Tuning Limited Domain Speech Synthesis Using General TTS System*

Markéta Jůzová and Daniel Tihelka

University of West Bohemia, Univerzitní 8, Plzeň, Czech Republic
juzova@kky.zcu.cz, dtihelka@kky.zcu.cz
http://www.kky.zcu.cz

Abstract. The subject of the present paper is the building of a limited domain speech synthesis system, where longer units, like words and phrases, can naturally be concatenated together. However, instead of building a single-purpose domain-oriented engine working with longer units, we show that a general-purpose TTS system can be used as a good emulation tool to ensure that a real domain-oriented engine will work correctly. Since the current general speech synthesis system embedding unit selection method concatenates short speech units (diphones), the selection algorithm has been modified to pretend the concatenation of words or even the whole phrases, while still concatenating diphones internally. The behaviour of the system is tested on two limited domains and its output is compared to the output of general (unmodified) version of the same TTS system. The results show clear encouragement for the build of the "real" domain-oriented engine.

Keywords: limited domain, concatenative speech synthesis, speech units, units concatenation, target cost, concatenation cost

1 Introduction

Limited domain speech synthesis (LDTS, [1,2,3,4]), i.e. the domain-oriented synthesis embedding unit selection method [5,6], attracts with the advantage of using longer units for concatenations instead of short units, e.g. diphones. The usage of longer units brings a lower number of concatenations in the synthesized sentence, which positively affects the probability of the speech artefacts occurrence, as well as it lowers the computational complexity due to the much lower number of candidates of speech units to search optimal path through. While it may seem for the first time that the build of LDTS system is fairly easy task, the opposite is true – the concatenation of longer units, being natural-sounding by their nature, can cause a phenomenon called "Uncanny Valley", describing the level of aversion to unnatural artefacts in otherwise natural surrounding.

To ensure the high-quality LDTS, the limited domain (LD) text corpus has to be prepared carefully. It is recommended that the corpus should cover well the given

* This work was supported by the European Regional Development Fund (ERDF), project "New Technologies for Information Society" (NTIS), European Centre of Excellence, CZ.1.05/1.1.00/02.0090, the Technology Agency of the Czech Republic, project No. TA01030476 and SGS-2013-032.

P. Sojka et al. (Eds.): TSD 2014, LNAI 8655, pp. 408–415, 2014.

domain [1,8], i.e. to contain the most (or ideally all) common words and phrases in contexts. The presence of contexts brings the advantage of finding the optimal concatenation point in the overlap, which is much better compared to concatenations in pauses between words. The context enlarges the corpus a bit, but it is still much smaller (and cheaper) than the corpus for a general-purpose TTS system. The general text corpus should, on the contrary, meet the requirement of a good coverage of diphones in different phonetic and prosodic contexts [7], since they are required when synthesizing a text unknown beforehand.

1.1 Our Goal

Based on the assumption about the ability of LD system to generate higher–quality speech, we aimed to create such a system (and verify if it is truth) and use it for applications, in which it would fit better than a general synthesizer. Since we have only specialized LD text corpus at disposal, but not any real LDTS system, we decided to use our general TTS system ARTIC [6] working on diphones, and twist it in such a way that it works exactly as LDTS system should work. That means, the emulation tool created in this way and presented in this paper is a general TTS system containing several modifications to ensure the pretence of the concatenation of longer units. The outputs of the designed system are compared with the outputs of the general (not modified) system to find out, if the LD synthesis achieves higher level of naturalness and thus if LDTS system is worth building and using instead of the general one.

1.2 Concatenative Method Analysis

In both versions of speech synthesizer, general and LDTS emulator, the optimal sequence of speech unit candidates is searched for the given input text. Generally, after decomposition of the text into speech units, candidates of these units stored in the speech units database are used to build the graph being evaluated by *target cost* (nodes) and *concatenation cost* (edges):

- *Target cost* $C^t(t_i, u_i)$ quantifies the difference between the speech unit candidate u_i and the requirements for the unit t_i; this regards the phonetic context, sentence type, position features etc. [10]; $C^t = 0$ for candidates originating in the same surroundings as required to be placed into.
- *Concatenation cost* $C^c(u_{i-1}, u_i)$ indicates how much the join of candidates would fit together; $C^c = 0$ for candidates neighbouring in the source corpus.

The *cumulation cost* is defined as

$$C(t_1^N, u_1^N) = \sum_{i=1}^{N} C^t(t_i, u_i) + C^c(\#, u_1) + \sum_{i=1}^{N} C^c(u_{i-1}, u_i) + C^c(u_N, \#)^1 \qquad (1)$$

Having evaluated the graph, the Viterbi algorithm [11] is used for the optimal path search, choosing the path with the minimum cumulation cost.

[1] # = pause

2 Phrase Segments Searching and Dividing into Chunks

The process of text-to-units decomposition differs significantly between the two versions of the synthesizer. In the LDTS case, we used the approach described in [4], where the longest phrase segments were searched for. For the given input sentence, this method first creates a set of segments by detracting words from the end and from the beginning of the sentence, as shown below:

Dnes bude zataženo a zima.	*Today it will be cloudy and cold.*
Dnes bude zataženo a ...	*Today it will be cloudy and ...*
... bude zataženo a zima.	*... it will be cloudy and cold*
Dnes bude zataženo ...	*Today it will be cloudy ...*
... bude zataženo a ...	*... it will be cloudy and ...*
... zataženo a zima.	*... cloudy and cold.*
Dnes bude ...	*Today it will be ...*
... bude zataženo ...	*... it will be cloudy ...*
... zataženo a ...	*... cloudy and ...*
a zima.	*... and cold.*
Dnes ...	*Today ...*
... bude ...	*... it will be ...*
... zataženo ...	*... cloudy ...*
... a ...	*... and ...*
... zima.	*... cold.*

It is evident from the example that the term "phrase segment" refers to any word sequence (sometimes even containing only one word).

After decomposing the given sentence into segments, the longest segments contained in the LD speech corpus must be found. Let's say, the corpus contains only one sentence and one phrase:

Dnes bude slunečno a větrno.	*Today it will be sunny and windy.*
... bude zataženo a ...	*... it will be cloudy and ...*

For the given sentence

Dnes bude zataženo a zima.	*Today it will be cloudy and cold.*

the following longest segments are found in the corpus:

Dnes bude ...	*Today it will be ...*
... bude zataženo a ...	*... it will be cloudy and ...*

The fact that the phrases overlap with the word "bude" is intentional (the corpus has been designed to ensure this, as described in [8]), and the advantage of the overlap is discussed further in Section 3. Unfortunately, they do not cover the whole given sentence – there is the last word

... zima.	*... cold.*

not contained in the corpus. The reason for this may be that this word does not belong to the domain, or due to a huge size of the domain the word is missing in the corpus. Note that this is a model example, in which we intentionally do not include the word "zima" in the corpus to show the behaviour of the system in such a case. In reality, we try to

prevent this situation as far as possible when building the specialized text corpus by the algorithm introduced in [8].

Before the next step we transform the segments into three types of chunks:

- type 0 – chunks appearing in found phrase segments only once
  `Dnes ...` *Today ...*
  `... zataženo a ...` *...cloudy and ...*
- type 1 – chunks representing the overlapping regions of segments, in which the optimal concatenation point is searched, see Section 3
  `... bude ...` *... it will be ...*
- type 2 – chunks corresponding to the part not found in the corpus
  `... zima.` *... cold.*

3 Synthesis of Individual Sentence's Chunks

The "real" LD synthesizer would just take type 0 chunks, find the optimal concatenation point to concatenate them together and, if needed, the general TTS system will be used for the synthesis of chunks of type 2 using short speech units.

Since we use the general TTS system as the underlying platform, it decomposes the synthesized text into diphones[2]. To ensure that the general system emulates the LDTS way of work, we modified the unit selection algorithm [11] to pre–prune the candidates graph according to the type of synthesized chunk.

Type 0 The real LDTS system would take the chunk of type 0 as an atomic unit, and will as a whole concatenate it with the following chunk of type 1 or 2. Therefore, we have to ensure that only unit candidates from the phrase representing the chunk will remain in the selection algorithm, as drawn in Figure 1. Although we cannot avoid the evaluation of the costs, they are both $C^t(t_i, u_i) = C^c(u_{i-1}, u_i) = 0, \forall i$ and there is only one possible path through the graph.

Type 1 In the case of overlapping segments, the real LDTS system must choose the optimal concatenation point within the segment. To achieve this, the emulator prunes out all unit candidates except those from the two overlapping segments, as illustrated in Figure 2. During the costs evaluation, $C^t(t_i, u_i) = 0, \forall i$ due to the same reasons as for type 0, and $C^c(u_{i-1}, u_i) = 0$ for all i except the point in segments transition.

Type 2 The chunks marked as type 2 are not contained in the LD corpus at all. Therefore, all candidates for the given units can be used to find the optimal sequence in the same way as it works in the general-purpose TTS; see Figure 3 for the illustration. Both costs are important (getting generally non-zero values), because there are many candidates available for every unit in the chunk.

[2] Let's note that all examples are demonstrated on letters for their better understandability; the fact that the real TTS works with diphones does not affect the described algorithms in any way, one must just put diphone in place of letter in the examples.

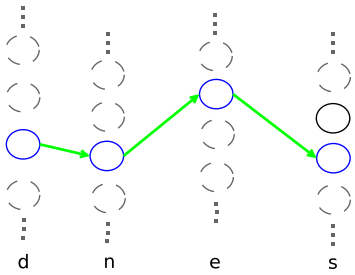

Fig. 1. Graph structure for units in the chunk of type 0 (*dnes* = *today*). The dashed–lined circles represent the unit candidates originated from different places in the LD corpus, which must not be considered. The arrows represent the only possible path.

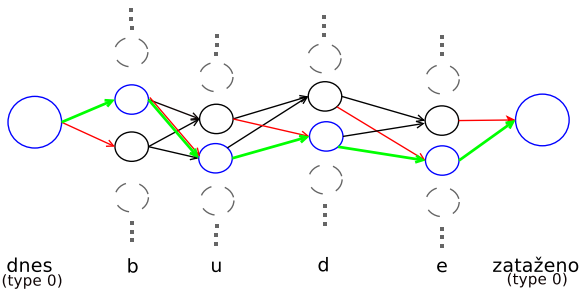

Fig. 2. Graph structure for a chunk of type 1 (*dnes* = *today*, *bude* = *it will be*, *zataženo* = *cloudy*). The edges connecting the nodes from the first segment to the nodes from the second segment represent the possible concatenation points in the chunk, the highlighted ones represent some of the optimal paths through the graph.

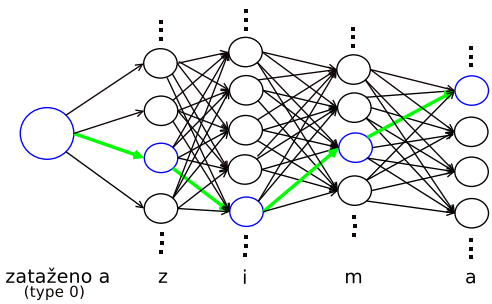

Fig. 3. Graph structure for a chunk of type 2 (*zataženo a* = *cloudy and*, *zima* = *cold*). The highlighted edges represent an example of the optimal path in the graph.

4 Evaluation

Although the LDTS emulator builds much less extensive graph of candidates for Viterbi searching, which significantly reduces the computational complexity of the synthesizer, there are still unnecessary cost evaluations which will not occur with larger units (all the zero costs in the chunks of type 0 and 1). Nevertheless, rough estimation of performance gain has been concluded in Section 4.1.

However, the reduction of computational cost is only one of the reasons why to prefer domain-oriented system to that general–purpose. The main motivation was to verify whenever speech generated from much smaller speech corpus using LDTS system can outperform the naturalness of speech generated from large speech corpus [7] by our well tuned general TTS system; see Section 4.2.

4.1 The Evaluating of the Computational Complexity

In the case of general synthesis, there are tens or hundreds of candidates for every diphone in the corpus [11], leading to huge amount of $C^t(t_i, u_i)$ and $C^c(u_{i-1}, u_i)$ evaluations. It is illustrated in Table 1 that there is only a fraction of evaluations for the same synthesized phrase in the emulator, even when the unnecessary evaluations are considered as well. On the other hand, the task of text–to-units decomposition is more complex in case of LDTS system (and not very optimized in this experimental version). Therefore, the performance gain of LDTS version is bit lower than it would be estimated from the ratio of costs evaluations, but still an order of magnitude above the general–purpose TTS system. This is summarized in t_1/t_2 column in Table 1.

Table 1. The evaluating of the computational complexity in the number of costs evaluations. t_1/t_2 is the ratio of times required to synthesize the same set of LD texts.

		LDTS emulator		**general TTS**		
number	chunks' types	C^t	C^c	C^t	C^c	t_1/t_2
1	only type 0	38	36	35 824	18 538 254	1/96
2	only types 0 and 1	62	99	41 612	20 230 985	1/35
3	one short chunk of type 2	505	767	44 393	30 397 914	1/23
4	types 0, 1, 2	8 683	1 484 595	42 508	30 613 890	1/16

4.2 The Evaluating of the Quality of Synthesized Sentences

To obtain the comparison of the quality of LDTS-generated speech with the quality achieved by the general TTS system, we carried out listening tests on two limited domains for which we have speech corpus recorded. The first one is the telephone automaton informing about results of exams on our university, the second is the automatic system informing about departures and arrivals of trains and ticket prices [9]. All the generated prompts were evaluated by 57 listeners.

The prompts for the listening tests were divided into 2 groups, which were first evaluated separately, then together:

- 1^{st} group: prompts consisting only of chunks with types 0 and 1
- 2^{nd} group: prompts with at least one type 2 chunk, i.e. with at least one word out of the domain, not being recorded in the speech corpus

The same number of prompts in both groups were purposefully chosen to represent, in authors' view, "the worst case" — in reality, there would be more phrases from the 1^{st} group during the usual LDTS system operation, supposing that the domain is covered well with the specialized corpus [8].

For the evaluating, we used *CCR tests* with a simple 3-point scale:

- 1 (LD) - the output of the emulated LDTS system sounds better,
- 0 (S) - the outputs are about the same (sound either good or bad),
- -1 (G) - the output of general TTS system sounds better.

The quality is then defined as

$$q = \frac{1 \cdot LD \; + \; (-1) \cdot G}{LD + S + G} \; , \tag{2}$$

where LD, S, G are the numbers of listeners' answers in the given category. The positive value of q means that the output of the LD synthesizer sounds better.

To ensure the validity of results, we also carried out the *sign test* with the null hypothesis "H_0: *the outputs of the both synthesizers are of the same quality.*" compared against the alternative hypothesis "H_1: *the output of one synthesizer sounds better,*" the p-values were determined for the significance level $\alpha = 0.05$.

All the results are shown in the Table 2. It is evident, that the quality of LDTS is evaluated higher than the quality of the general synthesis.

Table 2. The evaluating of the quality. The numbers of listeners' answers, the value q counted using the (2) and the $p - value$ of *sign test*.

	1^{st} group	2^{nd} group	all sentences
LD system is better (LD)	280	159	439
general TTS is better (G)	105	184	289
systems are of the same quality (S)	71	56	127
q	0.384	-0.063	0.175
p-value for $\alpha = 0.05$	< 0.0001	0.0975	< 0.0001
conclusion	LD system better	the same quality	LD system better

5 Conclusion

Summing up the results, it was confirmed that the LDTS system achieves better results, both in terms of quality and performance. Although this is rather expected result, most importantly it confirms that the development of LDTS system still makes a lot of sense for applications with requirements for the highest possible quality of generated speech.

On the other hand, it must be mentioned that the outputs of a domain-oriented LDTS system can be better only if the domain is covered well with the speech corpus used together with the LDTS system. The results show that if the synthesized text contains unknown word, the quality is, naturally, lowered to the level achieved with the general–purpose TTS system. It is the fact that the 100% coverage of given domain can not usually be met due to constraints put on the corpus size, recording time or development budget. There still may be chunks not coverable by the corpus at all (e.g. surnames, street names, etc.), or there may appear new texts required to be synthesized which had originally not been considered and recorded. Therefore, the corpus preparation must be taken very carefully, and it generally wise to extend the LD corpus with an appropriate amount of out-of-domain texts to ensure the coverage of rare (as related to the domain, not rare in general) units which will have to be used for type 2 chunks.

References

1. A.W. Black. Perfect Synthesis for all of the people all of the time. In: *IEEE Workshop on Speech Synthesis, Santa Monica, USA, 2002.*.
2. A. W. Black and K. A. Lenzo. Limited Domain Synthesis. In: *ICSLP2000, 2000*.
3. J. Yi and J. Glass. Natural-sounding speech synthesis using variable-length units. In: *Proceedings of ICSLP, Sydney, Australia, 1998*, pp. 1167–1170.
4. R. E. Donovan, M. Franz, J. S. Sorensen, and S. Roukos. Phrase splicing and variable substitution using the ibm trainable speech synthesis system. In: *Proc. of the Acoustics, Speech, and Signal Processing, 1999.*, pp. 373–376, Washington, DC, USA, 1999. IEEE Computer Society.
5. J. Matoušek, J. Romportl, D. Tihelka and Z. Tychtl. Recent Improvements on ARTIC: Czech text-to-speech system. *INTERSPEECH 2004 – ICSLP, Proc. of the 8th Int. Conf. on Spoken Language Processing*, pp. 1933–1936, Korea, 2004.
6. J. Matoušek, D. Tihelka and J. Ramportl Current state of Czech text-to-speech system ARTIC. In: P. Sojka et al. (Eds.) *Proc. of Text, Speech and Dialogue 2006*, pp. 439–446, 2006.
7. J. Matoušek, D. Tihelka, and J. Romportl. Building of a Speech Corpus Optimised for Unit Selection TTS Synthesis. In: *LREC 2008, Proc. of 6th Int. Conf. on Language Resources and Evaluation*, ELRA, 2008.
8. M. Jůzová and D. Tihelka. Minimum Text Corpus Selection for Limited Domain Speech Synthesis. In: P. Sojka et al. (Eds.) *Proc. of Text, Speech and Dialogue*, 2014.
9. J. Švec and L. Šmídl Prototype of Czech Spoken Dialog System with Mixed Initiative for Railway Information Service In: P. Sojka et al. (Eds.) *Proc. of Text, Speech and Dialogue 2010*, pp. 568–575, 2010.
10. J. Ramportl and D. Tihelka Exploring Automatic Similarity Measures for Unit Selection Tuning. In: *Proc. of Int. Conf. Interspeech 2009*, pp. 736–739, 2009.
11. D. Tihelka, J. Kala and J. Matoušek Enhancements of Viterbi Search for Fast Unit Selection Synthesis. In: *Proc. of Int. Conf. Interspeech 2010*, pp. 174–177, 2010.