Institute of Phonetic Sciences,
University of Amsterdam,
Proceedings 22 (1998), 115 - 124

PRE-PROCESSING INPUT TEXT: IMPROVING PRONUNCIATION FOR THE FLUENT DUTCH TEXT-TO-SPEECH SYNTHESISER

Rik Jansen [1], Arjan J. van Hessen [1] and Louis C.W. Pols

Abstract

To improve pronunciation of the Fluent Dutch Text-To-Speech Synthesiser, two pre-processors were built that try to detect problematic cases in input texts and solve these automatically if possible. One pre-processor examines the pronounceability of surnames and company names by checking whether their initial and final two-letter combinations can be handled by the grapheme-to-phoneme rules of the Fluency TTS system, and correcting those automatically when and if possible. Also, common disambiguous abbreviations are properly expanded. The second pre-processor tries to realise pronounceable forms for numbers that do not have a straightforward pronunciation. Structural and contextual information is used in an attempt to determine to what category a number belongs, and each number is expanded according to the pronunciation conventions of its category. It can be said that these pre-processors are a useful aid in offline pronounceability examination (for names) and improvement of performance at run-time (for numbers), although ambiguity and redundancy in the input text illustrate the need for semantic and syntactic parsing to approach human text interpretation skills.

1. Introduction

During the process of generating synthetic speech from text, one encounters various kinds of problems. One of the most serious problems in converting text to synthetic speech is that the standard written form of (any) language gives an imperfect and often distant rendition of the corresponding spoken forms. In a written text, there will often be occurrences of numerals, abbreviations, special symbols (such as %, @) et cetera. These form a serious obstruction to the generation of natural sounding speech. It is therefore necessary to pass the text through a pre-processing stage that converts all such occurrences to the appropriate pronounceable words, unless in particular cases it is better to remove them instead.

Any reasonable pre-processing module must of course perform some disambiguation of the input text that it is expanding: for example, the string *Fl. 2,50* (2.50 Dutch guilders) contains information that 2,50 is a money amount, and therefore is expanded differently from the way in which 2,50 (= 2.50) would be expanded. Also, the full stop in *Fl. 2,50* does not indicate a sentence ending, but an abbreviation

[1] Comsys International B.V., Zeist

of *gulden* (guilder). A simple substitution does not suffice here: *gulden* must be placed elsewhere in the string (between *2* and *50*).

Another main problem in generating speech from text is, obviously, obtaining the correct pronunciation of whole words. Finding the appropriate phoneme selection and lexical stress assignment is essential for the intelligibility and acceptability of a text-to-speech system. Orthography is not related very straightforward to pronunciation. In Dutch (as in any language), loan words occur that do not apply for the letter-to-sound rules that are used to obtain phonetic transcriptions, so the only way to pronounce them correctly is to include them in an exception dictionary (Syrdal, 1995). Often, the contents of these dictionaries is not limited to just loan words but may contain as many words as possible of the subject language too to improve overall performance.

Proper names provide the same difficulties as loan words. Pronunciation of names is a difficult task because they might stem from foreign origin or contain archaic spelling and therefore rarely occur in the synthesiser's dictionary. If they do, their stress pattern might differ from the lexical item. For instance, the Dutch word *goedkoop* (cheap) has primary stress on the second syllable, but the surname *Goedkoop* is pronounced with initial stress.

Maybe even more difficult than proper names are company names when stored in a clientele database of a company (for instance, a bank). These often contain ad hoc abbreviations that are not in the synthesiser's dictionary. For example: *Herv Gem Wijkraad Kerkv Amsterd-W.* requires a lot of inference and knowledge of the context to be expanded successfully to *Hervormde Gemeente Wijkraad Kerkvereniging Amsterdam-West.* This calls for a special dictionary containing those abbreviations, since they are not very likely to occur outside this kind of databases. However, one has to be alert not to include ambiguous abbreviations that would automatically change abbreviations to an incorrect expansion. For instance, the abbreviation *maatsch* can mean *maatschappelijk* (social) or *maatschappij* (society).

In an attempt to tackle the two problems illustrated above, the first author has built two pre-processors that deal with pronunciation difficulties. They both are designed to be used for the Fluent Dutch Text-To-Speech synthesiser developed by Arthur Dirksen (FLUENCY). The first pre-processor is a number solver that tries to identify the nature of complex numbers (numbers that should not be pronounced as integers) by examining their structure and context in the sentence. An attempt is made to categorise them as a date, time, telephone number, bank account number, money amount, ordinal number, fraction, or an area code. Once identified, they are expanded to an appropriate pronounceable form. This is done by inserting orthographic clues and phonetic phrasing cues.

The second pre-processor is designed to improve pronunciation of names and detect cases that would obstruct the pronunciation generation process. Checking the beginnings and endings of words turned out to be an effective way to locate words that could cause pronunciation problems. Words with letter combinations at word-initial or word-final position that conflict with Dutch pronunciation rules are marked, to indicate that they need to be examined closer. For example, African names like *Nkono* or *Mkumba* have initial consonants that serve as an entire syllable and therefore should be transcribed likewise.

The purpose of this pre-processor is to function as an aid in constructing a database of names and their correct pronunciations, analogous to the ONOMASTICA project (Gustafson, 1994). These names can be either surnames or company names, based on the way in which banks generally register their clientele. Having the disposal of such a database makes it possible for commercial institutions to automatise telephone dialogues with their customers by using text-to-speech technology. Since company names may contain abbreviations, these are expanded if possible to reduce the number

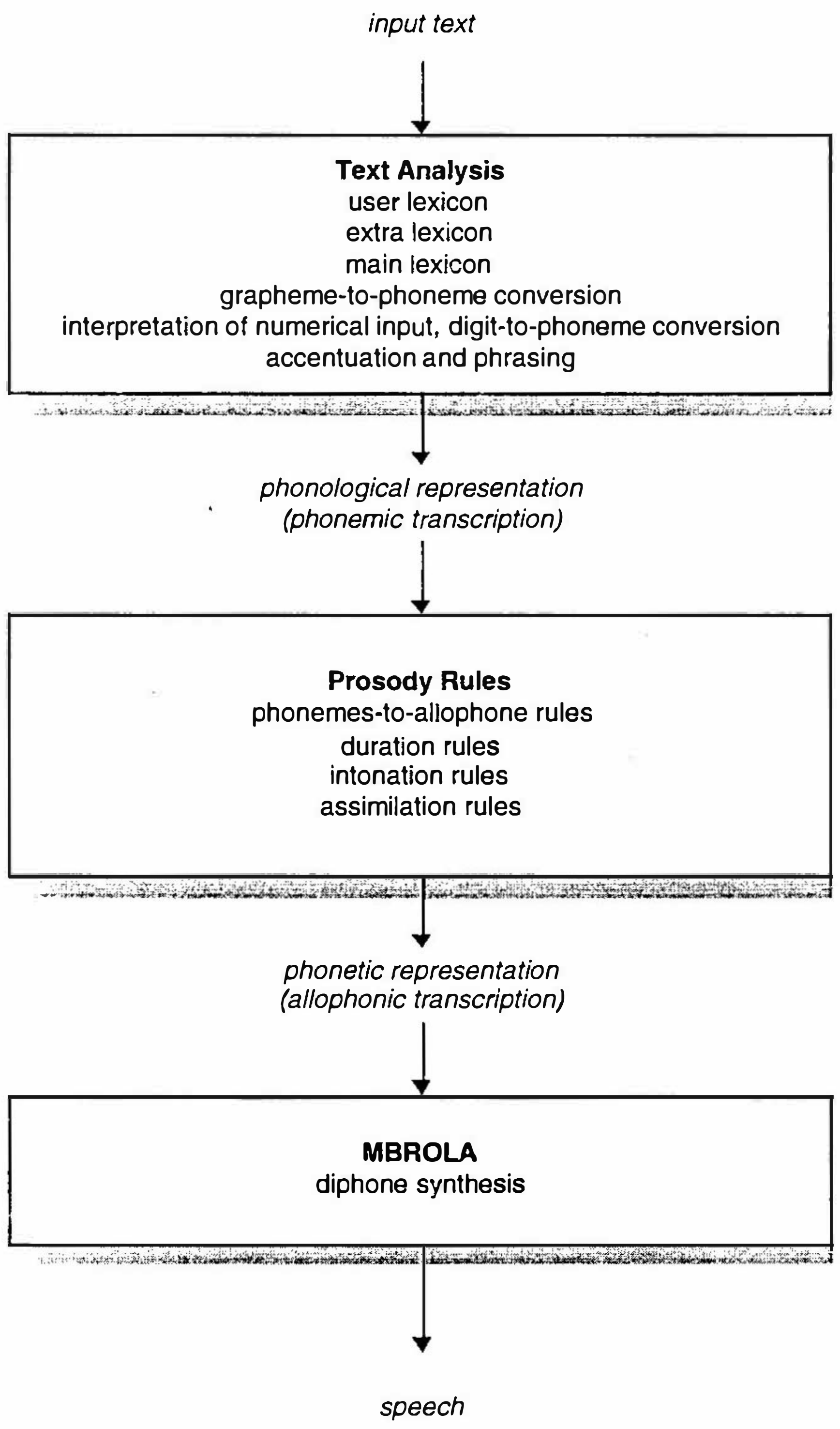
input text
Text Analysis
user lexicon
extra lexicon
main lexicon
grapheme-to-phoneme conversion
interpretation of numerical input, digit-to-phoneme conversion
accentuation and phrasing
phonological representation
(phonemic transcription)
Prosody Rules
phonemes-to-allophone rules
duration rules
intonation rules
assimilation rules
phonetic representation
(allophonic transcription)
MBROLA
diphone synthesis
speech

Figure 1: The Fluent Dutch Text-To-Speech Synthesiser

of pronunciations generated incorrectly. After the pre-processing stage, names that are assumed correct according to the initial and final letter combinations of their words are stored separately. They are still likely to contain small errors such as incorrect stress patterns or unexpanded abbreviations, and can be scanned for this rather quickly.

The names indicated as incorrect need a more thorough inspection. Most names will have to be manually rewritten to a phonetic transcription that can be included in the database to specify their pronunciation (for they are very likely to be foreign names).

2. The Fluent Dutch Text-To-Speech synthesiser

The Fluent Dutch Text-To-Speech synthesiser converts text to speech by using a Dutch diphone database and an MBROLA diphone synthesiser (MBROLA). An outline of its structure is shown in Figure 1. First, the text is analysed with the use of three lexicons. The main lexicon is the basic lexicon that contains many Dutch words and some common English words, specified by their phonemic transcription. The user lexicon and extra lexicon can contain uncommon words to make the synthesiser suitable for use in specific applications. Words that occur in one of the dictionaries are given the indicated transcription, and numerals, special characters and words that are not in the dictionary are converted to phonemic representations by using letter-to-sound rules. Also, pitch and phrase characteristics are determined. Text analysis transforms the text to a phonological representation containing information about phonemes and prosodic structure that can serve as input to the prosody rules. These convert the phonemic transcription to a phonetic transcription, consisting of allophones, specified by their duration and pitch points.

Based on this phonetic information, the MBROLA diphone synthesiser concatenates the appropriate diphones with their appropriate duration and pitch. This is done phrase by phrase, and produces the desired waveform.

3. Pre-processing words for letter combinations

To be able to inspect letter combinations at word-initial and word-final position, two tables were composed, one table containing initial combinations and one containing final combinations. The tables are 31 by 31 matrices in which the letters of the alphabet and the five most common ASCII characters (*ë, ê, é, è* and *ï*) are set out. In this way, the correctness of each possible letter combination at the beginning or ending of a word can be plotted in the corresponding cell of the appropriate table.

The tables were filled with letter combinations that are phonologically correct in Dutch. In addition, combinations that are pronounced correctly by the synthesiser were included, to account for alternative spelling forms of Dutch names that do not influence pronunciation. These might contain grammatical inaccuracies, for instance *-pd, -cd, -kd,* and *-hd.* Although grammatically incorrect, the synthesiser will not spell those sequences and regard the final *d* as a /t/. Since *-td* appears to be spelled out by the speech synthesiser, this combination is not permitted. If possible, other special cases will be rewritten to a form that produces the desired phoneme sequence. This is done before the table comparison takes place, to minimise the number of 'rejected' combinations.

To reduce the number of unjustly rejected words even more, the pre-processor uses extra lexicons that can be composed and customised by the user. This enables the processor to be used for various purposes. The function of these lexicons is to replace known problematic structures by pronounceable orthographic alternatives, thus improving pronunciation and reducing illegal letter combinations. There are three lexicons:

- a lexicon containing priority words
- a lexicon containing character substitutions
- a lexicon containing word substitutions

The lexicon containing priority words should contain the words that have to be marked 'correct' independent of the way they are spelled. These words may include company names that are included in the user lexicon of the speech synthesiser, and whose letter combinations would not pass the table check.

The character substitution lexicon contains punctuation marks and other special characters that should be substituted in the text because they would otherwise cause a correct word to be marked 'incorrect'. For example, consider the database entry *Comité "Stop Racisme"*. When the words *Stop* and *Racisme* would be checked for their initial and final letter combinations, the combination of the two symbols *"S* as well as *e"* would be marked as incorrect, since they do not occur in the tables. Whenever an entry of this lexicon is encountered in the text, it is replaced by the suggested substitution.

The word substitutions lexicon operates in the same way as the character substitution lexicon, the only difference being that this deals with words instead of characters. These are mostly abbreviations that have to be replaced by their fully written form or a standardised abbreviation that is present in the user lexicon of the speech synthesiser.

After substituting occurrences of lexical items with their desired alternatives, all words are ready to be checked for their initial and final letter combinations, except for the special cases. These special cases include:

- Lexicon words. These have been marked 'correct' in an earlier stage of the process.
- Single letters. These are automatically spelled, but would found to be incorrect in a table check.
- Numbers. These are not in the table, and therefore would not pass the table check.
- Abbreviations ending in a full stop. All full stops have been removed from the text at an earlier stage, so the full stops found belong to abbreviations that have been inserted during a user specific substitution routine. For example, expressions such as '*de heer*', '*dhr*', '*Hr*' et cetera all have been replaced by '*dhr.*', an abbreviation that is present (or that can be included) in the user lexicon of the speech synthesiser. Since these titles are all pronounced unstressed in this context, they are included in the lexicon without stress. This remarkably improves pronunciation of proper names preceded by a title.
- Standard abbreviations. These include common abbreviations used in company names such as '*BV*', '*NV*' et cetera.
- Words consisting entirely of capitals. These words are likely to be company names, and probably need to spelled out if not pronounceable (like 'KLM').

After the process of checking letter combinations, a rudimentary separation of pronounceable and problematic words has been realised. To verify this, a small test was carried out. A list containing 24,309 unique family names was processed in the way discussed above. Table 1 shows the results.

After replacing problematic character sequences and checking the letter combinations of those 24,309 names, 24,135 names were found to be correct, and only 174 names were considered to be incorrect. Although the words that have been indicated as being correct do not necessarily have to be correct, a check on a randomised subset containing 2,000 of the 24,135 'correct' names has learned that for 98% of those names, a pronunciation was generated without resorting to spelling mode. Although this pronunciation was not always perfect (mainly caused by an incorrect intonation pattern), it was understandable for the listener.

Table 1. Results of a test on 24,309 unique family names.

N = 24,309	amount	percentage completely pronounced of subset of words
judged correct	24,135	98
judged incorrect	174	33

Of the names that were classified as 'incorrect', 33% caused a spelled pronunciation. Moreover, the quality of the pronunciation of the names that were not spelled was poor compared to the names that had passed the test, mainly because the rejected letter combinations were foreign combinations pronounced in the Dutch way. Below, an example is given of several private and company names, and the judgement by the pre-processor. The names in the first column all have undergone a letter combinations check. The output results can be seen in the second column. Braces indicate incorrect words (the braces are on the side of the incorrect letter combination). Column 3 indicates if the item as a whole (i.e. the entire line), is correct according to the computer (C?) and column 4 indicates if this is a justified decision (J?).

Table 2. Examples of results of checking letter combinations. For more details, see text.

Checked Name	Output results	C?	J?
Hr Ayvas	Dhr. {Ayvas	N	N
Mw Baiwir	Mevr. Baiwir}	N	N
Jonas Bjorkman	Jonas {Bjorkman	N	Y
Hr Pieters	Dhr. Pieters	Y	Y
Dhr Pietersz	Dhr. Pieters	Y	Y
Knbrd	Knbrd	Y	Y
S Valckx	S Valcks	Y	Y
KLM personeelsver	KLM Personeelsvereniging	Y	Y
Mr van Hooff	Mr. van Hoof	Y	Y
admin der herv gem Haarlem	Administratie der Hervormde Gemeente Haarlem	Y	Y
afd Lemmer kon ned ver Rode Kruis	Afdeling Lemmer Koninklijke Nederlandse Vereniging Rode Kruis	Y	Y
A E Goedmann kunsts grafische en tekenmatr	A E Goedman Kunsts Grafische en Tekenmatr}	N	Y
adm hfdrek bvo min van alg zkn	Administratie Hoofdrekening {Bvo Ministerie van Algemeen Zaken	N	Y

Summarising, the pre-processor for letter combinations seems a useful timesaving device. The expanding of standard abbreviations can be a time-consuming activity, especially when large databases have to be processed. Moreover, a lot of common letter combinations in names that are problematic for the speech synthesiser are solved automatically too. Results of pronunciation performance tests indicated that names that were judged to be correct were pronounced considerably better than names judged to be incorrect. To produce pronunciation results that are desired in customer-computer communication, every entry in the company's clientele database should be manually checked. However, if this is not possible due to lack of budget or time, the program can be relied on to filter out the problematic names and the names that are considered correct will be pronounced understandable most of the time.

4. Improving the performance of the Fluent Dutch Text-To-Speech Synthesiser by pre-processing numbers

The pronunciation of numbers by the Fluency Dutch Text-To-Speech Synthesiser (TTS) is a process that very often goes wrong. The synthesiser interprets any number as if it were an integer, which leads to a very unorthodox way of pronouncing common digit sequences like telephone numbers, area codes, bank account numbers et cetera. The TTS itself accounts for a few special cases, such as time and date, but this only works when written in the exact format (e.g., a time is read for *12:30*, but not for *12.30* or *12:30:23*). Minor deviations in spelling will cause the TTS to resort to spelling mode.

The number solver tries to solve this problem by examining if the numbers found in the text can be placed into categories that have equal pronunciation.

First, the text is scanned for numbers (in this case, a number is considered any sequence of characters that contains at least one digit). If a number is found, it undergoes a series of tests that try to determine what category the number belongs to. The different categories include:

1. area code
2. ordinal number
3. national telephone number
4. international telephone number
5. amount of money
6. time
7. date
8. fraction
9. bank account number
10. combination
11. spelled number

A crucial element in successfully tagging the text is to allow for variability in structure of the number. For instance, a number may be placed between brackets, occur at the end of a phrase or sentence, or contain white spaces. When a number contains white spaces, it is difficult to determine where a number stops and the next one begins. This can be seen in expressions like *I have dialled 06-54645433 3 times.* Therefore, airtight number identification requires syntactic parsing of some form. The pre-processor discussed here does not use syntactic parsing, but structure analysis includes contemplating the possibility that a number exceeds word boundaries (for example, *020 6 454 828* will be recognised as a Dutch telephone number).

It should be noted that, since the pre-processor was designed for a Dutch TTS system, the number identification is based on Dutch number structures. Since each country has its own writing conventions, it would be an impossible task to cover for all those different styles. To illustrate this, *06/05/98* is interpreted as May 6th, 1998 in Dutch, whereas it means June 5th, 1998 in English. In this system, Dutch conventions have priority over the English conventions, except when this would lead to meaningless structures, say, a date like *5/14/98*.

If a number meets the criteria of one of the categories listed, a tag with the category's name is attached to it. Only one tag per number can be given, and overruling is impossible at this level. The examination process walks through the categories in the order shown above, so for instance if a number is found to be a date, it can not be tagged as a bank account number anymore. The only exceptions to this are fractions and spelled numbers. Fractions can be altered to dates if there is contextual evidence for this (for example *d.d. 6/10*), and any category can be changed to a spelled number if the context contains clues for this. However, since the tagging criteria show virtually no overlap, it is most likely for the tag to be correct when a number is tagged.

The second step in the process is the investigation of the contextual environment of the number. Whereas in the first step the only concern was the number itself, now it is being looked at as part of a sentence. In the event of letters occurring at word-initial or word-final position, they are peeled off, and the remainder of the number is being checked for categories again (which could even cover for typing mistakes!). If, after this process, still no tag has been found, the immediate surroundings of the number in the sentence are searched. If a word, occurring within a distance of two words from the number, could indicate that the number belongs to a certain category, and the number satisfies the minimal conditions for that category (for example, an *area code* number should be at least four digits long), the tag is given.

To scan the context words of a number, each category has its own dictionary containing characteristic words for that category. For example, *bellen* (call) is one of the words of the telephone dictionary. The dictionaries also contain information about whether the context words should precede or follow the number in order to be assigned to that number.

The dictionary that contains the most matches is assumed to be of the correct category. Also, each category has its own, predefined priority number. If two or more dictionaries find an equal number of matches, the category that has the highest priority number prevails. Priority is based on three considerations:

- given a lexical item of a specific category matching a context word, what is the probability that the number belongs to that category?
- what is the probability that a number of a particular category will be recognised from context but not from structure analysis?
- suppose the wrong tag would be assigned, what would be the consequence for the eventual pronunciation of the number?

Spelled numbers get highest priority, since the presence of dictionary items for this category offers a fair chance that indeed a spelled number is present, and, more importantly, the number could not have been recognised earlier during the process. Furthermore, the rewriting procedure for pronunciation of a spelled number allows for any character combination without risk of losing information. Bank account numbers and telephone numbers are next, mainly because giro numbers and local telephone numbers contain no information and cannot be categorised in the initial tagging routine (but, on the other hand, would be understood if classified as a spelled number). Area codes are positioned at the bottom of the list since their structure is so well defined that they should be categorised straight away in the initial tagging routine. Date and time also receive low priority since their phonetic transcription routine is disastrous to numbers unjustly categorised this way.

The final step is to expand the tagged numbers to produce an input text for the TTS. Numbers that have not been tagged are assumed to be normal integers, and since the TTS pronounces them correctly, there is no need for further processing. The program scans through the text looking for a tag, and if one is found, the tagged number is cut out of the sentence, the tag is disposed of and the number undergoes the expanding process appropriate for the tag.

This expanding process can take place at two levels: phonemic or orthographic. In the first case, the rewritten number is built up from small pieces of phonetically transcribed text, added as the process goes along. This is done to allow phonetic cues to be inserted in the text more easily, or to give the syllables the correct stress. For example, expanding ordinal numbers such as *12301ste* can be done by first obtaining the phonemic transcription of the integer and next change the ending / `?*en-st@` / to /`*er-st@` /.

In the second case, numbers are rewritten orthographically. In this way, the program can make use of the ability of the TTS to pronounce certain strict 'formats', for instance the time format as shown above. By converting the number to the TTS format, with optional supplementary information, it can be transcribed as a whole.

In this way, a sentence at orthographic level is created that contains the entire text, constructed of sentences combined with expanded numbers and phonemic insertion tags.

The *Number Solver* creates alternative pronunciation directives that result in smoother pronunciation of most of the numbers. It is, however, not possible to capture all numbers into their appropriate categories, and sometimes numbers end up with a correct tag (through the context search) but still are pronounced incorrectly (for example, when a number is tagged during the context search while the structure can not be correctly rewritten in the expanding routine).

This problem can be overcome in two ways. The expanding routine can be refined to produce correct pronunciations for more different structures, or the tags should be redefined more accurately. This would involve defining more categories, and, for instance, the implementation of expanding directives within tags. Expanding directives are already successfully implemented in the SABLE SAYAS markup language (Sproat, 1998) (that uses a similar tagging system to indicate pronunciation for numbers) and they mainly include word order directives (for example, the *date* tag can be refined with a Day-Month-Year mode type option, that indicates the structure of the date as, say, MDY or MD or DMY).

5. Conclusion

Pre-processing texts before feeding them to a Text-To-Speech synthesiser considerably improves pronunciation and can create the illusion of computer intelligence. Nevertheless, it does not account for all the problem cases that one encounters in texts. The pre-processor for names is useful as a timesaving device that successfully solves some basic problems in name pronunciation and abbreviation expanding. However, names are too unpredictable to assume that if this pre-processor indicates a name as being correct, it will be pronounced correctly. The only way to be sure that TTS systems generate correct name pronunciations is to design the pre-processing stage with the use of the same syntactic knowledge that humans use when generating a pronunciation. This implies including an ability to identify different languages and their language-specific variation such as archaic or illogical linguistic usage. Depending on what language is most likely to have been used, letter-to-sound rules for that specific language will determine eventual pronunciation.

When the most common structures of numbers are known to a pre-processor, a fair deal of them can be identified correctly. Performing a context scan can improve identification even more. What causes most of the problems that occur in number identification is the redundancy that is often used in text writing. Whereas humans usually have no problems in inferring the meaning of character strings, subconsciously using syntactic and semantic context information, computers need more information to determine the identity of a number. The pre-processor that is discussed in this master thesis (Jansen, 1998) mainly relies on the use of clear structures, and can cover for some redundancy by doing a context search that tries to suggest semantic parsing. When texts are written without a lot of redundancy, most numbers are identified correctly.

Summarising, it can be stated that the two pre-processors that were discussed here perform as they were originally intended to do. However, considering the results produced by both pre-processors, it seems safe to conclude that although the majority of cases that form an obstruction to the pronunciation generation process can be solved successfully, there seems to be a limit in pre-processing performance that cannot be exceeded unless some form of semantic and syntactic parsing is used to correctly identify all words in the text, in order to provide an appropriate pronunciation for them.

REFERENCES

Gustafson, J. (**1994**). "ONOMASTICA - Creating a multi-lingual dictionary of European names", Fonetik'94, Papers from the 8th Swedish Phonetics Conference, Lund, Sweden, 66-69.

Jansen, R. (**1998**). "Pre-processing input text: Improving pronunciation for the Fluent Dutch Text-To-Speech Synthesiser", Master Thesis, Institute of Phonetic Sciences, Amsterdam and Comsys International B.V., Zeist, 62 pp.

Sproat, R., Hunt, A., Ostendorf, M., Taylor, P., Black, A., Lenzo, K. & Edgington, M. (**1998**). "Sable: A standard for TTS Markup", Proceedings International Conference on Spoken Language Processing (ICSLP'98), Sydney, Australia, Vol. 5, 1719-1722.

Syrdal, A.K. (**1995**). "Text-to-Speech Systems", In: A.K. Syrdal, R. Bennett & S. Greenspan (Eds.), Applied speech technology, CRC Press, London, 99-126.

FLUENCY: http://www.fluency.nl

MBROLA: http://tcts.fpms.ac.be/synthesis/mbrola.html