Word Class Assignment and Syntactic Analysis for Text–to–Speech Conversion

Rijk Willemse

November 1992

(The author wishes to thank Lou Boves for his useful comments and suggestions.)

1 Computational Linguistics in Practice

There is a gap between scientific computational linguistics and linguistic knowledge expressed and implemented in a real–time system for text–to–speech (TTS) conversion. This is the gap between theory and practice. The real–time requirements of a TTS system do not allow for the application of non–deterministic techniques for syntactic and morphological analysis: all input for a TTS system — even corrupt input — must be processed in such a way that only one linguistic analysis is produced, in real–time. Scientific computational linguistic methods very seldomly generate only one analysis. In most cases a large number of analyses is produced. In other cases no analysis is produced at all, because the formal rules do not describe the linguistic material that was input. Therefore, probabilistic and deterministic computational linguistic algorithms are used in a TTS system.

Linguistic knowledge is necessary in a TTS system in order to improve the prosodic quality of the speech generated. Syntactic analysis provides information that feeds into a module for calculating a natural sounding prosody. Morphological analysis provides information that feeds into a module for grapheme–to–phoneme conversion: the syntactic category of a word is, in

some cases, necessary to determine its correct pronunciation.

A probabilistic or a deterministic approach may lead to occasional errors — e.g. an erroneous word class, or an unwanted syntactic structure — that affect the prosodic quality of the speech generated. However, this must be preferred to a situation where no speech is generated at all, as a consequence of intensive and time–consuming calculations.

Accepting the possibility that errors occur is not a way of justifying a sloppy approach of the linguistic problems that must be solved during text–to–speech conversion. It is, on the contrary, the best approach obtainable when considering silence generated by a TTS system, instead of speech, as a severe error. Moreover, it must be accepted that natural language is such a complex and irregular system that errors inevitably occur when trying to apply computational linguistic methods in order to convert unrestricted text to natural sounding speech. The errors made by a human speaker when reading aloud, 'a prima vista', are a good illustration of the complexity of the linguistic problems encountered during text–to–speech conversion.

The gap between scientific computational linguistics and the linguistic knowledge used in a TTS system is also illustrated by the fact that linguistic knowledge used in a TTS system has been especially tuned to the problems that are encountered in this environment. For that reason, special purpose rules and probabilities are needed, instead of knowledge that is theoretically oriented. This implies, for example, that the surface grammar used in a TTS system must be tuned to two requirements: it must be parsable in real–time, and it must express exactly the knowledge needed for the calculation of a natural prosody. A surface grammar for Dutch as described in Van Bakel (1984), for example, is far too powerful for the job.

Despite the gap observed between scientific computational linguistics and linguistic knowledge used in real–time systems, there is an unbreakable bond between the two. All applied computational linguistics is inspired by the theoretically oriented computational linguistics, and (hopefully) vice versa. It would not have been possible, for example, to implement a special purpose surface grammar without the extensive research on this topic.

In this paper we will illustrate the purpose specific use of computational linguistic techniques by showing how we solved some of the linguistic problems that are encountered in a TTS system. The accent will be put on word class assignment and on syntactic parsing: both necessary for an acceptable prosody on sentence level. The linguistic knowledge implemented for word class assignment and syntactic parsing has been inspired by computational linguistics, but it has been tuned to the requirements of the TTS system.

2 The POLYGLOT Architecture

We have developed and implemented a multi–lingual, language–independent and modular architecture for assigning, in real–time, syntactic classes to words and syntactic structures to sentences. This implies that all language–dependent knowledge is kept strictly separated from the source code that constitutes the TTS system. Rules in easy–to–read formats, in combination with compilers for these rules, are used to express the linguistic knowledge necessary for the process of TTS conversion. This makes the system easy to maintain and develop.

The module for assigning word classes and syntactic structure is a part of the TTS system developed during the ESPRIT–project POLYGLOT (Vittorelli et al. 1990), and it consists of three parts. The first part assigns all possible word classes to the input words, based on a 16 k words lexicon, on deterministic morphological decomposition, and on rule–based Word Class Assignment (WCLA). The second part reduces the number of words in the word class lattice by means of a Hidden Markov Modeling (HMM) constrained by Viterbi Beam Search. The third part consists of a so–called Wild Card Parser (WCP), a deterministic parser that assigns phrase structures to input sentences. The sequence of word classes and the syntactic structure that is produced, is used for the calculation of a natural sounding prosody: this process is not described here, however.

3 Word Class Assignment

WCLA in a TTS system is necessary for correct grapheme–to–phoneme conversion. The pronunciation of various words depends on their syntactic category. Furthermore, WCLA is necessary as a precursor for syntactic analysis. On the basis of syntactic structures assigned to sentences, natural sounding intonation contours can be calculated.

In the majority of the cases, the syntactic categories of the words can be found in the lexicon of the TTS system. However, since the words of a language do not form a closed set (especially in languages like German and Dutch that can freely form new compounds), a lexicon will always be incomplete and thus must be supplemented by modules for morphological decomposition (Gulikers & Willemse 1992), and rule–based WCLA (Willemse & Gulikers 1992).

Moreover, most words can have several word classes. Thus, WCLA requires statistical and syntactical evaluation of a word class lattice in order to find the most likely syntactic category of each word. No ambiguity can be allowed here, since the sentence can only be pronounced in one way.

The WCLA module of a TTS system takes a sequence of words as input. The lexicon of the TTS system assigns one or more word classes with corresponding probability values to each word. Some words cannot be found in the lexicon, however. For these words, a set of rules operates that assign one or more word classes and corresponding probabilities to each word that was not found in the lexicon. The output, which feeds into the WCP module, consists of a complete word class lattice: i.e. a sequence of words with one or more word classes associated with each word. Basically, the WCLA module consists of two parts:

1. a set of rules that generate word classes for the input words that have not yet been assigned word classes (described in Section 3.1);

2. an algorithm that estimates the optimal sequence of word classes (described in Section 3.2).

3.1 Word Class Assignment by rule

There are several types of rules for generating word classes. The rules of the WCLA module are used to assign word classes to words that did not receive word classes during lexicon look–up. The rules act on the basis of:

- the form of the words: morphological rules;
- the position of the words in the sentence: context rules; these rules require that some words of the sentence already posses one or more word classes;
- a combination of form–based rules, and position–based rules: a rule can consist of several parts that address different sorts of knowledge.

The flow of control in this part of the Word Class Assignment module is as follows (1). For each sentence:

$$\odot_{i=1}^{T} \left(\odot_{j=1}^{R} \left(apply(rule_j,\ word_i) \right) \right) \tag{1}$$

Where:

- $\odot_{x=1}^{Y}$ indicates a loop: $for\ (x = 1;\ x \leq Y;\ x{+}{+})$;
- T indicates the total number of words in the current sentence;
- R indicates the total number of rules in the rule set;
- *apply* indicates the evaluation of a rule; when this rule succeeds, a word class is assigned to the current word.

Thus: for all words in each input sentence the rules are applied in a crucial order. There are two sorts of rules: ambiguous rules and unambiguous rules. Depending on the sort of a rule, two things can happen after it has applied successfully:

- When the rule is ambiguous, the other rules are applied to this word, possibly causing more word classes to be assigned to the word.
- When the rule is unambiguous, all other rules are skipped, and the following word of the sentence is considered.

When all rules have been applied to all words of the sentence, the chance exists that the set of word classes corresponding to some words of the input sentence is still empty. For that reason, a default rule is applied when all other rules have been tried. This rule assigns a set of default word classes to words that do not yet possess word classes.

3.2 Word Class Assignment by means of HMM

The second part of the WCLA module consists of an algorithm that tries to find the optimal path through the word class lattice.

A word class lattice can be modelled by means of a Hidden Markov Model (HMM); c.f. Cox (1988) and Holmes (1988) for basic notions of HMM, and Merialdo (1991) for an application. The words of a sentence correspond to an ordered sequence of word classes. The observations of the HMM are represented by the words (w_t), and the states of the model are represented by the word classes (c_i). At a certain time t, the model occupies one of the states c_i while an observation w_t is made. At $t + 1$, the model moves to a new state — or stays in the same state — and another observation is encountered. Since, in general, an observation (a word) can correspond to several states (word classes), the states can be said to be hidden: the possibility exists that a sequence of observations is produced by more than one sequence of states, as is the case in the well–known example "Time flies like an arrow".

The HMM for modeling word class lattices consists of the following components:

- a set of S word classes, the states;
- a set of V different words, the observations;
- a transition probability matrix of size $S \times S$
- an emission probability matrix of size $(S - 2) \times V$

For each word of the sentence, one word class can be said to be the most likely to correspond to that word. The likelihood of a word class at time instant t depends (a.o.) on the previous word class, at time instant $t - 1$,

and on the joint probability of the word / word class combination. The HMM which we will use to model word class lattices consists of S states:

- the initial state B
- the final state E
- the word classes $c_1, c_2, c_3, ..., c_{S-3}, c_{S-2}$

Each state, except the initial state, can be preceded by all states (inclusive of the state proper), except by the final state. The initial state may be followed by any other state, and the final state may be preceded by any other state. This results in the topology represented in Figure 1.

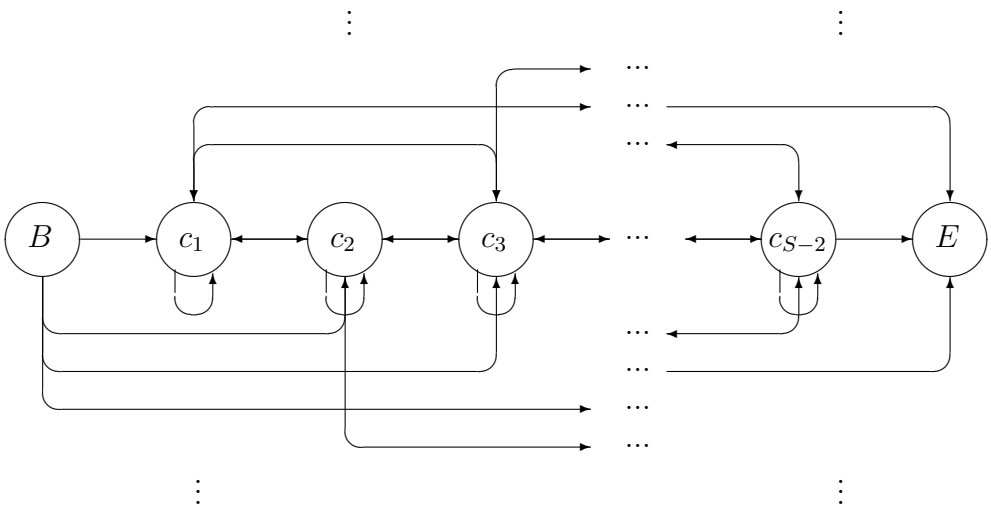

Figure 1: The states of a HMM which can be used for modelling a word class lattice.

The transition probability matrix of the HMM is used to assign probabilities to the arcs in Figure 1: the likelihood of moving from one state to any other state is tabulated here. In the emission probability matrix two types of probabilities can be tabulated, in principle. The joint probability of a word in combination with a certain word class, or the joint probability of a word

in combination with a transition of two word classes. In the HMM which will be applied here, the joint probability of a word in combination with a certain word class will be used. Future tuning of the HMM may make the second option more attractive.

Initially, the transition probability matrix and the emission probability matrix are estimated on the basis of a relatively large tagged corpus: i.e. a training text consisting of Z word / word class pairs. The transition probabilities of the word classes, a_{ij}, are obtained according to Equation 2.

$$a_{ij} \ = \ \frac{Probability\ of\ being\ in\ state\ c_i\ and\ making\ a\ transition\ to\ state\ c_j}{Probability\ of\ being\ in\ state\ c_i} \tag{2}$$

The estimation of the emission probability of a word / word class combination, $b_j(w_t)$, is obtained according to Equation 3.

$$b_j(w_t) \ = \ \frac{Probability\ of\ being\ in\ state\ c_j\ and\ observing\ word\ w_t}{Probability\ of\ being\ in\ state\ c_j} \tag{3}$$

Finding the most likely sequence of states given a sequence of observations — i.e. finding the most likely sequence of word classes corresponding to a sequence of words — can be done (a.o.) by means of Viterbi search: the most likely sequence of states is found by computing Equation 4. The optimal path ends at the word class corresponding to the last word of the sentence for which ϕ is maximal in comparison with ϕ for the other word classes corresponding to this word. Note that the search space that originally consists of S^T possible paths through the word class lattice is reduced to $S \times T$ when Viterbi search is used.

$$P \ = \ max_{j=1}^{S_T}\, \phi_T(j) \tag{4}$$

where $max_{j=1}^{S_T}$ indicates the maximum value of the argument in the range $1, ..., S_T$, i.e. the maximum value of ϕ selected from the word classes $1, ..., S_T$ at word w_T

Here, T equals the number of words of the sentenceInclusive of two empty words, that precede and follow the sentence in order to allow for processing

the begin state and the end state., and ϕ for word class j at a certain word (i.e. at time $t+1$) is recursively defined as:

$$\phi_{t+1}(j) \;=\; max_{i=1}^{S_t} \left[\phi_t(i) a_{ij}\right] b_j(w_{t+1}) \tag{5}$$

where:

- $t = 1, 2, ..., T-1$
- where $max_{i=1}^{S_t}$ indicates the maximum value of the argument in the range $1, ..., S_t$
- S_t equals the number of word classes corresponding to word w_t
- a_{ij} equals the probability of being at word class c_j at time $t+1$, and having made the transition from word class c_i at time t
- $b_j(w_{t+1})$ equals the joint probability of being at word class c_j, and observing word w_j, at time $t+1$

The transition probability matrix will always contain zero entries because certain transitions of word classes did not occur during the training phase. When using the HMM as a recognizer, looking for the optimal path through the word class lattice, zero transition probabilities will always cause the sequence of word classes they are a part of to be discarded (ϕ will become zero). Since this is an unwanted effect of zero entries, a very small value (ϵ) is used instead.

The emission probability matrix will contain zero entries also: word / word class combinations that did not occur during the training phase. These zero emission probabilities will set ϕ to zero when a word is encountered that is not contained in the lexicon. For such words, the WCLA rules generated a set of word classes, but the emission probabilities equal zero. In order to provide a way around the problems that are related to these zero probabilities, a method inspired by Good (1953), Gale & Church (1990) is used.

A basic notion of this method is the use of frequencies of word / word class pairs. Let $N_r(c_i)$ indicate the number of word / word class pairs that occur r times, where $r \geq 1$, and where the word class is of class c_i. For example,

$N_3(Noun)$ denotes the number of times that different word / '$Noun$' pairs occurred three times in the training corpus (of size Z, containing V different words). Let $b_i(*)$ denote the emission probability of an unknown word w_* in combination with word class c_i. We now assume that a zero emission probability of an unknown word corresponding to word class c_i equals the probability of the word / word class pairs $< w, c_i >$ that occurred only once in the training corpus: c.f. Equation 6.

$$b_i(*) \;=\; \frac{N_1(c_i)}{Z} \tag{6}$$

In Figure 2 an example of a word class lattice is shown after lexicon look–up and — for all words that could not be found in the lexicon — WCLA by rule. For the word classes corresponding to the words found in the lexicon, the emission probability has been determined during the training phase of the model (c.f. Equation 3). For the word classes corresponding to the words not found in the lexicon — for which the WCLA rules applied — an approximation of the emission probability has been made by using Equation 6.

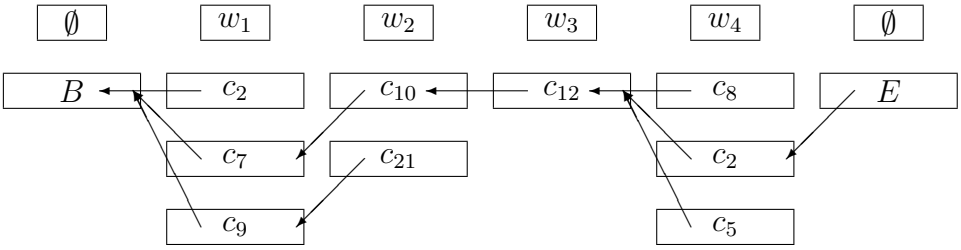

Figure 2: A word class lattice after lexicon look–up and WCLA by rule preceded and followed by dummy words. ϕ at each word class has been calculated and the maximum ϕ of the word classes corresponding to a word is indicated by an arrow (indicating the best previous word class). The probability values are not indicated.

Calculating Equation 4 for the final state E produces the sequence of states via trace back: $Bc_7c_{10}c_{12}c_2E$ corresponding to the sequence of observations $\emptyset w_1 w_2 w_3 w_4 \emptyset$.

The Viterbi search has been constrained with a Beam search which is defined as in Equation 7.

$$Beam \ = \ max_{b=1}^{S_T} \ \phi_T^{'}(j) \tag{7}$$

Where $\phi^{'}$ at a certain word is defined as a constrained version of Equation 5:

$$\phi_{t+1}^{'}(j) \ = \ \begin{cases} \text{undefined} & \text{if } \phi_{t+1}(j) < threshold_j \\ \phi_{t+1}(j) & \text{otherwise} \end{cases} \tag{8}$$

Where:

- $t = 1, 2, ..., T-1$
- $\phi_{t+1}(j)$ is defined as in Equation 5
- $threshold_j = max_{j=1}^{S_j} \phi_{t+1}(j) * thresholdfactor$
- undefined entities are removed from the word class lattice during processing

Since the possibility exists that the optimal sequence of word classes generated by the HMM approach contains errors — due to insufficient training of the model, and due to long distance syntactic phenomena (Derouault & Merialdo 1986) — several sub–optimal paths are output also. In this way, a number of state sequences are produced that correspond to the sequence of observations: a set of most probable sequences of word classes that correspond to the sequence of words. The module that processes the output of the HMM — the module for syntactic analysis — is suitable for evaluating these sequences of word classes, and producing a unique sequence of word classes.

4 Syntactic Analysis

After the WCLA module has applied, each word class in the word class lattice possesses a probability. The word classes are ordered according to this probability: the word class with the highest probability, i.e. the word class belonging to the optimal sequence of word classes, becomes the first member

of the ordered set corresponding to a word, the word class with the second highest probability becomes the second member, and so on. The word class lattice produced by the WCLA module is processed by the module for syntactic analysis, the Wild Card Parser (WCP).

The grammar of the WCP is designed for two purposes: disambiguating the word class lattice and generating syntactic structures that allow for calculation of prosody. It has not been designed for a maximal coverage of syntactic structures that can be found in natural languages (as is the case with Van Bakel 1984): the WCP applies a strategy for partial parsing when no analysis can be found for a sentence. The approach of syntaxis via partial parsing techniques has proved to be efficient in terms of processing time and to be robust when corrupt input is encountered (Hayes & Mouradian 1981, Ward et al. 1988).

4.1 Wild Card Parsing

WCP is a new application of context free grammars: it detects the set of maximum expansions of syntactic constituents — described in a context free grammar — that covers the maximum part of the input sentences, and it selects a set of word classes from the input word class lattice.

WCP provides one parse for each input sentence. In some cases, a parse contains Wild Cards, words that cannot be parsed as belonging to a constituent. This enables WCP to deal with unrestricted text, even if it contains non-grammatical constructions. Moreover, WCP provides a way around structural ambiguity. Where backtrack parsers tend to produce more than one analysis for a sentence, WCP provides only one parse: the first set of context free rules of the grammar that succeeds, is applied. By ordering the alternatives of the context free rules, preferences as to the application of the rules can be expressed.

The input of the WCP consists of a word class lattice; the result is the application of the WCP is a labelled bracketing. This bracketing represents the structure that the parser assigns to the input, according to the grammar. The output function of the parser selects two types of information from the bracketing:

- structural information needed for the calculation of the prosody for a sentence: the module that derives prosody from syntactic information transforms major constituents into intonational phrases, and assigns pauses and sentence accents on the basis of the syntactic structure.
- a word class lattice that has been reduced considerably as compared to the input word class lattice: one word class per word is output.

4.2 The Grammar

A WCP grammar is an ordinary context free grammar which has been tuned to describe sentences in a relatively superficial way, only producing structural information on syntactic categories without going into details as to semantics: due to the limited number of word classes (16) used in the grammar, a very limited amount of lexical and semantic information is expressed. The application of the context free rules is subject to three conditions, however:

- the rules apply deterministically
- a special WCP search strategy is used during parsing
- when no analysis can be made according to the context free rules, a minimum number of Wild Cards is allowed in the parse.

The alternatives of the context free rules are evaluated from top to bottom: when an alternative of a rule succeeds, the remaining ones are skipped. Ordering the alternatives appears to be a very powerful means to express linguistic intuitions. It should be realized, of course, that the use of a deterministic parser will, by necessity, result in occasional errors where ambiguity exists as to neighbouring syntactic categories. The example (Figure 3 rule 1, 2 and 3) demonstrates the kind of problems caused by ambiguities at borders of consecutive categories.

The ambiguity is due to the fact that all alternatives of rule 2 end with 'A', and that the right hand side of rule 3 begins with 'A', while — in the first rule — 'b' is followed by 'c'. When the string 'AAB', for example, is input to the WCP generated according to these rules, no parse can be produced. The correct analyses of the string 'AAB' would consist in the application of the second alternative of rule 2, followed by rule 3. However, since the first

```
(rule 1.)   a   :   b, c.
(rule 2.)   b   :   AA;
                    A.
(rule 3.)   c   :   AB.
```

Figure 3: A rule set that may cause problems for the WCP

alternative of rule 2 matches the first two characters of the input string, its second alternative will never be tried. The first member of rule 1 (i.e. 'b') succeeds and the cursor is moved two positions to the right: the rules have to apply beginning from this position. None of the rules of the grammar will match, causing the parser to generate a Wild Card: no backtracking takes place to the second alternative of rule 2. Reversing the order of the alternatives of rule 2 is no remedy. It will result in a parse for the input string 'AAB', but similar problems will arise when 'AAAB' is offered to the parser. It is our experience, however, that this drawback is by far outweighed by the advantage of a drastic reduction of the number of possible parses.

The heart of the WCP grammar is the so–called '*s_cat* rule'. The syntactic constituents that the parser looks for, are defined in the grammar by this rule. The Dutch *s_cat* rule is represented in Figure 4.

```
s_cat   :   sentence;
            verb_phrase;
            sub_ordinate_sentence;
            noun_phrase.
```

Figure 4: The Dutch rule for detection of constituents

By means of WCP, a search is performed for the set of maximum expansions of the syntactic category *s_cat* that covers the maximum part of the input. For the Dutch grammar this implies that the parser searches, in the first place, for a maximum expansion of the constituent *sentence*. When no *sentence* — as defined in the grammar — can be found in the lattice of word

classes representing the input sentence, the parser looks for a maximum expansion of the constituent *verb_phrase.* And when that fails, the parser looks for a maximum expansion of the next constituent, until all alternatives of the *s_cat* rule are exhausted. When all alternatives of the *s_cat* rule have been tried and still no constituent is found that covers the complete input sentence, the search strategy used by the WCP system looks for combinations of maximum length of the *s_cat* constituents according to the preferences expressed in the *s_cat* rule, eventually combining *s_cat* constituents with a minimum number of Wild Cards.

The WCP grammar for Dutch consists of 40 syntactic rules: a total of 124 alternatives. The number of word classes described in the Dutch WCP grammar equals 16. This is a relatively small grammar: it has been designed especially to feed syntactic information into the module for calculating prosody. For that reason, a large range of syntactic phenomena (structures not related to prosody) were not expressed in the grammar.

5 Conclusion

In this paper we have illustrated the purpose specific use of computational linguistic techniques by describing word class assignment and syntactic analysis under the conditions put forward by a TTS system. It appears that computational linguistic techniques used in real–time systems, such as a TTS system, must be robust, fast and probabilistically oriented. Since we are forced to use deterministic search techniques (only one solution is permitted and this solution must be found in real–time), certain analyses will never be found, and a very low number of errors is inevitable. However, ongoing research in this field, and increasing computational power will eventually lead to real–time systems in which more and more sophisticated computational linguistics is present — and necessary — that adheres to the demand of generating only one solution for unrestricted input.

References

Bakel, J. van[1984]*Automatic Semantic Interpretation*, Foris Publications, Dordrecht 1984.

Cox, S.J.[1988]'Hidden Markov models for automatic speech recognition: theory and application', in: *Br. Telecom Technol. J.*, Vol. 6, No. 2, April 1988, pp. 105–115.

Derouault, A. & B. Merialdo[1984]'Natural Language Modeling for Phoneme–to–Text Transcription', in: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI–8, No. 6, November 1986, pp. 742–748.

Gale, W.A. & K.W. Church[1990]'Poor estimates of Context are Worse than None', in: *Proceedings of the workshop Speech and Natural Language*, June 1990, pp. 283–287.

Good, I.J.[1953]'The population frequencies of species and the estimation of population parameters', in: *Biometrika* **40** 1953, pp. 237-264.

Gulikers, L. & R. Willemse[1992]'A Lexicon in a text–to–speech system', in: *Proceedings of the International Conference on Spoken Language Processing*, Banff, 1992.

Hayes, P.J. & G.V. Mouradian[1981]'Flexible Parsing', in: *American Journal of Computational Linguistics*, Vol. 7, Number 4, October–December 1981, pp. 232–242.

Holmes, J.N.[1988]*Speech Synthesis and Recognition*, 1988.

Merialdo, B.[1991]'Tagging text with a probabilistic model', in: *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, Toronto, 1991, pp. 809–812.

Vittorelli, V., G. Adda, R. Billi, L. Boves, M. Jack and E. Vivalda[1990]'Polyglot: multilingual speech recognition and synthesis', in: *Proceedings of the International Conference on Spoken Language Processing*, November 1990, Kobe, pp. 549–552.

Ward, W.H., A.G. Hauptmann, R.M. Stern and T. Chanak[1988] 'Parsing spoken phrases despite missing words', in: *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, Glasgow 1988, pp. 275–278.

Willemse, R. & L. Gulikers[1992]'Word Class Assignment in a text–to–speech system', in: *Proceedings of the International Conference on Spoken Language Processing*, Banff, 1992.