal with (undetected) problematic database units more sophisticated concatenation features need to be developed, even if their complexity allows their use just for references purposes.

6. Conclusions

Siemens took part in the second evaluation campaign with the text preprocessing module and the whole TTS system. The results of the text preprocessing are not as good as expected. Main reason is the lack of common knowledge resources like a lexicon and a tagged text for the training of the POS tagger. This lead to two problems: (1) The reference transcriptions are different from the entries in our internal lexicon. (2) For the training of the POS tagger the Wallstreet Journal corpus was used which has a different tag set. The conversion of this tag set to the reference set caused several mismatches. The results of the evaluation of the acoustic module were dominated by the errors in the database (mainly wrong phonetical annotation and wrong segmentation). This especially showed the need for (purely) signal dependent transition features, which also ease the rejection of odd units. We hope that for the third evaluation we will be able to use suitable knowledge bases which allow a clear assessment of the algorithms.

7. References

G. Flach, M. Holzapfel, C. Just, A. Wachtler, and M. Wolff. 2000. Automatic learning of numeral grammars for multi-lingual speech synthesizers. *ICASSP*, 3:1291–1294. Istanbul, Turkey.

Horst-Udo Hain, Thomas Volk, and Tim Fingscheidt. 2003. Preprocessing and prosody generation for a TTS system with a very small footprint. *Electronic Speech Signal Processing*. Karlsruhe, Germany.

Horst-Udo Hain. 2000. A hybride approach for grapheme-to-phoneme conversion based on a combination of a partial string matching and a neural network. *ICSLP*, III:291–294.

Horst-Udo Hain. 2005. Classification and pronunciation of numbers for a TTS system. *Electronic Speech Signal Processing*. Prague, Czech Republic.

Achim F. Müller and Hans-Georg Zimmermann. 2001. Symbolic prosody modeling by causal retro-causal NNs with variable context length. *ICANN*.

C. Samuelsson. 1996. Handling Sparse Data by Successive Abstraction. In *Proc. of the COLING'96*, Kopenhagen, Denmark.

B. Santorini. 1990. Part-of-Speech Tagging Guidelines for the Penn Treebank Project. Technical report, University of Pennsylvania, Philadelphia, USA.

D. Sündermann and H. Ney. 2003. synther – a New M-Gram POS Tagger. In *Proc. of the NLPKE'03*, Beijing, China.