We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

7

Multilingual and Multimodal Corpus-Based Text-to-Speech System – PLATTOS –

Matej Rojc[1] and Izidor Mlakar[2]
[1]*Faculty of Electrical Engineering and Computer Science, University of Maribor,*
[2]*Roboti C.S.,*
Slovenia

1. Introduction

Over the last decade a lot of TTS systems have been developed around the world that are more or less language-dependent and more or less time and space-efficient (Campbell & Black, 1996; Holzapfel, 2000; Raitio et al., 2011; Sproat, 1998; Taylor et al., 1998). However, speech technology-based applications demand time and space-efficient multilingual, polyglot, and multimodal TTS systems. Due to these facts and due to the need for a powerful, flexible, reliable and easily maintainable multimodal text-to-speech synthesis system, a design pattern is presented that serves as a flexible and language independent framework for efficient pipelining all text-to-speech processing steps. The presented design pattern is based on time and space-efficient architecture, where finite-state machines (FSM) and heterogeneous relation graphs (HRG) are integrated into a common TTS engine through the so-called ''queuing mechanism''. FSMs are a time-and-space efficient representation of language resources and are used for the separation of language-dependent parts from the language-independent TTS engine. On the other hand, the HRG structure is used for storing all linguistic and acoustic knowledge about the input sentence, for the representation of very heterogeneous data and for the flexible feature constructions needed by various machine-learned models that are used in general TTS systems. In this way, all the algorithms in the presented TTS system use the same data structure for gathering linguistic information about input text, all input and output formats between modules are compatible, the structure is modular and interchangeable, easily maintainable and object oriented (Rojc & Kačič, 2007). The general idea of corpus-based speech synthesis is the use of a large speech corpus for acoustic inventory and for creating realistic-sounding, machine-generated speech from raw waveform segments that are directly concatenated without any or only minimal signal processing. Since only a limited size speech corpus can be used, a compromise between the number of speech units in different prosodic contexts and the overall corpus size should normally be reached. On the other hand, the unit selection algorithm has to select the most suitable sequence of units from the acoustic inventory, where longer units should be favoured. Namely, when using longer units, the number of concatenation points can be reduced, resulting in more natural synthetic speech. The performance of the overall unit selection algorithm for corpus-based synthesis, regarding quality and speed, depends on the solving of several issues, e.g. preparation of text corpus, acoustic inventory construction

using non-uniform units, reduction of unit search space, detection and removal of acoustically very similar units, off-line calculation of concatenation costs between all speech units in the acoustic inventory, their efficient representation, and their fast access within the on-line system. Further, the optimisation of weights used within cost function is an important issue, since these weights mainly influence the unit selection process performance regarding synthesised speech quality and naturalness (Black et al., 1997; Christophe et al., 2002). In the presented design pattern for corpus-based TTS systems, a gradient descent based unit selection optimisation algorithm is proposed for optimising unit cost functions' weights. Furthermore, the presented unit selection process addresses issues, such as: efficient acoustic inventory construction, reduction of unit search space, detection and removal of acoustically similar units, calculation of the concatenation costs, efficient representation of concatenation costs, and fast lookup. An important aspect of the presented cost functions' weights optimisation is that it also reduces laborious manual involvement when preparing new voices and tuning the best possible quality of the corpus-based TTS system. No matter what age, cultural background, or even what language people might speak, facial expressions and different body gestures always occur in natural human-human dialogues. Even when the dialogue is not face-to-face, people are prone to describing key issues by using different facial expressions or even by hands that remain free. Therefore, the first reason for using non-verbal modalities together with the TTS system, is to better emulate the natural course of the dialogue, and to make people feel more comfortable when ''speaking'' to a machine. The second reason is hidden in those issues that occur during the usage of human-machine interaction systems. The need to repeat and the misinterpretation of speaking terms are common features regarding the majority of users. Such behaviour usually leads towards less-functional and less-efficient spoken dialogue systems (Cassell, 2000). If we were to have more appropriate social responses from the machine through personification of the TTS system by using embodied conversational agents (ECA), people will more readily respond with emotive socially-coloured responses. Therefore, human-human-like communicative behaviour may be evoked in this way, giving the spoken dialogue system the ability to shape and adjust the dialogue to its own rules. TTS systems and believable characters (ECAs) can be used together to evoke communicative behaviour. ECAs can often, by expressing social tendencies, shape and also lead the dialogue. Understanding of attitude, emotion, together with how gestures (facial and hand) and body movements complement, or in some cases, override any verbal information produced by the TTS system thus providing crucial information for modelling both the dialogue and the ECA's socially-oriented responses. The social response (naturality) of the TTS system fused with ECA can then be presented to the user in a more human-like form, using not just audio but also facial expressions, such as: facial emotions, visual animation of synthesised speech, and correlated head, hand, and body movements. Therefore, personificated TTS systems enable the development of more advanced, personalized, and more natural multimodal-output-based human-machine interfaces that are in demand more and more for today's applications and environments. The time and space-efficient architecture of the corpus-based TTS system is presented in Section 2. The unit selection process for corpus-based TTS systems is then described in Section 3. The next section describes in detail the novel EVA framework that enables personification of the general TTS system. Slovenian implementation of the multilingual and multimodal corpus-based PLATTOS TTS system is presented in Section 5. A novel approach to distributive evaluation and the testing of TTS systems is presented in Section 6. Conclusions are drawn at the end.

2. Time and space-efficient TTS architecture

The corpus-based TTS architecture of the PLATTOS TTS system presented in Figure 1 is modular, time and space-efficient, and flexible (Rojc, 2003; Rojc & Kačič, 2007). By following the multilingual aspect, the language-dependent resources are separated from the language-independent core TTS engine. Its modular structure allows for all modules within the system to be easily maintained, and further improved by easy integration of new algorithms into the TTS system.

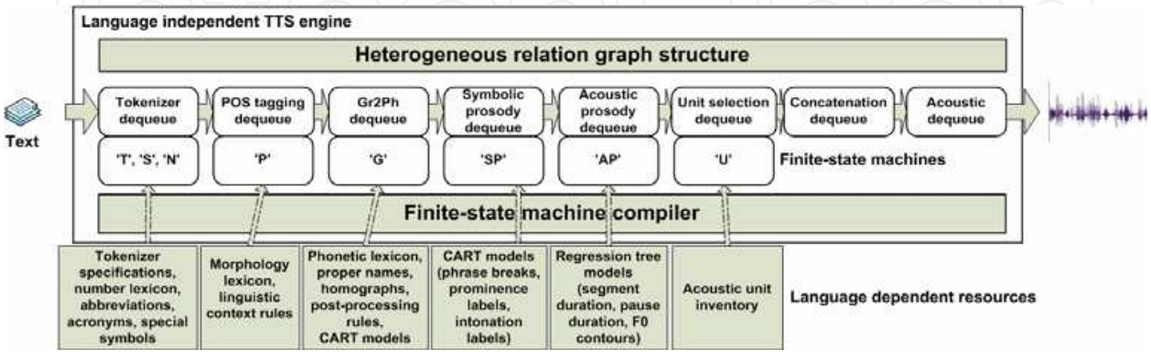

Fig. 1. The time and space-efficient architecture of the corpus-based TTS system.

2.1 Queuing mechanism used in the TTS architecture

An efficient queuing mechanism is implemented in the presented TTS architecture (Rojc & Kačič, 2007), where each double-linked list is used for one processing step in the TTS system. In this way all TTS processing steps are pipelined together. A queuing mechanism enables flexible addition and removal of dequeues from the mechanism, thus allowing for the merging of already existing processing steps, or adding new ones. The overall text-to-speech process runs in a loop, when processing the input text. All TTS engine dequeues are empty at the start. Firstly, the tokenizer module starts generating tokens from the input text by using a finite-state machine (FSM) based lexical scanner. Two additional token types are added for marking end-of-sentence or end-of-file conditions. These two tokens are only used for controlling the overall queuing mechanism. Immediately after detection, either of these two tokens, the following part-of-speech (POS) tagging dequeue is activated, taking all

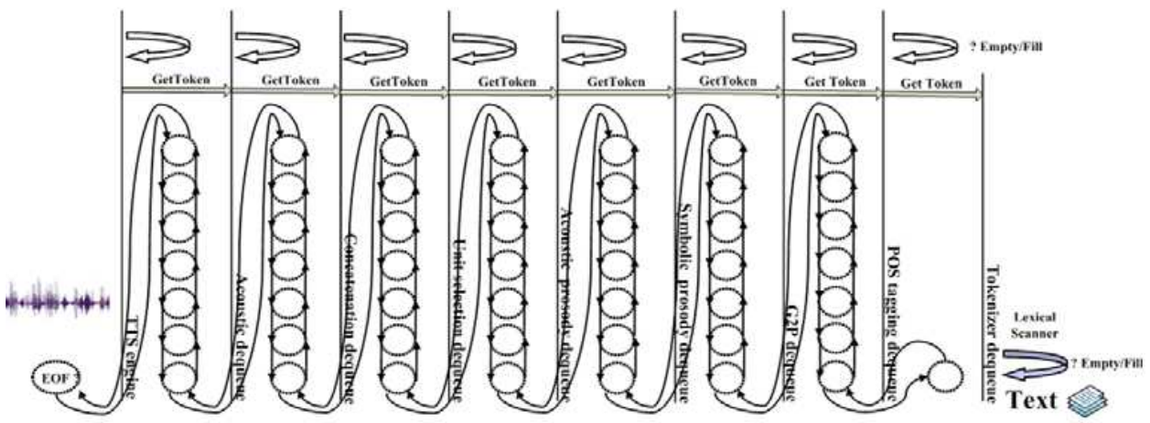

Fig. 2. The queuing mechanism.

tokens from the previous tokenizer dequeue (for the current sentence). After the tagging process, the grapheme-to-phoneme (G2P) conversion dequeue activates and grabs all tokens from the POS tagging dequeue. In this way (at the sentence level) the text-to-speech process continues until acoustic dequeue, where the speech signal for the corresponding sentence is finally generated. All text-to-speech processing steps are sequential processes. Nevertheless, the processing of several sentences within the presented queueing mechanism can run in parallel, by processing each sentence within its own thread. At the end, only the correct order from the input must be preserved, before playing-out generated speech signals.

2.2 Heterogeneous relation graphs used in the TTS architecture

All TTS processing steps contribute to the linguistic information used for generating the speech signal. The heterogeneous relation graph (HRG) structure provides clean general-purpose mechanisms for storing and representing all the information extracted by the TTS system (Rojc & Kačič, 2007; Taylor et al., 2001). In the PLATTOS TTS architecture, one HRG structure is used per each text sentence, and is accessible by all dequeues used in the TTS system. In this way, all algorithms are able to access, change, or enrich stored information when appropriate. Figure 3 illustrates the integration of the HRG structure into a queuing mechanism. The HRG structure demonstrates the use of two different relation-structures for storing extracted information, linear lists and trees. The linear lists are named Segment, Syllable, Word, Phrase, IntEvent, and SynUnits in Figure 3, whilst the tree structures are named SyllableStructure, PhraseStructure, IntonationStructure, SynUnitsStructure. The linguistic objects within the relation-structures are e.g. words, syllables, segments, phrase-breaks, intonation events, synthesis units, enriched with several attributes determined by the algorithms used within the processing dequeues. Attributes are the properties used in TTS system modules, e.g. part-of-speech, duration, phone-class and properties, intonation event type, phrase-break type, prominence label-type, to name just a few. Linear lists are used to specify the relation between linguistic items found in the specific processing step.

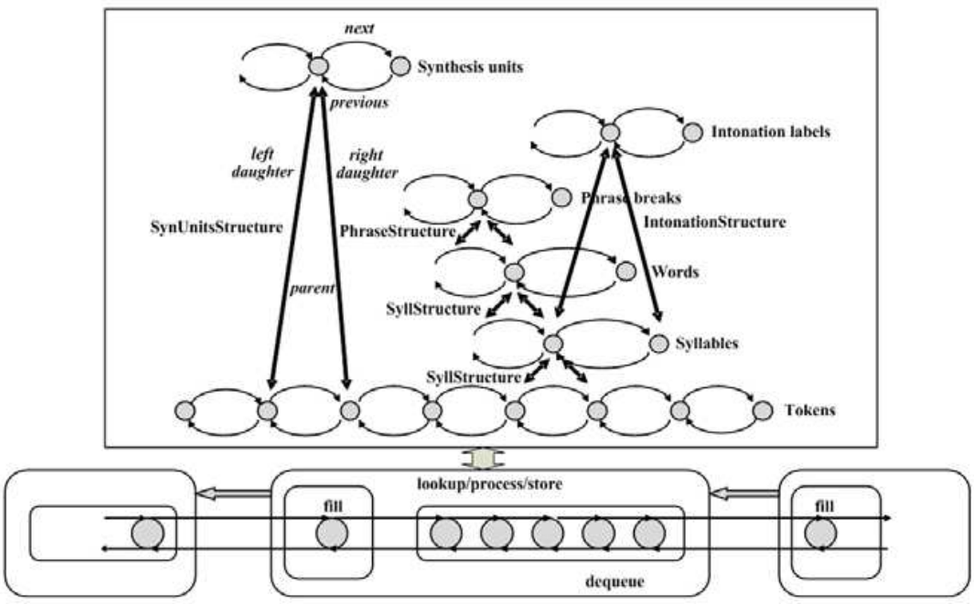

Fig. 3. Interaction of a queuing mechanism and a HRG graph structure.

Forward and backward traversals are possible within the structure. Additional tree relation-structures add vertical information between those linguistic objects included in different linear lists. In this way, very complex features for machine-trained models (e.g. CART trees, NNs etc.) can be generated from the linguistic information stored in the HRG structure, without any additional processing or extra work on feature construction. Furthermore, the relation-structures used within a HRG structure can easily be changed and adapted to different structures, following the processing needs of the modules used in the TTS system.

2.3 Finite-state machines used in TTS architecture

For multilingual and polyglot speech synthesis systems, it is important that the migration to new language can be done with little or no intervention in the algorithms used. This can be achieved by separating language-dependent language resources from the TTS engine, and obtaining a language-independent TTS engine. The efficient separation of language-dependent language resources is done within PLATTOS TTS architecture by finite-state machines (FSM) (Mohri, 1995; Rojc & Kačič, 2007). Furthermore, FSMs are also used for the representation of language resources and linguistic rules. FSMs can be constructed off-line and loaded into the TTS engine during on-line operation. The corresponding representation offers fast lookup, since the lookup does not depend on the size of the dictionary but only on the length of the considered input string. Minimization algorithms allow one to reduce the sizes of these devices to a minimum. The FSM compiler is used for the compilation of several regular expressions into the finite-state machine, construction of finite-state machine-based tokenizers, etc. In order to solve disambiguity problems, heuristically-defined or trained weights are assigned to FSM transitions and final states, yielding weighted finite-state automata and transducers (WFSA, WFST) that can be integrated into the TTS architecture (Mohri, 1995). In Figure 1 the tokenizer is marked as 'T' in the TTS architecture. At this processing level two-level rules or rewrite rules can be used, and compiled into finite-state machines by an FSM compiler (Mohri, 1996). Namely, these rules can resolve much of the language-dependent disambiguity present in the input texts. TTS system processes any given input text that often contains more or less spelling mistakes (e.g. e-mails, SMS messages). Therefore, the finite-state automaton 'S' follows (represents efficiently large lists of valid words), and is used by the spell-checking system (if it is included in the architecture). Using them, the spell-checking system is able to detect invalid words and can guess the most suitable replacements. Next, the POS-tagging module needs large-scale morphology lexicons. Therefore, the finite-state transducer 'P' can be used here for time and space-efficient representation of large-scale morphology lexicons. If TTS systems use rule-based POS-tagging algorithms (e.g. Brill, 1993), the POS-tagging rules can be further compiled into finite-state machines, and become a compact part of the TTS architecture (Emmanuel & Schabes, 1997). The grapheme-to-phoneme (G2P) conversion module uses, in general, large-scale phonetic lexicons for common words, proper names, and even foreign words, as found in the input text. All these resources can be represented by the finite-state transducer (FST) 'G', as presented in Figure 1. Decision-tree models can be included in the TTS architecture, since they represent efficient knowledge representation regarding time and space requirements. They can be used in the prosody modules (symbolic and acoustic prosody) for the prediction of phrase breaks, prominence and intonation event labels, segment durations, pauses between segments and the acoustic parameters of intonation events. Nevertheless, it has been shown that decision trees can also be represented by weighted finite-state machines (labelled as WFST 'SP', WFST 'AP') (Sproat & Riley, 1996). However, this step only makes sense when they are going to be merged with all

other finite-state machines, as decision trees are already efficient knowledge-representation structures. In corpus-based TTS systems, the unit selection search process represents a significant time and space issue (large unit search space). Finite-state machines can be used here for more efficient access to unit candidates stored in the acoustic inventory. In the concatenation and acoustic modules, digital signal processing algorithms are used for the processing of concatenation points, and for adapting unit candidate pitch and duration and, in general, no external language-specific resources are needed.

3. Time and space efficient unit selection in corpus-based TTS systems

All the data-preparation steps needed for general corpus-based TTS systems are shown in Figure 4. The acoustic inventory and concatenation costs (have to be represented in a time and space efficient way) calculated between unit candidates are the result of these data preparation steps. The optimality and quality of the final acoustic inventory (used by the unit selection process) depends on several previous steps e.g. text-corpus construction, segmentation, phonetic tree-based clustering of units, and the acoustic evaluation of unit candidates. The last step takes care of removing acoustically similar units (the so-called redundant units) that are unnecessary in the optimised acoustic inventory. Calculation of concatenation costs then follows with a quantisation-based compression of these, and their space and time efficient representation, where the concatenation costs' matrix indices can be stored in the form of FSM.

3.1 Acoustic inventory construction

In corpus-based TTS systems the idea is to use the whole speech database for acoustic unit inventory, selecting the longest possible existing phonetic segments, and matching the target unit's specification, as defined for the target sentence. Because of the complexity and combinatorics of languages, it is important to find the best compromise: that has, on the one hand, as small a speech database as possible and, on the other hand, 'enough' acoustic realizations of those units found in several phonetic and prosodic contexts. Defining such a compromise is one of the major issues for the corpus-based speech synthesis approach (Bozkurt et al., 2003, Rojc, 2003). In PLATTOS TTS architecture diphone and triphone units are used within a unit-selection algorithm, where the diphones are base units. The richness of text corpus (regarding diphones and triphones) then has a significant impact on the richness of the acoustic inventory, on the performance of the unit-selection algorithm, and on the expected naturalness of the synthesised speech signal. A detailed analysis of several tokenised text corpora has to be performed in order to collect the appropriate text of a given language when striving to good acoustic inventory at the end.

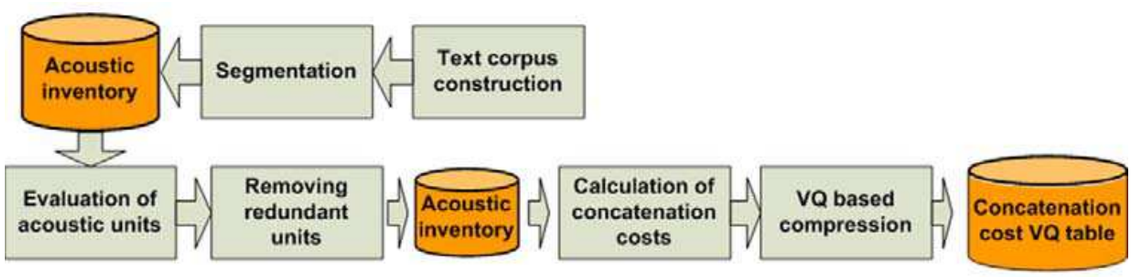

Fig. 4. Data preparation steps for the efficient unit selection process (off-line process).

After constructing a text corpus and recording speech database, the obtained database is segmented into unit candidates. Automatic segmentation procedures are preferred for segmentation of the database during the first step, but for optimal quality at least some manual checking usually follows. Better results can be expected when canonical phonetic transcriptions are verified and adapted to the recorded speech material, before running automatic segmentation. The final size of the constructed acoustic inventory is important in order to meet real-time requirements. In PLATTOS TTS architecture the starting acoustic inventory consists of a large set of non-uniform units (diphones and triphones). It can be expected that an acoustic inventory constructed directly from a segmented database, will contain acoustically similar units (can be qualified as redundant units) that can be removed. In order to detect these units, all units have to be acoustically evaluated regarding pitch, duration, and energy. When the text corpus is well-defined, the recorded speech material will more probably contain units with several distinct acoustical realisations, have less redundant units and, consequently, will allow for a better quality of synthesised speech from general input texts.

3.2 Acoustic inventory optimization

The search space for a unit-selection algorithm can already be reduced off-line during acoustic inventory construction, and also during the on-line unit selection process (within the TTS system) (Campbell & Black, 1996; Holzapfel, 2000). In the PLATTOS TTS system's unit selection approach, the reduction of the search space is proposed as a two-stage process (performed off-line). During the first stage, the diphone and triphone unit candidates are clustered according to their phonetic context and, during the second stage, acoustically similar units (similarity measurements are determined by considering pitch, duration, and energy) are automatically detected, and removed within the constructed tree-clusters. In order to detect acoustically similar candidates within the tree-clusters, detailed acoustic analysis is performed on all the cluster's unit candidates. Detecting acoustically similar units and removing them from the acoustic inventory can be performed in a manner analogous to the perceptual stimuli relationship. The final decision about which units should remain in the constructed clusters is done after the acoustic characteristics of all the clusters' unit candidates are obtained during analysis. The final optimised acoustic inventory contains, for each specified cluster, *n* units that are then used in the unit selection algorithm. The concept of suitability functions can be used in order to rank the unit candidates, where the setup and tuning of suitability functions can be performed by using a hybrid approach (Holzapfel & Campbell, 1998). First, the mean values of energy, pitch, and duration are calculated for each obtained tree cluster. All the mean values then represent those target values having a suitability value of 1.0. Other target values for energy, pitch, and duration, are defined for each unit candidate within a specific cluster by using the cluster's suitability functions' shape. Partial suitability functions must reflect acoustic differences between unit candidates within a specific cluster. Differences in duration amongst the units in the cluster 'i' are represented by using the following partial suitability function:

$$S\left[i\right]_{partial}^{dur} = \begin{cases} e^{-\frac{1}{2\cdot a^2}\cdot\left(\frac{dur-dur_{mean}}{dur}+a\right)^2}, & \frac{dur-dur_{mean}}{dur} < a \\ 1, & -a \le \frac{dur-dur_{mean}}{dur} \le b \\ e^{-\frac{1}{2\cdot b^2}\cdot\left(\frac{dur-dur_{mean}}{dur}-b\right)^2}, & \frac{dur-dur_{mean}}{dur} > b \end{cases} \quad (1)$$

Differences in pitch amongst the units in the cluster 'i' are calculated by using the following partial suitability function:

$$S[i]_{partial}^{f0} = \begin{cases} e^{-\frac{1}{2\cdot c^2}\cdot\left(\frac{f_0 - f_{mean}}{f_0}\right)^2}, & f_0 - f_{mean} < 0 \\ e^{-\frac{1}{2\cdot d^2}\cdot\left(\frac{f_0 - f_{mean}}{f_0}\right)^2}, & f_0 - f_{mean} \geq 0 \end{cases} \quad (2)$$

Differences in energy amongst the units in the cluster 'i' are calculated by using the following partial suitability function:

$$S[i]_{partial}^{en} = \begin{cases} 1, & en - en_{mean} > 0 \\ e^{-\frac{1}{2\cdot e^2}\cdot\left(\frac{en - en_{mean}}{en}\right)^2}, & en\text{-}en_{mean} \leq 0 \end{cases} \quad (3)$$

In this way, the overall unit candidate's suitability is ultimately defined by combining partial suitability functions for pitch, duration and energy within the given cluster:

$$S_{overall} = \prod_{i=1}^{N} S_{partial} \quad (4)$$

At the end of the ranking process unit candidates are ranked within the region of 0 to 1 in all tree clusters. This region is then divided into smaller sub-regions. Suitability values within a specific sub-region correspond to those candidates that have similar acoustic characteristics, meaning that they have small or insignificant differences regarding pitch, duration, and energy. In this case, only one candidate from a specific sub-region is kept in the optimised acoustic inventory, and all the others can be removed. Multiplication of the used partial suitability functions ensures that differences in certain acoustic parameter are noticeable within the overall suitability value for each unit candidate, and that the significance of a particular acoustic feature is reflected by the shape of the used partial suitability function.

3.3 Concatenation cost calculation and representation

The calculation of concatenation costs (CC) is a very time-consuming step for corpus-based TTS systems, especially if performed during the on-line unit selection process. Namely, the CC costs must be calculated between all phonetically-matched candidates for each of the two successive target units in the current sentence. Then, in order to evaluate any distortions at concatenation points, the corresponding speech samples of all these candidates have also to be loaded. In order to avoid this, the obvious solution can be the off-line calculation of all CC costs. The disadvantage of this solution is that the target unit sequences are unknown and, therefore, consideration of any phonetically-matched candidates in the acoustic inventory must be taken into account. Furthermore, concatenation costs have to be calculated between all unit pairs in the acoustic inventory (for large databases, non-uniform acoustic inventories can have a lot of units), and this results in large CC cost-matrix dimensions and storage requirements. In order to also solve this problem, the vector quantisation algorithm (VQ) can be used. By using the VQ technique, we are able to compress a CC cost-matrix into a much smaller one. This whole process can be easier

performed by first splitting the large CC cost-matrix into smaller sub-matrices (speed and memory problems). The CC costs for each pair of candidates are calculated for each sub-matrix. The calculated costs within each sub-matrix are then quantisized into a corresponding codebook of a pre-defined size (number of clusters) (Rojc & Kačič, 2007). Without using vector quantisation, the storage requirements for each sub-matrix are (W – size of the sub-matrix):

$$Storage = N \cdot W \cdot sizeof(float) \tag{5}$$

After using vector quantisation, the storage requirements drop down to:

$$Storage = VQ \cdot W \cdot sizeof(float) \tag{6}$$

VQ represents the codebook dimension regarding a pre-defined size for any specific sub-matrix. After calculating codebooks for all sub-matrices, they are merged into one common codebook, representing the compressed CC cost matrix. This representation is a space-efficient representation of concatenation costs between all candidates in the acoustic inventory. An index table is also built in addition to the constructed codebook, and used for accessing the concatenation costs. In order to also make the CC cost lookup also time and space-efficient, the CC cost indices are stored in the form of a finite-state transducer (FST) (Mohri, 1997). In this way we are able to perform an efficient lookup process in the unit selection algorithm.

3.4 On-line unit selection algorithm

The unit selection algorithm is a very important process in corpus-based concatenative speech synthesis, since it searches for the best matching sequence of unit candidates with those target units specified for the input sentence. The selection of non-uniform units (diphones and triphones) from the acoustic inventory is based on minimising those acoustic distortions that originate from concatenations, and minimising the needed modifications of the unit candidates. In the PLATTOS TTS architecture, these distortions are described in the form of two costs:

- target cost $C^t(u_i, t_i)$: represents an estimation of the difference between unit candidate u_i in the acoustic inventory and target unit specification t_i,
- concatenation cost $C^c(u_{i-1}, u_i)$: represents an estimation of the quality of the concatenation of two successive units u_{i-1} and u_i.

Target unit specifications include e. g. phonetic symbol, symbolic prosody information (e.g. stress indication), acoustic prosody information (e.g., desired unit duration and F0) etc. They are used for calculating target cost (TC) in the on-line unit selection algorithm. Concatenation cost (CC) is already calculated off-line (as suggested), and accessed through an efficient lookup process. Common cost then reflects the differences in target and acoustic realisations for specific unit candidates, and the expected distortions when the selected unit candidates are concatenated together. In corpus-based TTS systems, a pitch and duration modification algorithm (e.g. TD-PSOLA) is often applied to pre-stored candidates in the acoustic inventory, in order to guarantee that the prosodic features of synthetic speech meet the predicted target values. Then, using a good criterion for finding the best fitting unit sequence from the acoustic inventory is crucial for generating high quality speech. In the PLATTOS TTS architecture unit selection algorithm, the following equation is used for calculating the common cost for each unit candidate:

$$C(u_i) = C^t(u_i, t_i) + C^c(u_{i-1}, u_i)$$
$$C(u_i) = w_{unit} \cdot \left[\prod_{j=1}^{P} S_{partial} \right] + w_{unit} \cdot w_c \cdot \left[(S_{conc})^{w_{local}} \right] \quad (7)$$

It follows from equation (7), that the target cost uses partial-suitability functions for duration, pitch, and energy (equations (1), (2) and (3)). The mathematical framework behind the computation of this common suitability is based on fuzzy-logic (Holzapfel & Campbell, 1998). The performance of the unit selection algorithm and, consequentially, the quality of the synthesised speech, significantly depends on the partial suitability functions' parameters. Furthermore, weight w_{local} is additionally included in order to have control over the calculated concatenation cost between two unit candidates. Weight w_{unit} is included in order to favour the selection of longer units during the unit selection process (e.g. triphones, instead of diphones). Finally, weight w_c is included in order to control the influence of concatenation cost on the common cost $C(u_i)$. And S_{conc} represents the distortion measure between two successive unit candidates, based on an acoustic cost that is calculated by using signal processing based on spectral analysis (suggested to be performed offline).

3.5 Gradient-descent based unit selection process optimization

An important common cost calculation issue is the optimal setup and tuning of those parameters used within partial-suitability functions (equations (1), (2) and (3)), and other weights used in equation (7). Parameters a, b, c, d, and e influence the shapes of the partial-suitability functions and, consequently, influence the significance of a particular criterion (duration, pitch, energy) within the unit selection process. Furthermore, searching for the best unit sequence using a unit selection algorithm is a multidimensional problem. Heuristics is usually used for setting up parameters and weights, or extensive subjective listening tests are performed, resulting in more or less optimal solutions. Such an approach is at least time consuming and laborious. Besides, parameters and weights have to be adapted for each new TTS voice. Instead, the PLATTOS TTS system uses an automatic optimisation approach of cost function's weights based on a relaxed gradient descent algorithm (RGD) (Figure 5).

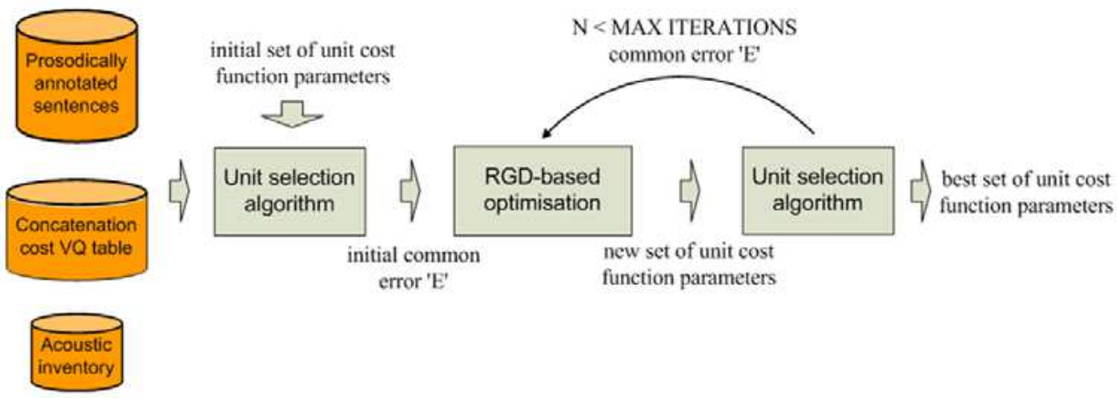

Fig. 5. Unit-selection optimisation process.

The input into the unit-selection optimisation process is a set of prosodically annotated sentences (HRG utterance structures), off-line calculated CC costs, and acoustic inventory. The unit selection algorithm then selects a sequence of units for each input sentence by using an initial setup of weights (the initial setup of values is set by a hybrid approach, as proposed in (Holzapfel & Campbell, 1998)). Automatic unit selection process evaluation is performed during the next step, by calculating the pitch deviations of neighbouring selected candidates, and deviations between selected candidates' durations and those durations predicted by prosody. The obtained evaluation result represents initial common error 'E' for the unit selection optimisation process. The process then keeps running within a loop, and the weights and parameters are iteratively updated by using the RGD technique. The optimisation loop consists of several processing steps. During each iteration, a common error 'E' is calculated as the sum of pitch differences (between predicted pitch $\hat{f}_0(k)$ and the selected unit candidate's pitch $f_0(k)$), and the duration differences (between predicted duration $\hat{d}(k)$ and the selected unit candidate's duration $d(k)$):

$$E = \sum_{k=1}^{N} \left[e(k) \right]^2 \quad , \quad e(k) = \frac{\left| \hat{f}_0(k) - f_0(k) \right|}{\hat{f}_0(k)} + \frac{\left| \hat{d}(k) - d(k) \right|}{\hat{d}(k)} \tag{8}$$

The computation of common error 'E' includes differences in durations (time) and F0 values (frequency). Therefore, differences in duration and F0 value are normalized in equation (8). In order to minimize these differences (and common error 'E'), the RGD technique is used, optimizing the initial setup of the unit-cost functions' weights and parameters. In other words, the goal is to minimize the cumulative common error 'E'. All weights and parameters are stored in a vector **p**:

$$p = \left[p_1, p_2, ..., p_l \right] \tag{9}$$

This vector is then iteratively updated in such a direction that the change results in a smaller common error 'E'. This direction is searched for by a gradient calculation, performed for each value in vector **p**: $\nabla E(p)$. An adjustment of each vector value is then performed by the following update rule:

$$p_{n+1} = p_n - diag(\mu_n) \cdot \nabla E(p_n) \tag{10}$$

The obtained gradient vector consists of the partial derivatives of the unit cost functions, with respect to each value of the vector **p** (e.g. partial derivative of 'E' with respect to p_1):

$$\frac{\partial E}{\partial p_1} = 2 \cdot \sum_{k=1}^{N} e(k) \cdot \frac{\partial e(k)}{\partial p_1} \tag{11}$$

The adaptation rate μ must be selected so that the convergence of the algorithm is guaranteed, since the performance of the algorithm is quite sensitive to a proper setting of the adaptation rate. Namely, if the adaptation rate is too high, the algorithm may oscillate and become unstable. On the other hand, if the adaptation rate is too low, the algorithm will take too long to converge. The optimisation process is repeated, until obtaining the

predefined minimal error. When this happens, the set of weights and parameters is stored, and can be used for the on-line unit selection algorithm in the corpus-based TTS system. The approach is fully automatic, and can be repeated for each new voice used in the TTS system.

4. Personification of the TTS systems

The idea of advanced human-machine interfaces and spoken dialogue systems is to emulate natural and highly-complex human-human interactions. Substantial effort by several researchers has already been devoted to this task, by taking into account multimodal-input and multimodal-output contexts. An understanding of attitude, emotion, together with how gestures (facial and hand) and body movement complements, or in some cases even overrides verbal information, provides crucial information about modelling interactive management. It influences both input and output perspectives of the realization of natural human-machine interaction and, consequently, the personification of those TTS systems used in e.g. spoken dialog systems. Personification of TTS systems, therefore, not only relates to the transformation of a TTS system's output into ECA's visually-presented articulation within the mouth region (visualizing verbal behaviour), but also to the visualization of non-verbal behaviour. The most natural way to visualize (emulate face-to-face conversation) both verbal and non-verbal information is to translate it into a human-body representation. Embodied conversational agents (ECA's) are widely used concepts for the visualization of conversation and are used in many spoken dialogue systems. ECA implementations range from talking heads (Poggi et al, 2005), to agents that can move and use the whole representation of the human body (Heloir & Kipp, 2009; Thiebaux et al., 2008). There are many implementations of ECA's that can, in one way or another, emulate natural human behaviour and evoke emotional and social responses within human-machine dialogue. (Ball & Breese, 2000) describe the generation of emotional responses and the recognition of emotions by humans, and the additional adaption of ECA's personality to that of the human. (Poggi &Pelachaud, 2000) generate communicative behaviours on the basis of speech acts and concentrate on one facial expression and speech act performatives. Performatives are a key part of the communicative intent of a speaker, along with propositional and interactional acts. In general terms, all conversational behaviour in conversational models must support conversational functions and different input/output modalities. Any conversational action in any modality can result in several (sometimes contradictive) communicative goals. The general architecture of a system that can visualize and personificate a general TTS system, used in e.g. a spoken dialog system, is formed as shown in Figure 6.

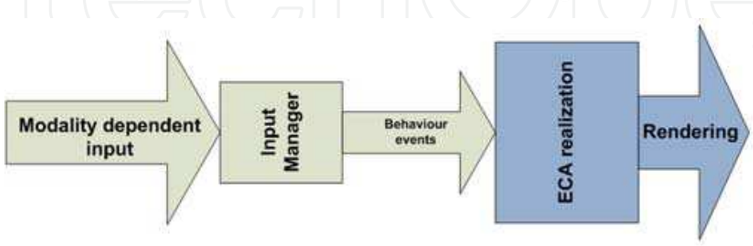

Fig. 6. General architecture of an ECA visualization system.

The idea of visualizing suggests that different input modalities are combined into different behavioural events. Different input modalities are commonly generated as abstract behaviour descriptions provided in XML based description schemes such as Affective Presentation Mark-

up Language (APML) (DeCarolis et al., 2004), and behavioural mark-up language (BML) (Vilhjalmsson et al., 2007). These are general description languages that can be used to describe any movement/action realized within the scope of human-machine dialogue. The task of an input manager, commonly referred to as a behaviour modeller, is to process different modality-dependent inputs, and to transform them into a time-referenced set of behavioural events. The key concern of such a time-scheduling process is to synchronize verbal with non-verbal behaviour, such as facial expressions, head movements, gaze and head gestures. Such behaviour often relies on the semantic information of data, such as non-standard sync-points at word breaks, dialogue markers, etc. The behavioural events (behaviour controllers) then form speech-synchronized descriptions of motion that should be transformed into movement on an ECA's articulated model (body). Different types of articulated bodies can be used (Güdükbay et al., 2008). In general 3D models can be grouped into:

- *Stick figure models*: models based on sets of rigid elements, and connected to joint chains.
- *Surface models* (mesh-based models): represent an upgrade of stick figure models. In this case, a polygonal mesh-layer (skin) is applied on the skeleton chains.
- *Volumetric models*: use simple volumetric primitives such as spheres, cylinders and ellipsoids, in order to construct the body shape.
- *Multilayered models* (muscle-based models): present anatomically-correct models. The animator of such models introduces different kinds of constraints to the relationship between layers.

The ECA realization engine (Figure 6) is used to store the articulated models of different ECA's (different bodies), and to apply behavioural events in the form of different transformations on the control units (parts of the articulated model used to generate movement). These transformations result in animated movement. The type of animation technique used depends on the type of articulated model. Most commonly, such animations are performed in the form of skeletal joint transformations and morphed-shaped transformations. The proprietary EVA framework (Mlakar & Rojc, 2011), developed to evoke a social response in human-machine interaction, is a python-based software environment that can convert a TTS system's output into audio-synchronized animated sequences of speech. ECA's provided by EVA framework can generate social responses in the form of facial expressions, gaze, head and hand movement and, most importantly, in the visual form of synthesized speech. The EVA framework provides a description script, an animation engine and articulated 3D models, and provides visual representation of synthesized speech sequences in the form of different types of video streams (in addition to synthesis into a video file/screen). Figure 7 outlines the modular architecture of the EVA framework.

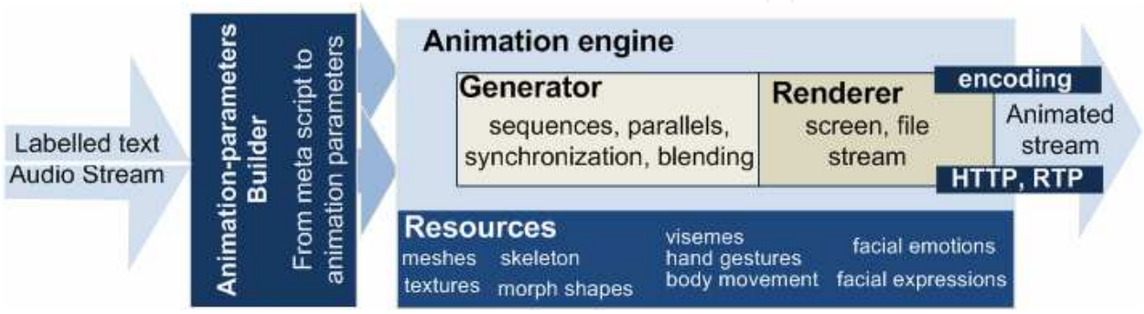

Fig. 7. Architecture of the EVA framework.

In order to personificate a TTS system by using the EVA framework, the TTS system has to produce TTS output according to the framework's specifications, based on the EVA Script XML scheme (Mlakar & Rojc, 2011). These XML schemes specify the desired ECA facial animations and body movements. The animation engine translates them onto the articulated 3D model of a human body in the form of animated movement. A TTS system's output can contain acoustic, linguistic, syntactic, semantic, and temporal information about general input texts that can be realized within a two stage visualization concept. The first stage is called 'Animation building' and the second 'Animation realization'. The Animation building stage transforms TTS output in the form of the EVA script XML scheme into animation parameters mapped to different control units of the ECA's 3D articulated model. The transformation from abstract to animatable content is then performed by the Animation parameter's builder. Such a transformation can be described as interfacing different XML tags using different ECA resources. Each ECA generated by the EVA framework has two types of resources. The 3D multi-part actor resources (Figure 7) contain different 3D-submodels of body (e.g. hair-style, eyes, teeth, dresses, etc.). Each 3D-submodel is associated with its corresponding textures, polygonal meshes and sets of different control units (morphed shapes, skeletal chains). The Personality template resources contain different behavioural templates written in the EVA script. These templates describe the common articulation of an ECA (e.g. how should, in general, specific viseme be formed), triggering words to gesture translations (e.g. what is a common gestural sequence when a certain word occurs), and other distinctive features of an ECA (e.g. eye-blinks, probability of gesturing, etc.) making each ECA an individual 'person'. The Animation parameters builder, therefore, translates the labelled text by interfacing each EVA script tag with a control unit, or behavioural template, and forms different groups of movements. Each group of movement is defined by semantic (which control units in which order), temporal (the duration of stroke, hold, and retraction phases) and spatial features (ending position of the control unit). The Animation realization phase transforms animation parameters into animated sequences. The animation parameters present raw data that describes how the Animation engine should move different control units. The Animation engine of the EVA framework takes care of animating and rendering the obtained animation parameter sets. It is based on the Panda 3D game engine (Goslin & Mine, 2004). In essence, this animation engine transforms the animation parameter sets into corresponding sequential and/or parallel movements of control-points (bones, or morphed shapes) lerp intervals. Each control-point in 3D space can be moved, either by 3D transitional or 3D rotational vectors (as specified in MPEG4 standard). The Forward kinematics and animation engine's generator provide procedures for the synchronization of such movements, and implement the animation-blending technique used on those animated segments that have to be controlled by different gestures at the same time (e.g. both smile and viseme can try to control the lower jaw joint; in such cases most of the influence is given to the viseme, and only a small portion is left to the facial gesture smile). Based on the semantics of the animation parameter sets, the animation groups the control units into sets of sequential and concurring movements and associates each movement with its temporal and spatial features, therefore forming personalized body movements. The EVA framework also presumes that no movement is linear and should, therefore, be interpolated against its interpolation curve. The EVA framework provides three types of non-linear interpolation for each personalized movement: EaseIn (slow-start and ramp-to-full, abrupt finish), EaseOut (starts with full speed and in the last n frames decelerates to a slow stop), and EaseInOut (starts slowly, ramps to full speed and after the

constant phase, if it exists, slowly decelerates to full stop). The rendering process is frame based and at each frame is interpolated against its non-linear interpolation curve. At any given frame the animation can also be stopped/paused or re-adjusted to its given temporal/spatial features. The EVA scripts describe both verbal and non-verbal behavior, independently. The verbal behaviour is contained within speech XML tags named speech, and the non-verbal behaviour may be contained within fgesture and bgesture tags describing facial expressions, and different body gestures. The verbal parameters can be described by the semantic, temporal, and articulation features of a sequence, whereas non-verbal behaviour involves describing the presence level of facial expressions and different body gestures. All speech-driven non-verbal behaviour can be defined in the TTS system's output directly, or indirectly by derivation of several non-verbal parameters found in the TTS generated output. Non-verbal parameters, such as emphasis, phrase/word breaks, and key phrases (e.g. dialogue discourse markers) are used when the non-verbal behaviour is controlled by a TTS system. The non-verbal feature allocators, fgesture and bgesture unify a set of control units, assigned to control different parts of the body. The facial expressions contain control units that can be physically assigned to the human face (e.g. control units such as lower-jaw, mouth corners, etc.). Similarly, body gestures allocate control units, such as: left elbow, neck, control units for fingers, etc. In addition, the left and right-eye control units are also assigned to the body gesture group. By describing the temporal and spatial features of movement in the form of sequential or parallel groups, the EVA framework enables hierarchical levels of animation for both fgesture and bgesture objects. Each movement can, therefore, be built from different control units with either sequential or concurrent movement. In the context of spoken dialogue systems using TTS systems, the EVA framework not only enables more realistic human-machine interaction, but can also evoke emotional and social responses that exist in face-to-face human-human spoken dialogues.

5. Multilingual and multimodal PLATTOS TTS system for the Slovenian language

This section presents an implementation of the corpus-based PLATTOS TTS system for the Slovenian language, using a concatenative approach and a TD-PSOLA speech-synthesis algorithm. The dequeues are tied together into a common time and space-efficient TTS engine, using the HRG structure for the representation of linguistic information. Finite-state machines, however, are used for efficient language resource representation, and separation of the language-dependent part from the language-independent TTS engine. The fsmHal library is used to efficiently construct the necessary finite-state machines used (Rojc, 2000; Rojc 2003). All modules, as specified in the TTS system architecture (Figure 1), are included and used. In the following subsections, implementation of those modules used for the personalized PLATTOS TTS system regarding the Slovenian language is presented in detail.

5.1 Tokenizer dequeue

All tokens are specified off-line in the form of regular expressions. Then the FSM compiler is used for the construction of a tokenizer finite-state machine. The additional part of the tokenizer module is the spell checker. It is used in order to prevent erroneous words that corrupt the performance of other modules within the TTS system, e.g. obtained prosody patterns result in speech signals with lower intelligibility. The spell-checking algorithm uses

a large word list, containing a set of valid words. Represented as FSA, the corresponding list is then used for edit distance calculations and searching for the best possible replacements for the misspelled words found in the input text. An additional part of the tokenizer is the normalisation process. Number tokens' factorization is performed firstly, in order to convert the numbers into the corresponding word forms. Some languages (e.g. German and Slovene) need additional filter (FST), for handling language-specific decade flop phenomenon. The core number lexicon is constructed from the SIlex lexicon (Rojc & Kačič, 2000), represented as FST. Additional rewrite rules are used for language-specific word insertions (special words such as ''and'' (English), ''und'' (German) or ''in/and'' (Slovene). Compiling rewrite rules into a FST is performed, since it is more efficient and requires a limited number of operations (Sproat, 1998). Furthermore, an important issue is the normalisation process of abbreviations, especially in the cases of inflectional languages. When considering the context a decision has to be made about which conversions are possible and which are impossible. The marking of unacceptable and acceptable conversions for a given context is done using the rewrite rules, and written by an expert. For processing special symbols (e.g. %), the construction of FSM representing lexical analysis for a given symbol, is performed for the conversion of a special symbol into word forms. In those cases where more possible conversions are preserved at the end, the most appropriate one is obtained using the BestPath algorithm (Rojc & Kačič, 2007).

5.2 POS-tagging dequeue

The POS tagging approach performed in the PLATTOS TTS system is based on the Brill's POS tagging approach. The POS tagging process consists of several steps. Firstly, the morphology lexicon is used as obtained from the training. Within this lexicon each entry is assigned the most probable POS tag found in the training corpus. If a word is not found, the SImlex morphology lexicon (Rojc & Kačič, 2000) is used next. Deterministic and minimized FST representation of SImlex lexicon represents time and space-efficient representation and fast lookup time. Morphological analysis then follows, which uses the so-called guessing automata, constructed for unknown words (this FSA tries to guess the POS tag by analysing word endings) (Daciuk, 1998). POS tagging context rules are used at the end. Within the scope of the post-processing stage, local grammars are used to resolve possible remaining ambiguities, e.g. as a consequence of systematic tagging errors that are unsolved during the POS-tagging process (Rojc & Kačič, 2007).

5.3 Grapheme-to-phoneme conversion dequeue

The SIflex phonetic lexicon for common words is first used during the unified approach to grapheme-to-phoneme conversion (Rojc & Kačič, 2007). Additionally, the SIplex phonetic lexicon for proper names is included, followed by the homograph detection step. Next, possible unknown words are converted into phonetic transcription by using trained CART tree models for stress, grapheme-to-phoneme, and syllable prediction. The HRG utterance structure is used as a linguistic knowledge source and for feature construction. Therefore, several complex features can be easily constructed by using a textual list of the linguistically attributed names. Syllable markers are also inserted into phonetic transcriptions (in the case of unknown words), since this information is important later for prosody modules. In the final stage of the unified G2P process, several rules have to be applied for performing the post-processing of the canonical phonetic transcriptions, by also considering cross-word

contexts. Namely, in the Slovenian language cross-word context has a significant impact on pronunciation and must be considered within the whole G2P conversion process. The expert defines these rules for all phoneme conversions, occurring at word beginnings and word endings. Furthermore, input texts often contain words or phrases from some other language. The first problem that has to be solved is to detect such words in efficient ways, and the second is to specify the corresponding pronunciations. When e.g. the Slovenian input sentence contains a German name "Gerhard Schröder'', these two words have to be detected, and then converted into the phonetic transcriptions using Slovenian phonemes. As suggested in (Rojc & Kačič, 2007), a G2P conversion module for the German language (using SIplex lexicon) is used first, and then German phonemes are mapped into the most suitable phoneme substitutions defined for the native language. This mapping can be done by using the phoneme mapping table constructed by the phonetic experts. This polyglot functionality is currently supported for German and English names.

5.4 Symbolic and acoustic prosody dequeues

Within symbolic prosody module the prediction of phrase breaks, prominence labels, and Tilt intonation labels (based on syllable level) is performed (Taylor, 2000). CART trees are used, since classification is performed on several discrete linguistic attributes during training. The phrase break prediction model inserts phrase break labels, the prominence prediction model marks the prominent syllables, and the intonation prediction model assigns Tilt intonation labels to each syllable. In the PLATTOS TTS system for the Slovenian language, a B3 label is used for labelling major phrase breaks, and a B2 label for minor phrase breaks. Additionally, phrase break positions are used for pause insertions in the sentence. Prominence labels on syllables are marked as PA (primary accent, assigned to the most accentuated syllables inside the intonation prosodic phrase), and as NA (marking secondary accents in the prosodic phrase). Tilt intonation event labels (a c l m fb rb afb arb lfb mrb mfb lrb) are assigned to each syllable in the sentence. In the acoustic prosody module prediction is performed for segment durations, pause durations at phrase break positions, and the prediction of Tilt acoustic parameters for each Tilt intonation event. Here, regression trees are used because of the nature of the used data. Separate prediction models are used for vowel phoneme duration prediction, and for the prediction of consonant phoneme durations. An additional tree model is trained for the prediction of pause durations, using only sentence internal pauses in the recorded Slovenian speech database (female voice). After the Tilt acoustic parameters have been predicted, reconstruction of the specified F0 contour can also be performed (for subsequent modules), and is stored within the HRG structure (Rojc & Kačič, 2007).

5.5 Unit selection dequeue

The input text corpus (newspapers, literature, internet) used for recording the Slovenian speech database consists of approximately 31 million words. The main criteria for selecting sentences were: richness with different diphone and triphone units, maximal final size of the speech database, and the minimal and maximal lengths of the sentences (Rojc, 2003). Before the segmentation process of the database into monophone, diphone and triphone units, the canonical phonetic transcriptions were manually verified, and adapted to the recorded material of the database. The initial acoustic inventory was constructed from a large set of non-uniform units (diphones and triphones). Then, the two-stage search space reduction

process was performed, as presented in section 3.2. During the first stage, all diphones and triphones were clustered by considering the phonetic contexts of the units (by using a tree-based clustering technique). The constructed trees were used in the next stage - in the process of eliminating acoustically-similar unit candidates (redundant units). Since calculation of concatenation costs is a very time consuming process, they are calculated off-line, as presented in 3.3, in order to achieve better time and space efficiency of the on-line unit selection process. A vector quantisation algorithm (VQ) was also used in order to minimize the huge CC cost matrix. The final VQ codebook and the FST with CC cost indices then enable an efficient CC cost lookup process in the on-line unit selection dequeue. As already mentioned, at the end of symbolic and prosodic dequeue, the sequence of target units with predicted symbolic and acoustic prosodic parameters is already defined. The next step is then to search in the optimised acoustic inventory for the best matching unit candidates. The basic strategy in the PLATTOS TTS system is to find the longest non-uniform unit for each target, ensuring also that the acoustic features and phonetic contexts of the unit candidates are as close as possible to the target unit specifications (stored as HRG utterance structure).

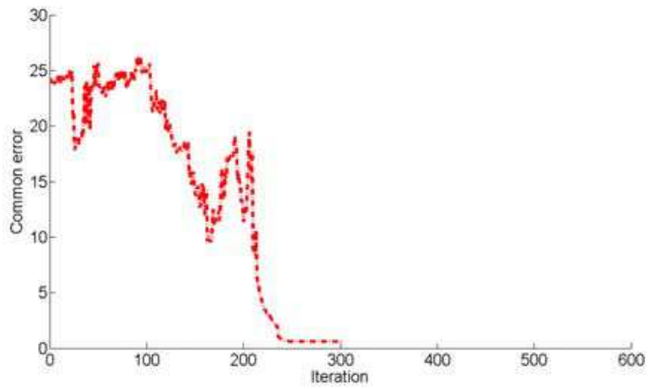

Fig. 8. Common error 'E' during the unit selection optimisation process.

As presented in 3.5, an important issue for an on-line unit selection process based on common unit cost calculation is the proper setup and tuning of partial suitability functions' parameters and weights, as used in equation (7). The tuning of these parameters and weights is performed from non-optimised acoustic inventory containing all database diphone and triphone units, together with the off-line calculated concatenation costs. The initial weights and parameters are defined by using the hybrid approach. The input into the unit selection optimisation process is a set of prosodically-annotated database utterances (100 sentences). When considering the defined prosody and the existing unit candidates in the acoustic inventory for each sentence unit, the optimisation process iteratively searches for a sequence of such units that would result in minimal F0 mismatches between selected candidates, and in minimal duration deviations towards the predicted prosody. After each iteration, the RGD algorithm evaluates the common error 'E' made by selecton of candidates (regarding pitch and duration), and updates parameters and weights before the next unit selection process occurs. Common error 'E' distribution across all iterations for the set of database utterances (female voice) is presented in Figure 8. The iteration having the smallest common error 'E' specifies the proper set of weights and parameters to be used in the on-line unit selection algorithm. As can be seen, the RGD algorithm does not stop if the

common error 'E' starts to increase, in order to avoid local minima - otherwise the unit cost functions' weights would have been already specified at visible minimum found after 30 iterations. The optimised acoustic inventory, tuned unit cost functions' parameters, and weights are then used in the on-line unit selection algorithm. The best sequence of unit candidates is found by using finite-state machine based BestPath algorithm that minimises the two costs along the input sentence: target cost and concatenation cost. The unit selection algorithm significantly reduces the amount of needed signal processing in order to meet the predicted prosody characteristics at the end, which naturally improves the quality of the generated speech. Figure 9 shows the common error 'E' per sentence (in the set of 100 sentences). It is composed of F0 mismatches between selected candidates, and of selected units' duration deviations towards unit target specifications, defined by both prosody modules. All errors are summed within each sentence and then divided by the number of selected units in the sentence. Therefore, the normalised common error values are actually presented, in order to compare the obtained values between sentences in the given test set. It can be seen that the common error 'E' across the whole set of sentences is significantly larger when running a non-optimised unit selection algorithm (x markers), and that the common errors 'E' in the case of the optimised unit selection algorithm are smaller (circle markers).

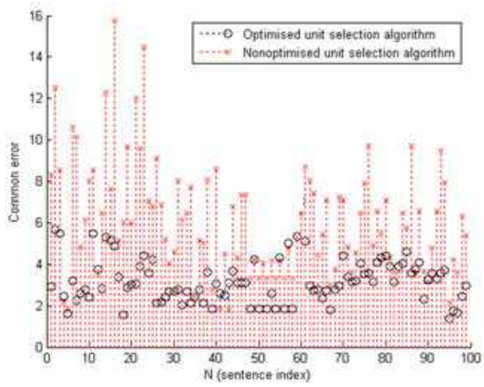

Fig. 9. Common errors 'E' on the set of test sentences (100 sentences).

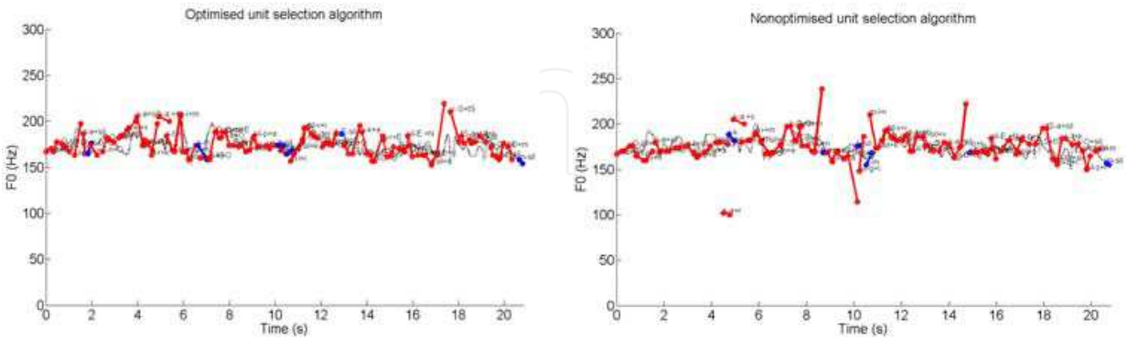

Fig. 10. Selected unit candidates when using (a) optimised cost functions' weights, and (b) non-optimised cost functions' weights.

Further, the sequences of the best candidates selected by using optimised and non-optimised unit functions' weights and parameters are shown in Figure 10. Here, each selected unit candidate (diphone/triphone) is represented by a straight line. The length

represents the duration of the selected unit candidate. The start and end-points of each line are characterized by the F0 values at the start and end of each unit candidate selected from the acoustic inventory. Naturally, the goal of the unit selection process is that observed F0 differences between successive straight lines are as small as possible, and that pitch values between candidates are also as close as possible. Namely, this will result in lesser-needed signal post-processing to be performed by TD-PSOLA. When comparing figures (a) and (b), it can be seen that by using an optimised unit cost functions' weights and parameters, the F0 mismatches are smaller (and F0 points between successive units are closer), which results in a more fluent and more natural synthesized speech signal.

5.6 Concatenation and acoustic dequeue

The concatenation module processes those units selected by the unit selection process in the previous dequeue. Since the following acoustic module is based on the TD-PSOLA algorithm, this module takes care of the following processing steps: calculation of analysis pitches, searching for an optimal concatenation point between two successive units, matching of analysis and synthesis pitches, and the smoothing of concatenation points. The acoustic module based on the TD-PSOLA algorithm is then used for changing the durations and pitch on those selected units, where existing F0 mismatches and duration deviations are unacceptable (Rojc & Kačič, 2007).

5.7 PLATTOS ECA – EVA –

ECA EVA (Mlakar & Rojc, 2011) is a PLATTOS TTS system's conversational agent that can be used in different spoken dialog systems. ECA EVA represents the personification of the PLATTOS TTS system, implemented by using the EVA framework. The PLATTOS TTS system output's synthesised speech and linguistic and acoustic data of the input sentence (contained in the HRG structure) in the format of the EVA XML script. Currently, these scripts contain sequences of phonemes, visemes, and gesture triggers. Each generated EVA script includes corresponding temporal (duration), and spatial information (e. g. articulation). By using this input, the EVA framework is then able to visualize a PLATTOS TTS system's output in the form of animated verbal and rule-based non-verbal behavioural response. ECA EVA is a female agent, since the selected ECA gender depends on of the voice used in the TTS system. It can synthesize expressive speech sequences based on different levels of co-articulation, head and hand gestures, facial expressions and emotions, and gaze. The lip-sync process synthesizes verbal features, and employs the articulation parameter (stress) at spoken sequence and utterance levels. At the spoken sequence level of articulation all utterances are additionally modified to meet the general articulation properties of the sequence as a whole. Articulation at the utterance level only modifies the spatial properties of the selected utterance. In this way, spoken dialogue-flow can not only adapt articulation, but also influence the speed at which a certain answer is spoken. Therefore, in addition to articulation relating verbal features, the general articulation can also define several personality features of an ECA (e.g. fast-speaker, speaker with good articulation etc.). If, for instance, the user did not understand the different parts of the spoken sequences, such sequences can be repeated at a slower rate and with a higher level of articulation. Therefore, such a personificated PLATTOS TTS system can also be used as a tool for learning pronunciations and other learning/entertainment applications. The rule-based non-verbal behaviour, such as: emotion, facial expressions, head and hand

movements, are generated based on linguistic and acoustic information (stored in the HRG structures) that the PLATTOS TTS system can currently provide (e.g. morphology information, phrase-break labels, prominence labels, trigger words and phrases, stress levels, pitch etc.). By using emphasis markers word/phrase-break markers, ECA EVA can generate different speech-driven pointing gestures that can visually emphasize a certain word/phrase. By interfacing words/phrases with different emotions, and facial expressions, EVA can visually generate speech-driven facial expression, such as: speaking with a gentle smile, saying something sadly, etc. All these features represent an essential part of the visual synthesis from TTS output. Figure 11 demonstrates the personification of the PLATTOS TTS system including expressiveness and emotions that are already well-supported by the EVA framework. Different speech segments can be accompanied by different facial gestures, e.g. emphasis can be defined by a higher level of articulation, slightly lower pronunciation rate, and by raising eyebrows. Negation can be further emphasized by repetitive nods.

Fig. 11. Personification of the PLATTOS TTS system output.

The gestures used on the right-hand side (Expressive behaviour) of Figure 11 are independent and don't directly influence each other. The animation blending technique enables the deployment of facial expressions, emotions, and speech, simultaneously. The bone-based ECA automatically removes most of the "jerky", or unnatural poses that usually result when animating expressive ECAs, such as: the eyes don't follow whilst the head is turning etc. Since the multipart concept uses a shared skeleton, even though the eyes and head are of different body types, the eyes will automatically be sub-parented to the joint chain of the head (to the one among the joints in the head joint-chain). This will result in the eyes following the head's movements. Therefore, when the eyes move, head will be uninfluenced, but when the head moves, eyes will move according to an automatic gaze generation process. Furthermore, gestures, emotions, gaze and verbal communication (lip-sync), can vary in composition (which combinations of control points are used to form them), in amplitude (to what extent a gesture forms; e.g. co-articulation of utterances), in speed, and in repetitiveness. The expressive behaviour presented was generated by specifying each of the gesture types in the form of the EVA Script, provided by the PLATTOS TTS system. ECAs generated by the EVA framework and PLATTOS TTS system can generate different speech-driven types of gestures, gaze, and both simple and complex

emotions, in an expressive, fully adjustable way. All the ECA's body movements within are defined and described hierarchically, as a composition of movements of the control units. The PLATTOS ECA-EVA enables the animation of rich sets of gestures, expressions, or event speech utterances that can vary in time, space and composition.

6. Evaluating multilingual and multimodal TTS systems

Constant evaluation as a constituent part of research activities has proven to be a successful approach for enhancing progress in almost all areas of speech technology, such as speech recognition, speech synthesis, or speech translation, especially if organized in the form of evaluation campaigns, e.g. TC-STAR[1], Blizzard[2] etc. (Rojc et al., 2009). As we know, the traditional evaluations are not performed 'on-line', the transport of test data and results has to be treated manually, and the test data are not 'secret'. Furthermore, the connecting of different developers' modules cannot be handled without an exchange of software to be integrated locally. In order to solve all these issues of traditional evaluations for testing TTS systems, a RES (remote evaluation system) evaluation framework has been established over recent years for speech synthesis technology within the ECESS consortium[3].

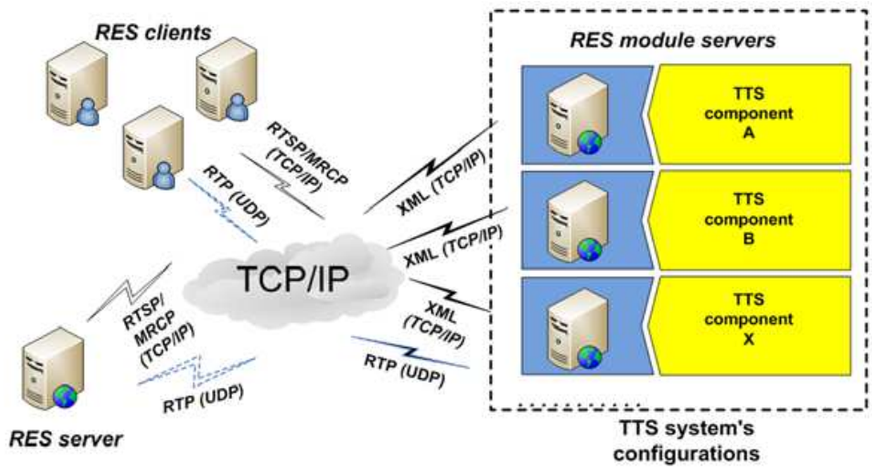

Fig. 12. RES framework for developing and evaluating multilingual and multimodal TTS systems and components.

The key element of the RES is its specification for a set of separate modules: e.g. for text processing, prosody generation, acoustic synthesis modules etc., that can be combined together into a complete text-to-speech system. Being able to split into any number of such modules has the advantage that the developers of an institution can concentrate its efforts on a single module, and test its performance within a complete system, using missing modules from the developers of other institutions etc. In this way high-performance multilingual and multimodal TTS systems can be built by using the high-performance modules of different institutions. A common evaluation methodology has been developed

[1] www.tc-star.org (EU project TC-STAR)
[2] http://festvox.org/blizzard/ (The Blizzard challenge)
[3] www.ecess.eu (ECESS - European Center of Excellence in Speech Synthesis)

to assess the performances of the modules that are based on the common use of those module-specific evaluation criteria and module-specific language resources needed for training and testing the modules. The RES was designed, not only to evaluate TTS modules, but also to support the developers of TTS modules. Developers/researchers can use RES in a test/development modus, in order to improve the performances of their TTS module(s), and evaluators can use RES in an evaluation modus for measuring the performances of the selected TTS modules. The distributed architecture of the RES is shown in Figure 12. As can be seen, the system consists of several RES clients (for developers, researchers, and evaluators), the RES server (managing unit), and RES module servers encapsulating the TTS modules (developers, and researchers). The RES server communicates simultaneously with several RES clients, and also supports the RES module servers when communicating with several RES clients at the same time. When performing testing or evaluating, developers, researchers and evaluators only select the desired TTS modules via RES clients and provide corresponding input for the selected task. The given input is then automatically transferred within the RES to the selected TTS modules, and generated output is returned to the RES client.

7. Conclusion

The presented design pattern for multilingual and multimodal corpus-based TTS systems shows that it is possible to integrate all modules of the TTS system, from text processing to acoustic processing, into an efficient and flexible queuing mechanism. Time and space-efficient FSMs are used for separating language dependent resources from a language-independent TTS engine, for the time and space-efficient representation of language resources, and for fast information lookup. A HRG structure is used for storing complex and heterogeneous sentence information, and for flexible construction of complex features. Furthermore, optimisation of the unit-selection process is one of the most important issues for corpus-based TTS systems, where several processing steps have important impacts on the achieved performance of the TTS system, regarding quality and efficiency. A RGD algorithm for cost functions' weights optimisation is proposed within the unit-selection process. Objective and subjective measures show that such optimisation results in a better quality of generated speech, a smaller common error 'E' regarding unit duration deviations and pitch disagreements between selected speech segments, is fully automatic and language-independent. It is, therefore, very helpful for tuning a general unit selection process, and can speed-up the generation of new voices for corpus-based TTS systems. The presented design pattern was demonstrated on the implementation of the Slovenian corpus-based PLATTOS TTS system; however, it can be used for the construction of TTS systems for other languages, for which the necessary language resources exist. By personification of the PLATTOS TTS system using ECA EVA, it can be used in advanced multimodal spoken dialogue systems. PLATTOS TTS system and EVA framework together provide flexible and efficient audio-visual multimodal output, enriched with a rich set of gestures, expressions, and emotions. Namely, by using EVA Script schemes, synthesized speech can be enhanced with several body movements, several types of visually represented articulation, different facial expressions (e.g. eye-lid movement, gaze, smile, emotions, etc.), and different body gestures (hand gestures, head

movement, etc.). All these features personificate machine generated responses, and provide means for more natural human-machine interaction to be used in multimodal spoken dialogue systems. The ability, not only to articulate but also to control the speed and level of articulation, additionally enhances human-machine interfaces.

8. Acknowledgment

Operation part financed by the European Union, European Social Fund.

9. References

Ball, G. & Breese, J. (2000). Embodied conversational agents, Emotion and personality in a conversational agent, 2000

Black, A. W. & Taylor, P. (1997). Automatically clustering similar units for units selection in speech synthesis. Proceedings of Eurospeech 2, 601– 1368

Brill, E. (1993). A Corpus-Based Approach to Language Learning. Ph.D. Thesis

Campbell, N. & Black, A. (1996). CHATR: a multi-lingual speech resequencing synthesis system. Institute of Electronic, Information and Communication Engineers, Spring Meeting, Tokyo SP-96-07

Cassell, J. (2000). Nudge Nudge Wink Wink: Elements of Face-to-Face Conversation for Embodied Conversational Agents, in Cassell, J. et al. (eds.), *Embodied Conversational Agents,* pp. 1-27. Cambridge, MA: MIT Press

Daciuk, J. (1998). Incremental Construction of Finite-State Automata and Transducers and their Use in the Natural Language Processing, Ph.D. thesis, Technical University of Gdansk, Poland

DeCarolis B.; Pelachaud C.; Poggi I. & Steedman M. (2004). APML, a mark-up language for believable behavior generation. *In H. Prendinger and M. Ishizuka, editors, Life-like Characters. Tools, Affective Functions and Applications*, pp. 65-85, Springer-Verlag Berlin Heidelberg

Emmanuel, R. & Schabes, Y. (1995). Deterministic Part-Of-Speech Tagging with Finite-State Transducers. Mitsubishi Electric Research Laboratories

Georgantas, G.; Issarny, V. & Cerisara, C. (2006). Ambient Intelligence, Wireless Networking, and Ubiquitous Computing, chapter Dynamic Synthesis of Natural Human-Machine Interfaces in Ambient Intelligence Environments. Artech House, July 2006

Goslin, M. & Mine, M. R. (2004). The Panda3D Graphics Engine, Computer, v.37, n.10, p.112-114, October 2004

Gratch, J.; Rickel, J.; Andre, E.; Badler, N.; Cassell, J. & Petajan, E. (2002). Creating Interactive Virtual Humans: Some Assembly Required, *IEEE Intelligent Systems* 17(4): 54-6

Güdükbay, U.; Özgüç, B.; Memişoglu A. & Yeşil M.Š. (2008). Modeling, Animation, and Rendering of Human Figures. Signals and Communication Technology, 2008, pages 201-238, Springer

Heloir & Kipp, M. (2009). EMBR—a realtime animation engine for interactive embodied agents. *In Proceedings of the 9th International Conference on Intelligent Virtual Agents (IVA '09)*, pp. 393–404, Springer, Amsterdam, The Netherlands, 2009

Holzapfel, M. & Campbell, N. (1998). A nonlinear unit selection strategy for concatenative speech synthesis based on syllable level features, In ICSLP-1998, paper 0521, Sydney, Australia

Holzapfel, M. (2000). Konkatenative Sprachsynthese mit grossen Datenbanken. Ph.D. Thesis.

Mlakar, I. & Rojc, M. (2011). EVA: expressive multipart virtual agent performing gestures and emotions. International journal of mathematics and computers in simulation, 2011, vol. 5, iss. 1, pp. 36-44

Mohri, M. (1995). On Some Applications of Finite-State Automata Theory to Natural Language Processing. Natural Language Engineering, 1

Mohri, M.; Fernando, C. N. Pereira & Riley, M. (1996). Weighted automata in text and speech processing. In: ECAI-96 Workshop, Budapest, Hungary. ECAI

Mohri, M. & Sproat, R. (1996). An efficient compiler for weighted rewrite rules. In: 34-th Meeting of the Association for Computational Linguistics (ACL 96), Santa Cruz, California

Mohri, M. (1997). Finite-state transducers in language and speech processing. Computational Linguistics 23 (2), 269–311

Poggi I. & Pelachaud C. (2000). Performative facial expressions in Animated faces, Conversational Agents, Emotion and personality in a conversational agent, In Book Embodied conversational agents, MA, USA, 2000

Raitio, T.; Suni, A.; Yamagishi, J.; Pulakka, H.; Nurminen, J.; Vainio, M. & Alku, P. (2011). HMM-based speech synthesis utilizing glottal inverse filtering. IEEE Transactions on Audio, Speech and Language Processing, 19(1):153-165, January 2011

Rojc, M. & Kačič, Z. (2000). A computational platform for development of morphologic and phonetic lexica. In Proceedings of the Second Language Resources and Evaluation Conference (LREC), Athens, Greece

Rojc, M. (2000). The Use of Finite-state Machines for Text-to-Speech Systems. Master thesis

Rojc, M. (2003). Time and Space Efficient Architecture of the Multilingual and Polyglot Text-to-Speech System – Architecture with Finite-State Machines. Ph.D. Thesis

Rojc, M. & Kačič, Z. (2007). Time and Space-Efficient Architecture for a Corpus-based Text-to-Speech Synthesis System, Speech Communication, Vol. 49 (3), 2007, pp. 230-249

Rojc, M. & Kačič, Z. (June 2007). A Unified Approach to Grapheme-to-Phoneme Conversion for the PLATTOS Slovenian Text-to-Speech System, Applied Artificial Intelligence, Vol. 21 (6), June, 2007, pp. 563-603

Sproat, R.; Riley, M. (1996). Compilation of weighted finite-state transducers from decision trees. In: Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics, pp. 215–222

Sproat, R. (1998). Multilingual Text-to-Speech Synthesis. Kluwer Academic Publishers, ISBN 0-7923-8027-4, Massachusetts, USA

Taylor, P.; Black, A. & Caley, R. (1998). The architecture of the Festival speech synthesis system. In: Proceedings of the Third ESCA Workshop in Speech Synthesis, pp. 147–151

Taylor, P.; Black, A. & Caley, R. (2001). Heterogeneous relation graphs as a formalism for representing linguistic information. Speech Communication 33 (1-2), 153–174

Vilhjalmsson, H.; Cantelmo, N.; Cassell, J.; Chafai, N.; Kipp, M.; Kopp, S.; Mancini, M.; Marsella, S.; Marshall A.; Pelachaud, C.; Ruttkay Z.; Thorisson, K.; van Welbergen, H. & van der Werf, R. (2007). The Behavior Markup Language: Recent Developments and Challenges. IVA 2007, LNAI 4722: 99-111, Springer-Verlag Berlin Heidelberg

Speech and Language Technologies
Edited by Prof. Ivo Ipsic

ISBN 978-953-307-322-4
Hard cover, 344 pages
Publisher InTech
Published online 21, June, 2011
Published in print edition June, 2011

This book addresses state-of-the-art systems and achievements in various topics in the research field of speech and language technologies. Book chapters are organized in different sections covering diverse problems, which have to be solved in speech recognition and language understanding systems. In the first section machine translation systems based on large parallel corpora using rule-based and statistical-based translation methods are presented. The third chapter presents work on real time two way speech-to-speech translation systems. In the second section two papers explore the use of speech technologies in language learning. The third section presents a work on language modeling used for speech recognition. The chapters in section Text-to-speech systems and emotional speech describe corpus-based speech synthesis and highlight the importance of speech prosody in speech recognition. In the fifth section the problem of speaker diarization is addressed. The last section presents various topics in speech technology applications like audio-visual speech recognition and lip reading systems.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen