INFORMATICA, 2014, Vol. 25, No. 4, 551–562
© 2014 *Vilnius University*
DOI: http://dx.doi.org/10.15388/Informatica.2014.29

# Building Text Corpus for Unit Selection Synthesis

Pijus KASPARAITIS*, Tomas ANBINDERIS
*Department of Computer Science II, Faculty of Mathematics and Informatics*
*Vilnius University, Naugarduko 24, LT-03225 Vilnius, Lithuania*
*e-mail: pkasparaitis@yahoo.com, tomas@anbinderis.lt*

Received: February 2012; accepted: October 2014

**Abstract.** The present paper deals with building the text corpus for unit selection text-to-speech synthesis. During synthesis the target and concatenation costs are calculated and these costs are usually based on the prosodic and acoustic features of sounds. If the cost calculation is moved to the phonological level, it is possible to simulate unit selection synthesis without any real recordings; in this case text transcriptions are sufficient. We propose to use the cost calculated during the test data synthesis simulation to evaluate the text corpus quality. The greedy algorithm that maximizes coverage of certain phonetic units will be used to build the corpus. In this work the corpora optimized to cover phonetic units of different size and weight are evaluated.

**Key words:** text-to-speech synthesis, unit selection, greedy algorithm.

## 1. Introduction

Unit selection has been one of the most popular speech synthesis methods since the late 1990s, although recently other methods (e.g. harmonic and formant Pyz *et al.*, 2011, 2014) have been intensively investigated. As a general speech synthesis framework unit selection was first published in Hunt and Black (1996). As compared with the fixed unit synthesis, unit selection allows the distortion at the concatenation points to be reduced because there are plenty of units to choose from. The distortion can be even equal to 0 if the consecutive units are found in the speech corpus. Thus unit selection synthesis is a search through a large corpus of continuous speech at the runtime seeking to find the best sequence of the recorded units to produce the desired speech output. Prior to the search a phonetic and prosodic target specification should be obtained from the text. The search is based on two types of costs: the target cost and the concatenation cost. The target cost estimates the suitability of a speech corpus unit instance for the specific position in the target specification. Usually it is based on prosodic features (pitch, duration, position in the word and so on). The concatenation cost estimates the acoustic mismatch between the pairs of the units to be concatenated. The aim is to minimize the sum of all target and concatenation costs.

An alternative costs calculation method proposes to use phonological features rather than prosody and acoustics. Acoustics is assumed to be appropriate if units are taken from

*Corresponding author.

phonologically matching contexts. Several implementations of this idea have been published, e.g. the phoneme context tree (Breen and Jackson, 1998), phonological structure matching (Taylor and Black, 1999). Another implementation is presented in Yi and Glass (2002) where concatenation costs between pairs of phoneme classes rather than pairs of phoneme instances are calculated. The target cost is replaced with the left-sided substitution cost and the right-sided substitution cost. On the basis of the ideas presented in Yi and Glass (2002) we adapted the definitions of phoneme classes, concatenation and substitution costs for the Lithuanian language. Besides we showed how the search can be optimized.

Working with classes of phonemes rather than their instances allows one to investigate various characteristics of a speech corpus without real recordings. It suffices to have the corpus containing transcriptions of sentences. Using such a corpus we can already simulate synthesis of a certain test text and calculate the cost of synthesis and other "more traditional" characteristics, e.g. the average length of a phoneme string found in the corpus.

The speech corpus is very important to unit selection. The set of sentences selected according to some coverage criteria outperforms the set of randomly selected sentences. The greedy algorithm presented in Buchsbaum and van Santen (1997) is usually used to select sentences that give the best coverage of the certain phonetic units. Investigations into various modifications of the greedy algorithm seeking to create the corpus with the highest coverage of diphones and triphones are presented in François and Boëffard (2002, 2001). The following question might arise: is the set with full coverage of diphones better than the set with 70% coverage of triphones? We propose to use the above-mentioned simulation of synthesis to calculate the synthesis cost and to use this cost to measure the corpus quality.

The aim of this work is to propose a tool for evaluating the corpus quality and to find the best method for creating a corpus.

## 2. Algorithms for Synthesis and Corpus Building

### 2.1. *Synthesis Algorithm*

We have chosen phonemes to be basic synthesis units. Suppose our task is to synthesize the phrase containing 3 phonemes $\alpha\beta\gamma$. Suppose the phoneme $\alpha$ has already been found in the corpus and the phoneme $\beta$ is to be concatenated to it; however, the phoneme $\beta$ in the corpus belongs to quite a different context, e.g. $\delta\beta\epsilon$. According to Yi and Glass (2002), the cost is calculated as follows:

$$P(\beta) = C(\alpha, \beta) + S_L\big([\alpha]\beta, [\delta]\beta\big) + S_R\big(\beta[\gamma], \beta[\epsilon]\big), \tag{1}$$

where $C(\alpha, \beta)$ is the concatenation cost, $S_L([\alpha]\beta, [\delta]\beta)$ is the left substitution cost (phoneme $\beta$ following $\alpha$ is substituted with phoneme $\beta$ following $\delta$), $S_R(\beta[\gamma], \beta[\epsilon])$ is the right substitution cost (phoneme $\beta$ preceding $\gamma$ is substituted with phoneme $\beta$ preceding $\epsilon$).

Concatenation and substitution costs can be tuned manually or computed from the data. This issue will no longer be discussed here. The cost matrices and phoneme classes presented in Yi (1998, 2003) will be used here after they have been converted from a graphical representation into a numerical one (values from 0 to 1) and adapted to the Lithuanian language (i. e. Lithuanian phonemes were assigned to respective classes, stops and fricatives were divided into voiced and unvoiced ones, affricates were attributed to the stops when talking about the right context and to the fricatives when talking about the left context). It is very important to note that the concatenation cost $C(\alpha, \beta) = 0$ if the instances of $\alpha$ and $\beta$ are consecutive phonemes in the corpus. Otherwise $C(\alpha, \beta)$ should be taken from the precalculated 2-dimensional cost matrix. Costs in this matrix don't depend on the positions of phonemes in the corpus. The substitution costs $S_L$ and $S_R$ are always taken from two precalculated 3-dimensional matrices.

The Viterbi algorithm is usually used to find the best sequence of the phonemes in the corpus. We optimized the Viterbi search on the basis of the above-mentioned fact of the concatenation costs of the consecutive and non-consecutive phonemes. Let us analyze separately the phonemes $\alpha$ before $\beta$ ($\alpha[\beta]$) and the phonemes $\alpha$ before any other phoneme except $\beta$ ($\alpha[\hat{\beta}]$). The concatenation costs can be written as follows:

$$C\big(\alpha[\beta], \beta\big) = \begin{cases} 0, & \text{if } \alpha \text{ and } \beta \text{ are consecutive,} \\ c, & \text{if } \alpha \text{ and } \beta \text{ are nonconsecutive,} \end{cases} \tag{2}$$

$$C\big(\alpha[\hat{\beta}], \beta\big) = c. \tag{3}$$

It is obvious that any instance of $\beta$ can be concatenated to $\alpha[\beta]$, and not only its neighbor $\beta$. It is impossible to know in advance (without the Viterbi search), which $\alpha[\beta]$ will belong to the minimum cost path. The case is quite different with $\alpha[\hat{\beta}]$. Those $\alpha[\hat{\beta}]$ that cannot belong to the minimum cost path can be immediately detected and removed from the search. Suppose we have $\alpha_1[\hat{\beta}]$ and $\alpha_2[\hat{\beta}]$, so that $P(\alpha_1) < P(\alpha_2)$. Thus the following inequality is correct: $P(\alpha_1) + P(\beta) = P(\alpha_1\beta) < P(\alpha_2\beta) = P(\alpha_2) + P(\beta)$ since $C(\alpha_1, \beta) = C(\alpha_2, \beta)$. The same holds true for longer sequences, i.e. $P(\alpha_1) + P(\beta\gamma\ldots) = P(\alpha_1\beta\gamma\ldots) < P(\alpha_2\beta\gamma\ldots) = P(\alpha_2) + P(\beta\gamma\ldots)$. This means that $\alpha_2[\hat{\beta}]$ can be excluded from consideration because it never belongs to the minimum cost path.

Now the search algorithm can be defined as follows: first of all we look for all the instances of $\alpha[\beta]$, memorize them and calculate their costs $P(\alpha)$. Then we look for all the instances of $\alpha[\hat{\beta}]$ but memorize only a single instance based on the minimum cost $P(\alpha)$ (see Fig. 1 left).

Next we look for the phonemes $\beta$ in the corpus. If the instance of the phoneme $\beta[\gamma]$ is found, we start a search in the memorized list seeking to find the instance of $\alpha$ with the minimum sum of the costs $P(\alpha\beta) = P(\alpha) + P(\beta)$. The cost $P(\alpha\beta)$ and the sequence of instances $\alpha\beta$ are memorized (bold lines in Fig. 1). If the instance of the phoneme $\beta[\hat{\gamma}]$ is found, we start a search in the memorized list in a similar way and find the instance of $\alpha$ with the minimum sum of the costs $P(\alpha\beta) = P(\alpha) + P(\beta)$. However, again we choose to memorize only one sequence of $\alpha\beta$ with the minimum cost $P(\alpha\beta)$ (see Fig. 1 right). After that the unused instances of $\alpha$ can be removed from the list and a search for the phoneme $\gamma$ can be started.

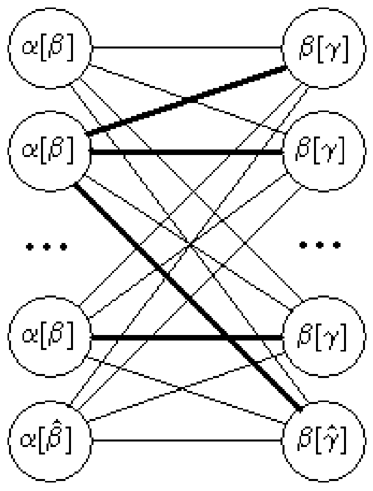

Fig. 1. Viterbi algorithm (optimized).

Since we use 92 phonemes, the proposed optimization allows us to speed up the algorithm by approximately 92 times (instead of examining all instances of the phoneme $\alpha$ prior to any of 92 phonemes we examine only the instances preceding the phoneme $\beta$).

## 2.2. *Corpus building algorithm*

The greedy algorithm presented in Buchsbaum and van Santen (1997) is usually used to create the text corpus that is read by an announcer and serves as a speech database in text-to-speech synthesis. This corpus should cover most phonetic units (e.g. diphones, syllables, etc.) found in a large set. Hence, we need a large set of sentences (their transcriptions) and a list of all phonetic units found in this set. The algorithm successively selects sentences and adds them to the corpus. The sentence with the largest number of different phonetic units will be selected first. All units occurring in this sentence are removed from the list of units. The sentence with the largest number of different remaining units will be the second selected sentence and so on.

The above-described method guarantees that the minimum number of sentences that cover a certain set of units will be selected. However, this method tends to select long sentences first. Usually we want the corpus to require the minimum amount of time for an announcer to read, so it should contain the minimum number of phonemes. This can be achieved by dividing the number of the new units found in the sentence by the sentence length (François and Boëffard, 2002).

In the above-described algorithm all units are assumed to have the same weights equal to 1 (in the future experiments we will denote this "1"). It is obvious that different weights can be used, e.g. directly proportional to the frequency of a unit (denoted "$f$"), or inversely proportional to the frequency of a unit (denoted "$1/f$"). We proposed to use weights equal to the sum of all concatenation costs in the unit (e.g. in the case of triphones $C(\alpha, \beta) +$

$C(\beta, \gamma)$) (denoted "$j$") and equal to the frequency multiplied by the sum of concatenation costs (denoted "$fj$").

It is stated in Buchsbaum and van Santen (1997) that in order to achieve a full coverage the weights "$1/f$" should be used, but in the case of incomplete coverage these weights give unsatisfactory results. As long as full coverage is hard to achieve, we suggest that the above-mentioned fact should be exploited in the following way: the most rarely used units should be removed from the list so that the remaining units cover 99%, the weights "$1/f$" should be used for the remaining units (denoted "$1/fr$").

## 3. Experiments of Corpus Building

Many experiments were carried out using a small amount of data. The aim was to reject those methods and algorithms that were not worth carrying out on a large amount of data. Later most promising methods were tested using a large amount of data. During these experiments corpora were built using the greedy algorithm and their quality (synthesis cost and other characteristics) was evaluated.

In order to ensure that a similar amount of data is used in different experiments, sentences were selected until the number of phonemes exceeded the predefined threshold. Thus the number of phonemes in the selected sentences varied only slightly, whereas the number of sentences could vary much more considerably.

Approximately 200 sentences were selected when the threshold of 6000 phonemes was used, 2000 sentences – 60 000 phonemes, 5000 sentences – 150 000 phonemes. For simplicity only the approximate number of sentences will be specified in the future.

### 3.1. *Experiments with a Small Amount of Data*

During the experiments as many as 675 short sentences were cut out of a literary text and their transcriptions were automatically generated. The phoneme system of the Lithuanian language described in Kasparaitis (2005) was used in this work. Stressed and unstressed sounds are treated as different phonemes, thus this system contains 92 phonemes in total. Approximately 200 sentences were selected with the help of the greedy algorithm, and the remaining unselected sentences were used for testing.

One group of experiments was carried out using $N$ consecutive phonemes as units of the greedy algorithm. We shall refer to them as $N$-phones. The average costs per phoneme (total cost divided by the number of phonemes) when various $N$-phones and various unit weights were used are presented in Fig. 2. 5-phones with a vowel in the third position (denoted as 5*phones) were also used. The latter units were introduced in order to constrain the number of units because in the case of 5-phones it grew significantly.

As can be seen from Fig. 2, the lowest cost was achieved when 3-phones were used. Slightly worse results were obtained when 4- or 5-phones were used. The worst results were produced when 2-phones were used. The best weighting method was "$fj$", and the method "$f$" was slightly worse. Our method "$1/fr$" outperforms only methods "$1/f$" and "1".

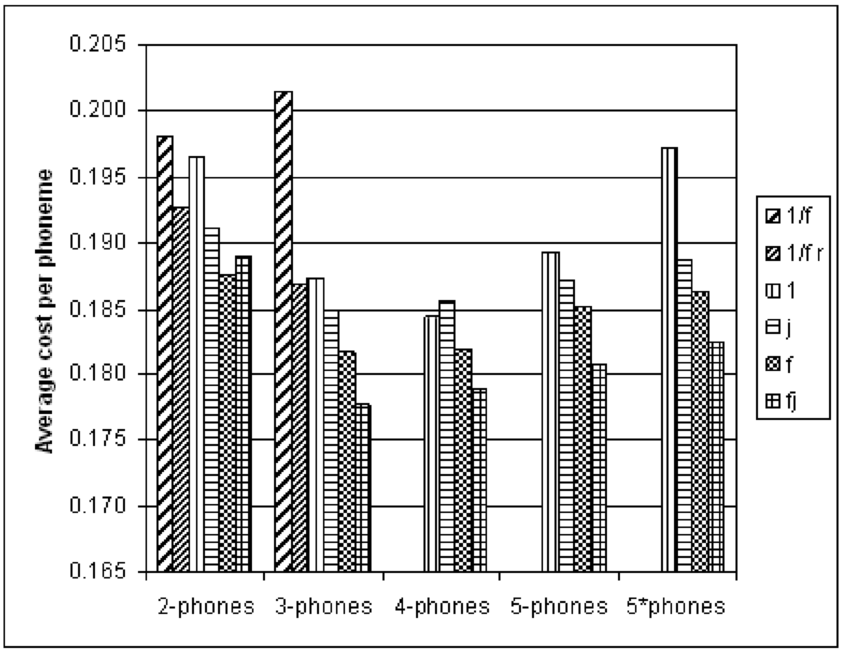

Fig. 2. Test data synthesis simulation costs for various $N$-phones and weighting methods (a small amount of data).

Another group of experiments was carried out using words and syllables as units. Besides, an experiment where both words and syllables were used to select a sentence was also conducted. The following weighting methods were employed: "$1/f$", "1" and "$f$". In addition, three improvements based on the idea proposed in Bozkurt *et al.* (2003) were examined. The idea was as follows: if a unit appears both in the selected and unselected sentences but within different contexts, the value of the unselected sentence should be increased by a certain amount. In the first case the context was a neighboring word/syllable and the amount was 0.2 (this experiment will be designated as "nws02"), in the second and third cases the context was a neighboring phoneme and the amount was 0.2 (designated "nph02") and 0.4 (designated "nph04"), respectively. In essence, these three methods are modifications of method "1".

The average costs per phoneme when word/syllable sized units and various unit weights were used are presented in Fig. 3.

As can be seen from Fig. 3, the cost slightly decreases when both words and syllables are used. The best results were achieved when the weighting method "$f$" was used. The last three modifications improved the results as compared with the method "1" but the results were still not as good as when the method "$f$" was used.

In order to compare the results achieved using $N$-phones with those achieved using words and syllables, general results when the weighting method "$f$" was used are presented in Fig. 4. Besides, results of three more sophisticated experiments are presented in Fig. 4. The first experiment was carried out to choose sentences with the highest synthesis

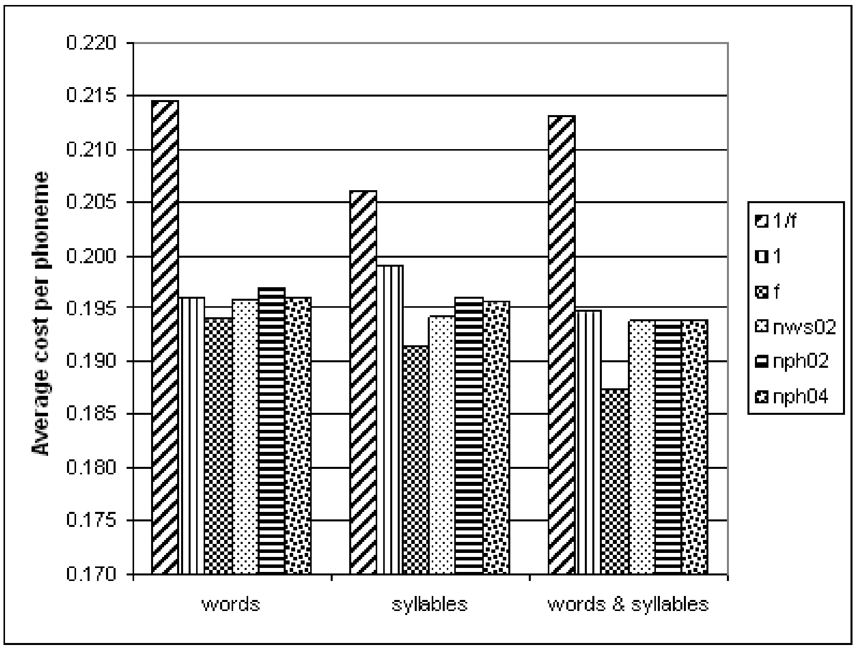

Fig. 3. Test data synthesis simulation costs for words, syllables and for both with various weighting methods (a small amount of data).

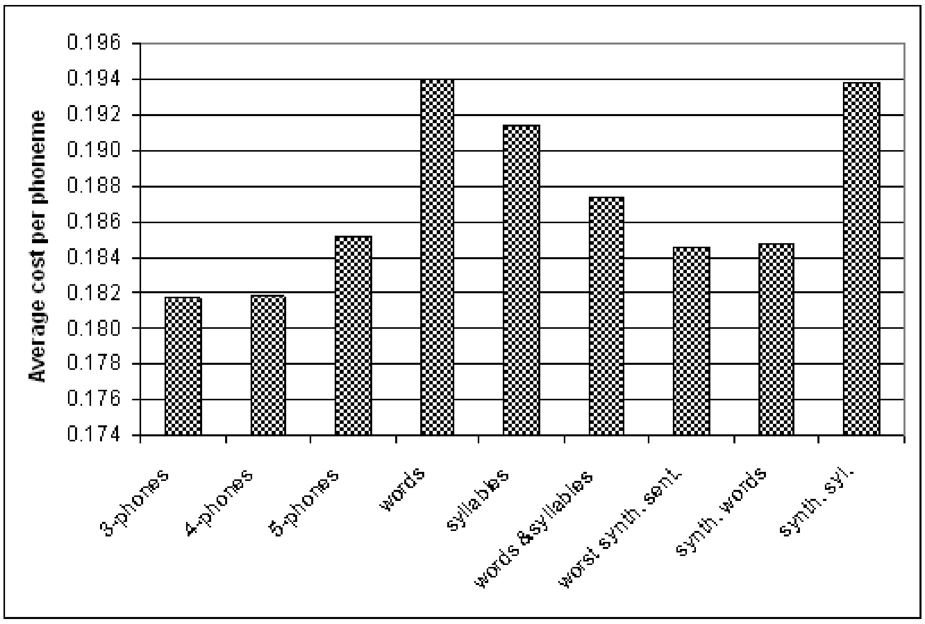

Fig. 4. Test data synthesis simulation costs. General results (a small amount of data).

Table 1
Evaluation of small corpora using "traditional" measures.

| Units | Weighting method | Initial algorithm | | | Reduced concatenation cost at word/syllable boundaries | | |
|---|---|---|---|---|---|---|---|
| | | Consecutive phonemes | Average phoneme string length | Concat. points inside a syllable (%) | Consecutive phonemes (%) | Average phoneme string length | Concat. points inside a syllable (%) |
| 3-phones | $f$ | 67.8 | 2.94 | 35.6 | 66.9 | 2.87 | 23.4 |
| | $fj$ | 67.3 | 2.90 | 36.4 | 66.5 | 2.83 | 23.6 |
| 4-phones | $f$ | **68.1** | **2.96** | 35.2 | **67.4** | **2.90** | 24.1 |
| | $fj$ | 67.9 | 2.95 | 37.3 | 67.2 | 2.89 | 26.1 |
| 5-phones | $f$ | 67.7 | 2.93 | 36.8 | 67.0 | 2.87 | 26.1 |
| | $fj$ | 67.3 | 2.90 | 37.6 | 66.6 | 2.84 | 27.0 |
| Words | $f$ | 66.5 | 2.83 | 35.0 | 65.8 | 2.78 | 24.4 |
| Syllables | $f$ | 66.5 | 2.83 | **31.3** | 65.7 | 2.78 | **19.8** |
| Words & syllables | $f$ | 67.4 | 2.90 | 33.5 | 66.6 | 2.84 | 22.0 |

cost, i.e. unselected sentences were synthesized using the already selected sentences, and the synthesis cost was estimated. The sentence with the highest cost was added to the corpus of the selected sentences and the process was repeated. The corpus built of sentences containing a single phoneme was used at the beginning of the process. Lists of words and syllables were used in other two experiments. The synthesis costs of these words and syllables were calculated using the already selected sentences. These costs multiplied by the frequency of a word/syllable were used as a unit cost. A new sentence with the lowest cost was added to the corpus, and the word/syllable costs were recalculated.

As can be seen from Fig. 4, $N$-phones outperform words and syllables. We discovered that the last three methods required a lot of computational time but the results were still inferior to those achieved using 3- or 4-phones. So we are not going to examine them in the future.

Using the synthesized test data the following more "traditional" measures can be calculated in addition to the synthesis cost:

- the percentage of the consecutive phonemes;
- the average length of a string of consecutive phonemes;
- the percentage of concatenation points inside a syllable etc.

The synthesized test data evaluated according to those three criteria are presented in Table 1 on the left. Nine methods with the least synthesis cost were employed.

However, the algorithm used does not take into account the fact whether the sounds are concatenated inside the syllable or at the boundary. It is obvious that concatenation points at word or syllable boundaries are somewhat less perceptible hence concatenation costs at these boundaries should be lower. The synthesis algorithm was modified as follows: concatenation costs at the syllable boundaries were multiplied by factor 0.6, and at the word boundary – by factor 0.3. Since the synthesis costs calculated using the modified

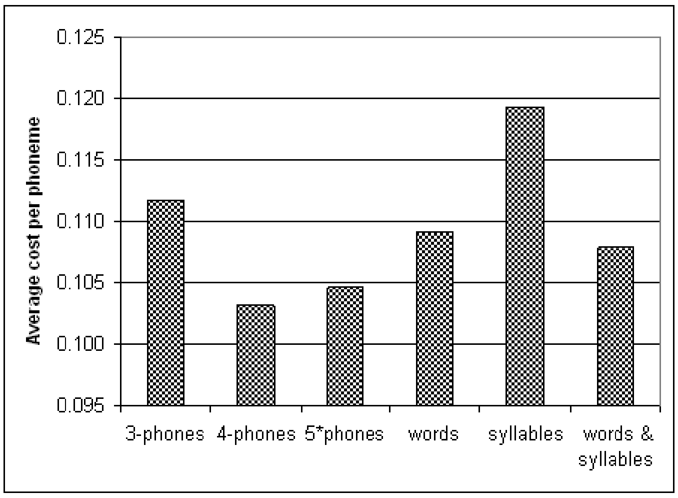

Fig. 5. Test data synthesis costs (a large amount of data).

algorithm cannot be compared with those calculated prior to modification, three above-mentioned "traditional" criteria were used to evaluate the algorithms (see the results in Table 1, on the right).

Table 1 shows that the highest percentage of consecutive phonemes and the longest strings of consecutive phonemes are found when 4-phones together with weighting method "$f$" were used. The least number of concatenation points inside a syllable was achieved when using syllables. The modified algorithm slightly decreases the percentage of consecutive phonemes and the length of strings of consecutive phonemes but the number of concatenation points inside a syllable decreases drastically. It is also worth noting that the method "$f$" outperformed the method "$fj$" in all cases.

## 3.2. *Experiments with a Large Amount of Data*

A large amount of stressed text containing about one million words (see Anbinderis and Kasparaitis, 2009 for details) was used in these experiments. The text was split into phrases according to the punctuation marks. If two consecutive phrases were shorter than 28 letters each and were separated by a comma, they were combined. This process could be continued iteratively using the already combined phrases. Only phrases of the length between 28 and 80 letters were selected thus producing a data set containing 84 024 phrases (sentences) and a testing set containing 20 640 sentences. Sentences were transcribed automatically. Corpora containing approximately 2000 sentences were built from the data set using six types of units that proved to be best when working with a small amount of data. Frequencies of units (method "$f$") were used as their weights. The test data synthesis simulation costs for various units are presented in Fig. 5. Other features of the corpora obtained using the initial and modified algorithms are presented in Table 2.

Table 2
Evaluation of large corpora using "traditional" measures.

| Units | Initial algorithm | | | Reduced concatenation cost at word/syllable boundaries | | |
|---|---|---|---|---|---|---|
| | Consecutive phonemes (%) | Average phoneme string length | Concat. points inside a syllable (%) | Consecutive phonemes (%) | Average phoneme string length | Concat. points inside a syllable (%) |
| 3-phones | 75.0 | 3.70 | 31.1 | 74.0 | 3.58 | 13.7 |
| 4-phones | 76.4 | 3.90 | 30.2 | 75.5 | 3.77 | 13.7 |
| 5*phones | **76.6** | **3.93** | 30.7 | 75.6 | 3.78 | 14.4 |
| Words | 76.4 | 3.90 | 30.0 | 75.7 | 3.80 | 13.8 |
| Syllables | 74.4 | 3.63 | 29.6 | 73.4 | 3.50 | **11.3** |
| Words & syllables | 76.5 | 3.92 | **28.8** | **75.8** | **3.82** | 11.9 |

Table 3
Changes in the corpus features by increasing the corpus size from 2000 to 5000 sentences.

| Cost | Consecutive phonemes (%) | Average phoneme string length | Concatenation points inside a syllable (%) |
|---|---|---|---|
| −22.3% | +2.4% | +0.385 phonemes (+10.0%) | −1.7% |

As can be seen in Fig. 5, the lowest cost was obtained using 4-phones. Very similar results were obtained using 5*phones, 3-phones produced significantly worse results. Thus the larger the corpus is, the longer units should be used. However, since the use of 5-phones was impossible (too many different units), we used only those with a vowel as the middle phoneme.

Other features of the corpus, i.e. the percentage of consecutive phonemes, the average length of a string of consecutive phonemes, also moved from 4-phones in the case of a small corpus to 5*phones in the case of a large one.

As it has been mentioned earlier, it is possible to reduce the number of concatenation points inside a syllable significantly by decreasing the concatenation costs at the word/syllable boundaries. In this case the largest percentage of consecutive phonemes and the average length of a string of consecutive phonemes were achieved by maximizing coverage of words and syllables (rather than 4-phones). The smallest number of concatenation points inside a syllable was obtained using syllables.

One more experiment was carried out using 4-phones seeking to examine how various features of the corpus changed by increasing the corpus size from 2000 to 5000 sentences. See the results in Table 3.

Table 3 shows that the percentage of consecutive phonemes and concatenation points inside a syllable improved only slightly, the average length of strings of consecutive phonemes increased more significantly and the synthesis cost decreased quite drastically. A large number of sentences seem to enable the segments with a significantly lower concatenation cost to be found. Thus the conclusion can be drawn that the synthesis cost is a better measure of corpus quality than other three above-mentioned measures.

## 4. Conclusions

The corpus building for unit selection synthesis was investigated in this work. If we move the calculation of the target and concatenation costs onto the phonological level, synthesis can be simulated without real voice recordings. In this case transcriptions of sentences are sufficient. In the present work we proposed to use the cost calculated during the test data synthesis as a quality measure of the text corpus. The method decreasing the search time by almost 100 times was also described. The greedy algorithm that maximizes coverage of certain phonetic units was employed to build the corpus. A great number of corpora were build using the greedy algorithm with units of various size and weight. We evaluated the quality of the corpora on the basis of the cost and other features. The following conclusions can be drawn:

- The lowest cost was obtained using 3-phones if the corpus was small, but in case of a large corpus the units had to be larger (4- or even 5-phones). The use of 5-phones was problematic because the number of different units grew rapidly so the number of units had to be limited. The use of 2-phones (diphones) proved to be useless despite the fact that they were often used by other authors.
- The percentage of consecutive phonemes and the average length of a string of consecutive phonemes were maximal using 4-phones in case of a small corpus and 5-phones in case of a large one.
- The smallest number of concatenation points inside a syllable was obtained using syllable-sized units.
- In the synthesis algorithm, the reduction of the concatenation costs at the word and syllable boundaries enabled the number of concatenation points inside a syllable to be reduced significantly. Thus the largest percentage of consecutive phonemes and the average length of a string of consecutive phonemes were achieved by maximizing coverage of words and syllables.
- The weights of units proportional to their frequency worked best in the greedy algorithm. In case of a small corpus a slightly better results were achieved by multiplying those weights by the sum of concatenation costs but in case of a large corpus the results were about the same.
- An increase in the size of the corpus decreases the synthesis cost significantly. Other features of the corpus improve only slightly. This leads to the conclusion that the synthesis cost is a good measure of the corpus quality.

**Acknowledgments.** This research has been supported by Algoritmų sistemos Ltd. and by the project "Services Controlled through Spoken Lithuanian Language (LIEPA)" (No. VP2-3.1-IVPK-12-K-01-001) funded by the European Structural Funds.

## References

Anbinderis, T., Kasparaitis, P. (2009). Disambiguation of Lithuanian homographs based on the frequencies of lexemes and morphological tags. *Kalbų studijos = Studies about languages*, 14, 25–31 (in Lithuanian).

Bozkurt, B., Ozturk, O., Dutoit, T. (2003). Text design for TTS speech corpus building using a modified greedy selection. In: *Eurospeech 2003*, pp. 277–280.

Breen, A.P., Jackson, P. (1998). Non-uniform unit selection and the similarity metric within BT's laureate TTS system. In: *Proceedings of the Third ESCA Workshop on Speech Synthesis*, pp. 373–376.

Buchsbaum, A., van Santen, J. (1997). Methods for optimal text selection. In: *Eurospeech 1997*, pp. 553–556.

François, H., Boëffard, O. (2001). Design of an optimal continuous speech database for text-to-speech synthesis considered as a set covering problem, In: *Interspeech 2001*, pp. 829–832.

François, H., Boëffard, O. (2002). The greedy algorithm and its application to the construction of a continuous speech database. In: *Proceedings of LREC 2002*, pp. 1420–1426.

Hunt, A., Black, A. (1996). Unit selection in a concatenative speech synthesis system using a large speech database. In: *ICASSP 1996*, Atlanta, pp. 373–376.

Kasparaitis, P. (2005). Diphone databases for Lithuanian text-to-speech synthesis. *Informatica*, 16(2), 193–202.

Pyz, G., Simonyte, V., Slivinskas, V. (2011). Modelling of Lithuanian speech diphthongs. *Informatica*, 22(3), 411–434.

Pyz, G., Simonyte, V., Slivinskas, V. (2014). Developing models of Lithuanian speech vowels and semivowels. *Informatica*, 25(1), 55–72.

Taylor, P., Black, A.W. (1999). Speech synthesis by phonological structure matching. In: *Eurospeech 1999*, pp. 623–626.

Yi, J. (1998). *Natural-sounding speech synthesis using variable-length units*. Master thesis. Massachusetts Institute of Technology.

Yi, J. (2003). *Corpus-based unit selection for natural-sounding speech synthesis*. Doctor thesis, Massachusetts Institute of Technology.

Yi, J., Glass, J. (2002). Information-theoretic criteria for unit selection synthesis. In: *Interspeech 2002*, pp. 2617–2620.

**P. Kasparaitis** was born in 1967. In 1991 he graduated from Vilnius University (Faculty of Mathematics). In 1996 he became a PhD student at Vilnius University. In 2001 he defended the PhD thesis. Current research includes text-to-speech synthesis and other areas of computer linguistics.

**T. Anbinderis** was born in 1981. In 2005 he graduated from Vilnius University (Faculty of Mathematics and Informatics). In 2005 he was admitted as a PhD student to Vilnius University. In 2010 he defended the PhD thesis. Current research interests include text-to-speech synthesis.

## Tekstyno vienetų parinkimo sintezei sudarymas

Pijus KASPARAITIS, Tomas ANBINDERIS

Šiame darbe nagrinėjamas tekstyno, skirto vienetų parinkimo sintezei, sudarymas. Sintezės metu skaičiuojamos tikslinės ir jungimo kainos, kurios paprastai remiasi prozodiniais ir akustiniais garsų požymiais. Perkėlus kainų skaičiavimą į fonologinį lygmenį galima imituoti vienetų parinkimo sintezę neturint balso įrašų, o tik teksto transkripcijas. Šiame darbe pasiūlyta testinių duomenų sintezės imitavimo metu apskaičiuotą kainą panaudoti tekstyno kokybei įvertinti. Tekstynui sudaryti naudotas algoritmas, kuris stengiasi kuo geriau padengti tam tikrų fonetinių elementų aibę. Darbe įvertinta tekstynų, optimizuotų padengti įvairaus dydžio elementus su įvairiais svoriais, kokybė.