



A LEXICON-BASED GRAPHEME-TO-PHONEME CONVERSION SYSTEM

J.M.G. Lammens^{*}

ABSTRACT

There are two ways to do grapheme-to-phoneme conversion for a text-to-speech system: through rules or through lexicons and lookup strategies. Both methods have their pros and cons. For Dutch text-to-speech conversion, only rule systems have been developed so far. In this paper a new lexicon-based system is described. The different modules of the system and the algorithm used are discussed along with some preliminary results.

INTRODUCTION

In a text-to-speech system, grapheme-to-phoneme conversion is required to convert normal orthography into phonemic spelling that serves to drive the synthesis component. There are a number of different ways to go about this task, but essentially it boils down to two approaches: rule-based and lexicon-based. The former approach uses only rules for the conversion, often resembling the well-known SPE rules (ref. 1). These rules are formalisms that reflect linguistic knowledge, and use single graphemes and phonemes as their basic entities. The latter approach uses lexicons and lookup strategies. In these systems linguistic knowledge is not necessarily reflected, and the basic entities are larger, say morphemes or words. In practical applications one usually finds that mixed systems are used, containing both lexicon and rule components.

When comparing the performance of different systems one should consider correctness, completeness and speed as important features. In general, lexicon-based systems may function more correctly than their rule-based counterparts (certainly when entire words are stored along with their transcription), but they are also less complete, since not all words can be stored in a lexicon whereas rule systems in principle always produce an output. In theory, lexicon systems can be made to function 100% correct, but never 100% complete, as language is a non-stationary open-ended phenomenon. As far as speed is concerned, however, lexicon systems are usually the clear winners.

Over the last few years, a number of grapheme-to-phoneme conversion systems have been developed for Dutch, all of them principally rule-based (ref. 2,3). These systems are typically rather slow and do not produce 100% correct output. A lexicon-based conversion system is now being developed. The aim of the research is to compare the performance of such a system to that of existing systems on one hand, and to provide a fast buffer for rule systems on the other hand. It is also a matter of interest to find out how far one can get without rules, the more so since the author believes that the lexicon approach is in most cases the more psychologically realistic one. One cannot doubt the fact that speakers use some kind of inference mechanism to decide on the pronunciation of new or nonsense words (e.g. some kind of rule system or analogy-based reasoning), but it is conceivable that frequently used words are stored as a whole with their pronunciation in a kind of mental lexicon.

^{*}Institute of Phonetics, University of Utrecht, Trans 14, 3512 JK Utrecht, The Netherlands.

SPECIFICATIONS

The system that is being developed is modular in structure, in principle memory-resident and flexible in its applications. It operates in real-time. The basic conversion entities are words.

Let us have a look at the different modules first. There is a module for segmentation of input text into coherent strings. It operates on the orthographic structure of the text and returns words, punctuation marks, etc. Punctuation marks are processed in a separate module, and may be deleted or converted into special symbols as required. There is also a module that converts numerals into phonemic form. As numerals constitute an open-ended class, this is done by rules. Another module treats abbreviations, which are stored in a separate fast lexicon. They may be converted into phonemic form or expanded to normal words for further processing. Normal words are treated by the most important module of the system, the dictionary module.

The dictionary module itself consists of several parts. It is split up into a high-frequency (HF), main and user lexicon. The former contains relatively few words with a high frequency of occurrence, permitting those words to be looked up very fast. The main lexicon contains less frequent words. It is much bigger and therefore somewhat slower to search than the HF lexicon. An important issue regarding lexicons is what kind of entries to store. One can store words as they appear in texts (with inflectional endings etc.), or lemmas as they usually appear in a dictionary (only basic forms without inflection). The system being discussed uses both. Storing only lemmas requires less space, but makes some kind of a rule component necessary for deduction of inflected forms. Storing inflected wordforms too demands more space but allows faster processing. Another important question is how much information one should store along with the orthographic form of words. Apart from the obviously required phonemic form, one can also store syntactic, semantic or prosodic properties of words that are not required for the conversion as such, but may be of great value for other components of a text-to-speech system, e.g. syntactic or prosodic analysis. In this respect lexicon systems could have a clear advantage over rule systems. The size of the lexicon is naturally quite important too. The more words one stores, the larger will be the proportion of input texts that can be converted successfully. But as memory space is restricted in practical applications, and conversion speed tends to decrease with increasing lexicon size, there are limits. Finally, the organization of the lexicon (entry order, access methods, data compression) is another determining factor for overall system performance. High-frequency and main lexicon are fixed, meaning they cannot easily be altered by the user. Another lexicon part provides for the possibility of updating and expanding, viz. the user lexicon. This lexicon contains jargon, symbols etc. and may be altered interactively by the user, or may be provided from a library of application-specific user lexicons.

In a lexical component of a text-to-speech system one can also provide a decomposition component for processing of words that are not stored as such (ref. 4). In the present system, words are split up in parts that can be found in one of the lexicons. Those parts are subsequently converted and reassembled into a full phonemic wordform. The decomposed parts are not morphemes in the theoretical sense, but only substrings of words, possibly slightly altered to compensate for spelling rules and the like. The decomposition stops as soon as all parts have been found, or when further decomposition is impossible. This component is quite important to increase the number of converted words when using a relatively small lexicon or one consisting of lemmas only. The module works with a mixed lookup and rule strategy, which shows that also in a lexicon-based system some rules have to be incorporated. Finally, there is a

component that functions as an exception handler. Exceptions are defined here as words that cannot be converted by other modules of the system. It is in fact a set of grapheme-to-phoneme conversion rules. Note that this is the reverse situation of a rule-based system, where a (small) lexicon is used to store exceptions that cannot be handled by the rules.

THE ALGORITHM

We now turn to the algorithm that is used for the actual conversion of input words into their phonemic form. It may be symbolically represented as follows:

while not end of input text do:

- (1) read an input string from the input text and decide on its label (punctuation, numeral, abbreviation, lexical)
- (2) take the appropriate action, depending on the label:
 - (A) punctuation: delete or convert
 - (B) numeral: convert, using rules
 - (C) abbreviation: expand or convert
 - (D) lexical:
 - (a) lookup in HF dictionary, if not found:
 - (b) lookup in main dictionary, if not found:
 - (c) lookup in user dictionary, if not found:
 - (d) decompose,
lookup parts as normal lexical strings
if all parts found, reassemble converted parts into one phonemic string
if not all parts found:
 - (e) pass string to exception handler

The parsing and labeling of input text is achieved by means of a transition network. The string that results from this module is routed to the appropriate module for further processing. Punctuation may simply be deleted or converted to e.g. symbols that mark the beginning and end of a sentence, which is important for suprasegmental (prosodic) control of an utterance. It is relatively easy to produce a compact and flawless rule set for the conversion of numerals to their phonemic form. Since this rule set is not likely to require changes once it performs correctly, it will be implemented in some fast compiled form. Abbreviations may be converted or expanded to their full orthographic form and subsequently converted as normal words. The latter option can prevent double storage of phonemic forms for those words that already exist in their unabridged form in one of the lexicons. As far as the different lexicons (HF, main and user) are concerned, it is important to choose the right order to consult them. Obviously, one needs to know more about the frequency of occurrence of words to obtain optimal ordering. It will be clear that the kind of text being processed can exert a great influence on the frequency of words (a text on chemistry e.g. will show a different word frequency rank order than a text on poetry). It is illustrative for the importance of frequency data that a HF lexicon of 200 newspaper words captures some 65% of an ordinary newspaper text (ref. 5). Even if one does not store the most frequent words separately, since there are search techniques available with search times virtually independent of lexicon size (ref. 6), one should make certain that these words are present anyway in the lexicon to ensure good performance of the system. In practical applications it may prove faster to consult the user lexicon before the others, or even to merge all three lexicons. The importance of the decomposition module increases with de-

creasing lexicon size and with increasingly lemmatised form of the lexicon entries. Note that in contrast with some existing text-to-speech systems where a morphological analysis is performed before (morpheme-) lexicon consultation, e.g. MITalk (ref. 7), decomposition is delayed here until after the first dictionary lookup has failed. If the lookup strategy is fast enough, this may prove quicker than attempting decomposition first (which always involves rather complex rule sets). The rule set is of a somewhat different nature too, as the lexicon acts as a filter, leaving only certain kinds of compounds or inflected words for processing by the decomposition module. It follows that the size and nature of the decomposition rules depends on the size and nature of the lexicons. Wrong decompositions can in part be prevented by storing special attributes along with words in the lexicon, and in part are discarded by the failure of lookup operations on their decomposed parts. The grapheme-to-phoneme rules that are finally used in the exception handler need not be the same as the ones used in rule systems either, as the lexicon again acts as a filter. It is conceivable that fewer rules of a different nature are needed for the exception handler, if the exceptions share some systematic property (e.g. compounds of a certain kind), but no decisive data on this subject is available yet.

RESULTS

Only some preliminary results are available yet. Experiments have been carried out with two kinds of lexicons (or subsets thereof): one containing 70038 lemmata, the other containing 12116 wordforms selected on frequency of occurrence from a newspaper text corpus of over 2 million tokens.

When using the lemma-lexicon ranging in size from 100 to 17500 entries (selected by frequency), no HF lexicon, no decomposition module, and a piece of newspaper text (25530 words) as input, the percentage of transcribed words as a function of lexicon size soon reaches an asymptote. Once the lexicon size increases over 3000 entries (60.36% words transcribed), the results hardly improve any more (64.29% words transcribed with a 17500-entry lexicon).

When using the full 12116-entry wordform-lexicon (hashed organisation), no HF lexicon, no decomposition, and the same newspaper text as input, 85.66% of the text is transcribed correctly (at a mean speed of 261.93 word/s cpu time on a VAX 11/750 computer, including file I/O and a primitive version of the input parsing routine). The same test with another newspaper text of 15706 words and some prose of 4207 words yields 85.33% and 88.45% respectively (at 267.76 and 289.09 words/s).

These figures, although preliminary, indicate that relatively small lexicons suffice for transcription of large proportions of ordinary text at high speeds. The next step is to increase the performance further by means of the decomposition and other modules.

REFERENCES

1. N. Chomsky & M. Halle, *The Sound Pattern of English* (Harper and Row, NY, 1968).
2. E. Berendsen, S. Langeweg & H. van Leeuwen, *Proceedings COLING 86*, 612-615 (1986).
3. J. Kerkhoff, J. Wester & L. Boves, *Proc Inst Phon Nijmegen*, 60-69 (1984).
4. R. Carlson & B. Granström, *Proc ICASSP 86 Tokyo*, 2403-2406 (1986).
5. M. van den Broecke et al., *De Nieuwe Taalgids* 79(6), 31-36 (1986).
6. D. Knuth, *The Art of Computer Programming vol 3* (Addison-Wesley, Reading Mass. 1973)
7. J. Allen, *Proc IEEE* vol 64, 422-433 (1976).