Speech Synthesis Development Made Easy: The Bonn Open Synthesis System

Esther Klabbers[†], *Karlheinz Stöber*[*], *Raymond Veldhuis*[†], *Petra Wagner*[*] *and Stefan Breuer*[*]

[†]IPO, Center for User-System Interaction, Eindhoven, The Netherlands
{E.A.M.Klabbers/R.N.J.Veldhuis}@tue.nl
[*]IKP, University of Bonn, Germany
{kst/pwa/breuer}@ikp.uni-bonn.de

Abstract

This paper describes a new open source architecture for unit-selection based speech synthesis called BOSS (Bonn Open Synthesis System). It is built up modularly, with communications between modules taking place in a fixed format. This makes the addition, deletion and substitution of modules very easy. The strict separation between data and algorithms allows for the simple creation of new speech corpora for different domains and languages.

1. Introduction

During the past few years, unit selection has become a very popular speech synthesis method [1, 2, 8]. This technique is based on the on-line selection of variable-sized units from a large speech corpus. The aim is to minimize the occurrence of audible discontinuities at junctions and minimize distortions due to signal manipulation, by selecting the largest possible units that most closely match a predefined target and context.

In this paper, we present the Bonn Open Synthesis System (BOSS), which offers a general framework for *non-uniform unit-selection-based synthesis*. The system is based on the synthesis concept used in Verbmobil [5, 7]. The software was completely redeveloped because 1) the software used in Verbmobil was not free, 2) the software in Verbmobil was closely intertwined with other modules, and 3) the software in Verbmobil could still be improved.

BOSS shows a clear separation between data and algorithms, allowing for the easy creation of synthesis systems for different languages and domains without the need to work on the algorithms.

BOSS is open source software. This introduces a great deal of flexibility, which one can use to tailor the system to the application. For instance, for domain-specific applications such as train timetable information systems, it is possible to add a domain-specific corpus, containing sentences from the domain. Additionally, it is possible to skip the linguistic analysis part when the synthesis is preceded by a language generation module in a data-to-speech system[1]. The open source character of BOSS also makes it an excellent research tool. Researchers can avail the modules and corpora they developed. The user is responsible for the speech corpus. This means corpora can also be made available commercially. Section 2 will show that BOSS has a predefined architecture which is set up in such a way as to make addition or substitution of different modules and corpora very easy.

[1]This is also referred to as concept-to-speech.

BOSS is still under development. This means that not all modules are fully implemented yet. It is our hope that more speech synthesis researchers will join us in the future to contribute their sources and corpora.

2. Architecture

BOSS is written in C++ and developed for the Linux platform. It falls for the most part under the GNU Public License. Currently, it is still in a development stage, meaning that not all modules are fully available. The architecture of BOSS is visualized in Figure 1. In BOSS, the system architecture is clearly separated from the data. This makes it easy to switch between different speech corpora and even different languages. In principle, no knowledge of the source code is required to create synthesis systems for different domains and languages. Section 3 will show that it was possible to create working systems for German and Dutch without any changes to the source code.

The data is stored in two databases: the speech database and the SQL database which contains information about the speech data. These will be explained in more detail in Section 3. As can be seen, the architecture is split up into a client and a server part. The client (Section 2.1) is a small program which can run remotely. It sends input in the required format to the server and receives the speech signal. The server (Section 2.2) contains the actual synthesis software. It consists of separate modules, which are independent of each other. Each module realises a synthesis step (e.g. transcription). The communication between server and modules takes place in a fixed format. Thus, modules can be developed and tested separately.

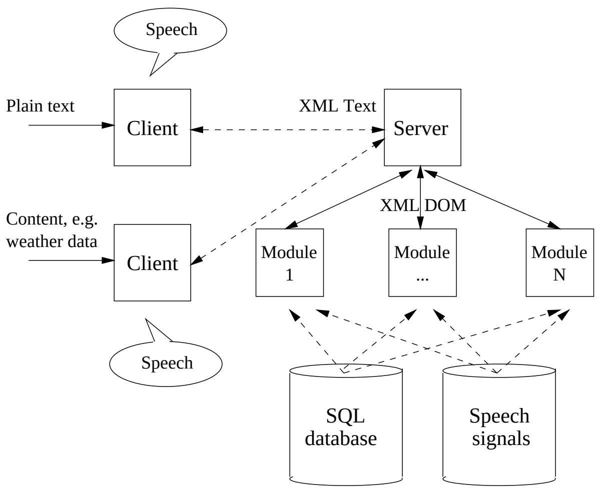

Figure 1: Overview of the architecture of BOSS.

2.1. The BOSS client

The BOSS client is a program that is responsible for sending input to the BOSS server and receiving output from the server. It can run remotely on different platforms. The input is plain text. The output is structured in an XML document. XML provides a powerful, flexible and intuitive way to structure data. Figure 2 shows an example document for a given text example. A text is hierarchically structured into sentences, which are divided into words. The sentence type is indicated. "." stands for declarative sentences and "?" for questions, other types may be distinguished in the future. We call this a minimal XML document.

Text example:
Guten Tag Herr Müller. Wie geht es Ihnen heute?

XML document:

```
<SENTENCE Type=”.”>
        <WORD Orth=”Guten”></WORD>
        <WORD Orth=”Tag”></WORD>
        <WORD Orth=”Herr”></WORD>
        <WORD Orth=”Müller></WORD>
</SENTENCE>

<SENTENCE Type=”?”>
        <WORD Orth=”Wie”></WORD>
        <WORD Orth=”geht”></WORD>
        <WORD Orth=”es”></WORD>
        <WORD Orth=”Ihnen”></WORD>
        <WORD Orth=”heute”></WORD>
</SENTENCE>
```

Figure 2: Example output from the BOSS client.

Different clients may be created for different purposes. For instance, in a data-to-speech system information about accentuation and phrasing can be inserted into the XML document. In this case, the applicable server module remains inactive and the utterance is produced with the phrase and accent structure as generated by the client.

2.2. The BOSS server

The BOSS server is the core of the system. It is structured modularly to enable integration of new modules at any time. The modules are made available in the form of C++ classes. The input to the server is a sentence coded in XML. The module in question then adds its output to the XML document (i.e., the transcription of a given orthography). The modules communicate with each other using the XML Document Object Model (DOM)[2]. This is an API for HTML and XML which defines the logical structure of documents and the way a document is accessed and manipulated. With XML-DOM, the server can navigate the XML documents and add, modify or delete elements and content.

The system can be divided into three main modules, which will be discussed in more detail below:

- Linguistic analysis
- Unit Selection
- Synthesis

2.2.1. Linguistic analysis

At the moment, linguistic analysis consists only of finding the phonetic transcription. Phrasing is based on the punctuation marks and accentuation is not computed. However, in the future this functionality would be desirable. The transcription module takes as input the minimal XML-document supplied by the client. It expands this document to include the phonetic transcription with stress marks and phrase boundary marks. The result is a maximal XML-document. Part of this XML document for the example given previously is depicted in Figure 3.

```
<SENTENCE Type=".">
    <WORD Orth="Guten" PInt="0" TKey="gu:t@n" CLeft="pau" CRight="t">
        <SYLLABLE TKey="gu:" CLeft="pau" CRight="t" Stress="1">
             <PHONEME Tkey="g" CLeft="pau" CRight="u:" Stress="0"/>
             <PHONEME TKey="u:" CLeft="g" CRight="t" Stress="1"/>
        </SYLLABLE>
        <SYLLABLE TKey="t@n" CLeft="u:" CRight="t" Stress="0">
             <PHONEME TKey="t" CLeft="u:" CRight="@" Stress="0"/>
             <PHONEME TKey="@" CLeft="t" CRight="n" Stress="0"/>
             <PHONEME TKey="n" CLeft="@" CRight="t" Stress="0"/>
        </SYLLABLE>
    </WORD>
    <WORD>
     ...
    </WORD>
</SENTENCE>
```

Figure 3: Part of the transcription for the sentence "Guten Tag Herr Müller".

In the current version of BOSS, the transcription task is solved by a process consisting of three stages: exception lexicon lookup, morpheme based transcription, and rule based transcription . For German, the lexical lookup

[2]XML-DOM is implemented using the XML4C package of IBM falling under the Apache license.

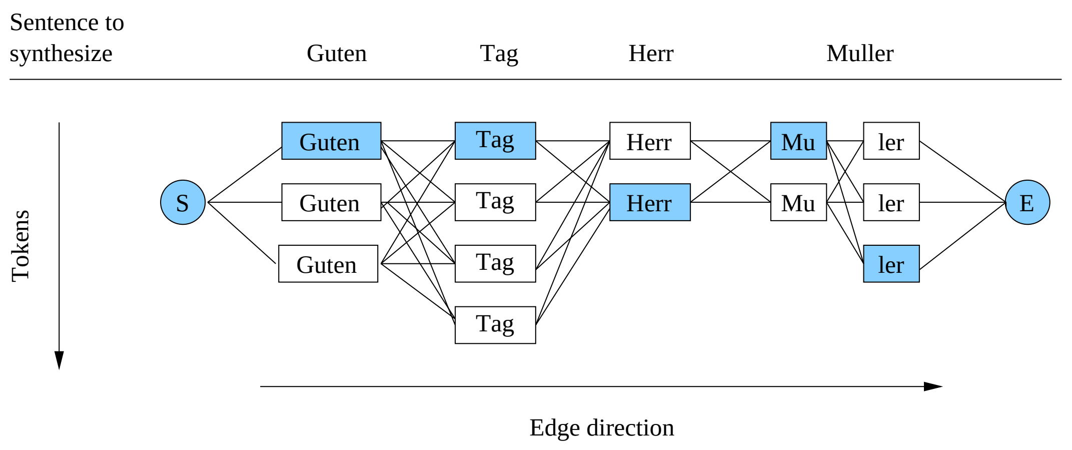

Figure 4: Example graph for the sentence “Guten Tag Herr Müller”.

(70,000 entries) and a morphology-based transcription module are available. For Dutch, the CELEX database is employed as an exception lexicon, containing approximately 300,000 entries. The table lists the orthographic and phonetic transcription. The phonetic transcription includes stress marks and syllable boundaries.

2.2.2. Unit Selection

The unit selection algorithm in BOSS is similar to the one in Verbmobil. In Verbmobil, only word concatenation and construction of missing words by phoneme concatenation was possible [6, 7].

The algorithm consists of two stages. In the first stage the available units are selected on the basis of the target segmental structure, unit location within the phrase and lexical stress (*candidate selection*). The basic building block in BOSS is the *word*. If a word is not available in the corpus, units are searched at a lower level, in this case the *syllable* level. If a syllable is not available, search continues at the *phoneme* level, which is the lowest level. This strategy makes unit selection very efficient and allows for the use of very large corpora.

In the second stage, a final selection is made (*unit selection*). For each possible unit, a set of possible candidates is available in the corpus. These candidate units are represented as nodes in a graph (see Figure 4). The edges represent the possible transitions between units. Cost functions are applied to these edges. Currently two cost functions are implemented, one enforcing units occurring adjacent to each other in a sentence to be selected over units from different sentences and the other using the Euclidean MFCC distance to measure spectral similarity. (At the moment we are exploring other cost functions since research by Klabbers and Veldhuis [4] has shown that the Euclidean MFCC distance does not correspond very well to human perception.) As described in Campbell and Black [2], the unit selection algorithm then tries to find the best path through the graph by computing the minimal sum of cost values on a path between start and end node.

No specific phonetic target values like duration and pitch are computed. Because the unit selection takes the hierarchical structure of the utterances into account, plus additional features such as location relative to phrase boundaries and stress, the selected units automatically match the desired context. This approach has a similar effect as the phonological structures used by Taylor and Black [9].

2.2.3. Synthesis

The synthesis component adds the selected units to the XML structure and concatenates them. No signal manipulation functionality is implemented yet. For limited domains this is no problem, since the corpus can be constructed in such a way that it perfectly covers the intended domain. However, for unrestricted text-to-speech it is impossible to get perfect coverage and a trade-off is necessary between corpus size and signal manipulations. These manipulations would be used at the occurrence of considerable pitch jumps or at the absence of the appropriate prosody in the corpus.

3. Data

3.1. Speech Corpus

For the German version of BOSS the Verbmobil corpus is used, containing 4545 sentences (automatically segmented phoneme boundaries, manually corrected word boundaries). The Verbmobil domain features approximately 10,000 words. The sentences are generated from actual planning dialogue transcriptions, where all needed words are included with sufficient variation.

For Dutch, a preliminary corpus of 297 sentences is implemented. These sentences were not specifically selected for use in a unit selection system, but they were already manually labeled, since they were used to create a duration model for Dutch [3]. The sentences came from news transcriptions and provide a good phonetic coverage. Still, the corpus is not nearly big enough to generate high quality speech. We are currently in the process of designing a new corpus for use in unrestricted text-to-speech.

3.2. SQL Database

The XML files that accompany the speech corpus and the transcription lookup table are stored in SQL tables to facilitate fast search and easy manual access/modification for these data. The *MySQL* package is used for maintaining these tables. The SQL database for a given language contains the following tables:

1. *Phoneme data*: currently containing all phoneme instances in the corpus together with their stress status and left and right context.
2. *Syllable data*: currently containing all syllables in the corpus.
3. *Word data*: currently containing all words in the corpus.
4. *Sentence data*: currently containing all sentences in the corpus.
5. *Exception lexicon*: currently containing the transcription lookup table.
6. *Morpheme lexicon*: currently containing the morpheme table.
7. *Phoneme definition*: currently containing the phoneme table used for adaptation to language specific phoneme sets.

The unit selection algorithm searches these tables in all stages, but highest flexibility is reached using SQL-queries during pre-selection. For instance, to find the word “Guten” it would issue the command:

```
SELECT * FROM word_data WHERE TKey=”gu:t@n”
and PInt=”0”.
```

4. Conclusion and future research

In this paper, we presented BOSS, a general open-source architecture for the easy development of unit-selection-based speech synthesis systems. By a strict separation between data and algorithms it is easy to develop synthesis systems for different domains and languages. BOSS is by no means a finished system. The following issues should still be addressed in the future:

- The linguistic analysis module needs to be extended to handle words not present in the lookup table. It should also be extended with an algorithm that reliably computes accent and phrase boundary locations. At present, the phrase boundaries are simply derived from punctuation marks and accentuation is not taken into account. Only word stress is included in the XML document.
- In future, F_0 and duration parameters need to be predicted. In order to achieve maximal naturalness, unit selection should also be used for this prediction.
- The synthesis module needs to be extended with signal manipulation functionality. That would make it possible to smooth large pitch jumps or to create appropriate pitch contours that can not be found in the corpus.

5. References

[1] M. Beutnagel, A. Conkie, J. Schroeter, Y. Stylianou and A. Syrdal. The AT&T NextGen TTS System. *Proceedings of the Joint Meeting of ASA, EAA and DAGA, Berlin, Germany*, 1999.

[2] N. Campbell and A. Black. Prosody and the selection of source units for concatenative synthesis. In: J. van Santen, R. Sproat, J. Olive and J. Hirschberg (eds.), *Progress in Speech Synthesis*, 1997, p279-292. New York: Springer Verlag.

[3] E. Klabbers and J. van Santen. Predicting segmental durations for Dutch using the sums-of-products approach. *Proceedings of the 6th Int. Conf. on Spoken Language Processing, Beijing, China*, 2000, vol. III, p670-673.

[4] E. Klabbers and R. Veldhuis. Reducing Audible Spectral Discontinuities. *IEEE Transactions on Speech and Audio Processing*, vol. 9, nr. 1, January 2001, p39-51.

[5] T. Portele and K. Stöber. Domain-specific prominence-based concatenation. *Invited Talk, FORUM ACUSTICUM, Berlin, Germany*, 1999.

[6] K. Stöber, T. Portele, P. Wagner and W. Hess. Synthesis by word concatenation. *Proceedings of the 6th European Conf. on Speech Communication and Technology, Budapest, Hungary*, 1999, vol. II, p619-622.

[7] K. Stöber, P. Wagner, J. Helbig, S. Köster, D. Stall, M. Thomae, J. Blauert, W. Hess, R. Hoffmann and H. Mangold. Speech Synthesis by Multilevel Selection and Concatenation of Units from Large Speech Corpora. In: *W. Wahlster (ed.), Verbmobil: Foundations of Speech-to-Speech Translation*, Symbolic Computation, Springer, Berlin, 2000.

[8] P. Taylor, A. Black and R. Caley. The architecture of the Festival speech synthesis system. *Proceedings of the 3rd ESCA/COCOSDA Workshop on Speech Synthesis, Jenolan Caves, Australia*, 1998, p147-152.

[9] P. Taylor and A. Black. Speech synthesis using phonological structure matching. *Proceedings of the 6th European Conf. on Speech Communication and Technology, Budapest, Hungary*, 1999, vol. II, p623-626.