GRAFON: A GRAPHEME-TO-PHONEME CONVERSION SYSTEM FOR DUTCH

Walter Daelemans

AI-LAB, Vrije Universiteit Brussel
Pleinlaan 2 K, 1050 Brussels, Belgium
E-mail: walterd@arti.vub.uucp

Abstract

We describe a set of modules that together make up a grapheme-to-phoneme conversion system for Dutch. Modules include a syllabification program, a fast morphological parser, a lexical database, a phonological knowledge base, transliteration rules, and phonological rules. Knowledge and procedures were implemented object-orientedly. We contrast GRAFON to recent pattern recognition and rule-compiler approaches and try to show that the first fails for languages with concatenative compounding (like Dutch, German, and Scandinavian languages) while the second lacks the flexibility to model different phonological theories. It is claimed that syllables (and not graphemes/phonemes or morphemes) should be central units in a rule-based phonemisation algorithm. Furthermore, the architecture of GRAFON and its user interface make it ideally suited as a rule-testing tool for phonologists.

1. INTRODUCTION

Speech synthesis systems consist of a linguistic and an acoustic part. The linguistic part converts an orthographic representation of a text into a phonetic representation flexible and detailed enough to serve as input to the acoustic part. The acoustic part is a speech synthesiser which may be based on the production of allophones or diphones. This paper is concerned with the linguistic part of speech synthesis for Dutch (a process we will call *phonemisation*). The problem of phonemisation has been approached in different ways. Recently, connectionist approaches (NETtalk: Sejnowski and Rosenberg, 1987) and memory-based reasoning approaches (MBRtalk: Stanfill and Waltz, 1986) have been proposed as alternatives to the traditional symbol-manipulation approach. Within the latter (rule-based) approach, several systems have been built for English (the most comprehensive of which is probably MITalk; Allen, Hunnicutt and Klatt, 1987), and systems for other European languages are beginning to appear.

Text-to-speech systems for Dutch are still in an experimental stage, and two different designs can be distinguished. Some researchers adopt an 'expert system' pattern matching approach /Boot, 1984/, others a 'rule compiler' approach /Kerkhoff, Wester and Boves, 1984; Berendsen, Langeweg and van Leeuwen, 1986/ in which the rules are mostly in an SPE-inspired format. Both approaches take the grapheme/phoneme as a central unit. We will argue that within the symbol manipulation approach, a modular architecture with the syllable as a central unit is to be preferred.

The research described in this paper was supported partly by the European Community under ESPRIT project OS 82. The paper is based on an internal memo (Daelemans, 1985) and on part of a dissertation (Daelemans, 1987b). The system described here is not to be confused with the GRAPHON system developed at the Technische Universität Wien (Pounder and Kommenda, 1986) which is a text-to-speech system for German. I am grateful to my former and present colleagues in Nijmegen and Brussels for providing a stimulating working environment. Erik Wybouw developed C-code for constructing an indexed-sequential version of the lexical database.

The architecture of GRAFON as it is currently implemented is shown in Figure 1.

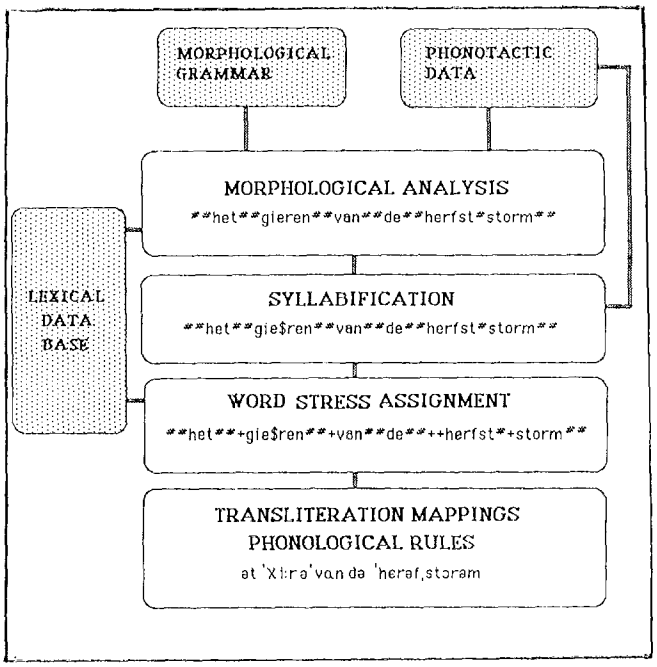

Figure 1. Architecture of GRAFON. Dark boxes indicate knowledge sources, white boxes processing modules. After computing morphological and syllable boundaries, the system retrieves word accent information and applies transliteration mappings and phonological rules to the input. Resulting representations are shown within the boxes.

An input string of orthographic symbols is first analysed morphologically. Then, syllable boundaries are computed, taking into account the morphological boundaries. Morphological analysis uses a lexical database, which is also used to retrieve word stress of monomorphematic word forms. The actual transcription takes the syllable as a basic unit and proceeds in two stages: first, parts of spelling syllables are transliterated into (strings of) phoneme symbols by a number of transliteration mappings. To this representation, context-sensitive phonological rules are applied, modifying parts of the syllables in the process. Any level of phonetic detail (between a *broad* and a *narrow* transcription) can be obtained by adding or blocking rules.

In the remainder of this paper, we will describe the different modules playing a role in GRAFON in some detail, go into some language-specific requirements, and discuss the advantages of our architecture to alternative designs.

2. SYLLABIFICATION

Information about the position of syllable boundaries in spelling input strings is needed for several reasons. The most important of these is that most phonological rules in Dutch have the syllable as their domain. E.g. Dutch has a *schwa-insertion* rule inserting a schwa-like sound between a liquid and a nasal or non-coronal consonant if both consonants belong to the same syllable. Compare *melk* (milk): /mɛlək/ to *mel$ken* (to milk): /mɛlkə/ ($ indicates a syllable boundary). Without syllable structure this problem can only be resolved in an ad hoc way. Furthermore, stress assignment rules should be described in terms of syllable structure /Berendsen and Don, 1987/.

Other rules which are often described as having the morpheme as their domain (such as *devoicing* of voiced obstruents at morpheme-final position and *progressive and regressive assimilation*), should really be described as operating on the syllable level. E.g. *het$ze* (/hɛtsə/: smear campaign; devoicing of voiced fricative at *syllable*-final position) and *as$best* (/azbɛst/: asbestos; regressive assimilation). These mono-morphematic words show the effects of the phonological rules at their syllable boundaries. Furthermore, the proper target of these rules is not one phoneme, but the complete *coda* or *onset* of the syllable, which may consist of more than one phoneme.

Although these examples show convincingly that syllable structure is necessary, they do not prove that it is central. However, the following observations seem to suggest the centrality of the syllable in Dutch phonemisation:
- The combination of syllable structure and information about word stress seems enough to transform all spelling vowels correctly into phonemes, including Dutch grapheme <e>, which is a traditional stumbling block in Dutch phonemisation. Usually, many rules or patterns are needed to transcribe this grapheme adequately.
- All phonological rules traditionally discussed in the literature in terms of morpheme structure can be defined straightforwardly in terms of syllable structure without generating errors.

These facts led us to incorporate a level of syllable decomposition into the algorithm. This module takes spelling strings as input. Automatic syllabification (or hyphenation) is a notoriously thorny problem for Dutch language technology. Dutch syllabification is generally guided by a phonological *maximal onset principle* a principle which states that between two vowels, as many consonants belong to the second syllable as can be pronounced together. This results in syllabifications like *groe-nig* (greenish), *I-na* (a name) and *bad-stof* (terry cloth). However, this principle is sometimes overruled by a *morphological principle*. Internal word boundaries (to be found after prefixes, between parts of a compound and before some suffixes) always coincide with syllable boundaries. This contradicts the syllable boundary position predicted by the *maximal onset principle*. E.g. *groen-achtig* (greenish, *groe-nachtig* expected), *in-enten* (inoculate, *i-nenten* expected) and *stads-tuin* (city garden, *stad-stuin* expected). In Dutch (and German and Scandinavian languages), unlike in English and French, compounding happens through concatenation of word forms (e.g. compare Dutch *spelfout* or German *Rechtschreibungsfehler* to French *faute d'orthographe* or English *spelling error*). Because of this, the default phonological principle fails in many cases (we calculated this number to be on the average 6% of word forms for Dutch). We therefore need a morphological analysis program to detect internal word boundaries. By incorporating a morphological parser, the syllabification module of GRAFON is able (in principle) to find the correct syllable boundaries in the complete vocabulary of Dutch (i.e. all existing and all possible words). Difficulties remain, however, with foreign words and a pathological class of word forms with more than one possible syllabification, e.g. *balletje* may be hyphenated *ballet-je* (small ballet) and *bal-le-tje* (small ball). Syllabification in languages with concatenative compounding is discussed in more detail in Daelemans (1988, forthcoming).

3. LEXICAL DATABASE

We use a *word form* dictionary instead of a morpheme dictionary. At present, some 10,000 citation forms with their associated inflected forms (computed algorithmically) are listed in the lexical database. The entries were collected by the university of Nijmegen from different sources. The choice for a word form lexical database was motivated by the following considerations: First, morphological analysis is reduced to dictionary lookup *sometimes* combined with compound and affix analysis. Complex word forms (i.e. frequent compounds and word forms with affixes) are stored with their internal word boundaries. These boundaries can therefore be retrieved instead of computed. Only the structure of complex words not yet listed in the dictionary must be computed. This makes morphological decomposition computationally less expensive.

Second, the number of errors in morphological parsing owing to overacceptance and nonsense analyses is considerably reduced. Traditional erroneous analyses of systems using a morpheme-based lexicon like *comput+er* and *under+stand*, or for Dutch *kwart+el* (quarter yard instead of quail) and *li+epen* (plural past tense of *lopen*, to run; analysed as 'epics about the Chinese measure li') are avoided this way. Finally, current and forthcoming storage and search technology reduce the overhead involved in using large lexical databases considerably.

Notice that the presence of a lexical database suggests a simpler solution to the phonemisation problem: we could simply store the transcription with each entry (This lexicon-based approach is pursued for Dutch by Lammens, 1987). However, we need the algorithm to compute these transcriptions automatically, and to compute transcriptions of (new) words not listed in the lexical database. Furthermore, the absence of a detailed rule set makes a lexicon-based approach less attractive from a linguistic point of view. Also, from a technological point of view it is a shortcoming that the phonetic detail of the transcription can not be varied for different applications.

Our lexical database system can be functionally interpreted as existing of two layers: a *static storage level* in which word forms are represented as records with fields pointing to other records and fields containing various kinds of information, and a *dynamic knowledge level* in which word forms are instances of linguistic objects grouped in inheritance hierarchies, and have available to them (through inheritance) various kinds of linguistic knowledge and processes. This way new entries and new information associated with existing entries can be dynamically created, and (after checking by the user) stored in the lexical database. This lexical database architecture is described in more detail in Daelemans (1987a).

4. MORPHOLOGICAL ANALYSIS

Morphological analysis consists of two stages: segmentation and parsing. The *segmentation routine* finds possible ways in which the input string can be partitioned into dictionary entries (working from right to left). In the present application, segmentation stops with the 'longest' solution. Continuing to look for analyses with smaller dictionary entries leads to a considerable loss in processing efficiency and an increased risk at nonsense-analyses. The loss in accuracy is minimal (recall that the internal structure of word forms listed in the lexical database can be retrieved).

Some features were incorporated to constrain the number of dictionary lookups necessary: the most efficient of these are a phonotactic check (strings which do not conform to the morpheme structure conditions of Dutch are not looked up), and a special memory buffer (substrings already looked up are cached with the result of their lookup; during segmentation, the same substrings are often looked up more than once).

The *parsing* part of morphological analysis uses a compound grammar and a *chart parser* formalism to accept or reject combinations of dictionary entries. It works from left to right. It also takes into account spelling changes which may occur at the boundary of two parts of a compound (these are called *linking graphemes*, e.g. hemelSblauw; sky-blue, eiERdooier; egg-yolk).

During dictionary-lookup, word stress is retrieved for the dictionary entries (this part of the process could be replaced by additional rules, but as word stress was available in the lexical database, we only had to define the rules for stress assignment in new compounds).

5. PHONOLOGICAL KNOWLEDGE

Knowledge about Dutch phonemes is implemented by means of a type hierarchy, by inheritance and by associating features to objects, in a standard object-oriented way. Information about a particular phonological object can be available through feature inheritance, by computing a method or by returning the stored value of a feature. However, the exact way information from the phonological knowledge base is retrieved, is hidden from the user. An independent interface to the knowledge base is defined consisting of simple LISP-like predicates and (transformation) functions in a uniform format. E.g. (obstruent? x), (syllabic? x), (make-voiced x) etc. The answer can be true, false, a numerical value when a gradation is used a special message (undefined), or in the case of transformation functions, a phoneme or string of phonemes. These functions and predicates, combined with Boolean operators AND, OR and NOT are used to write the conditions and actions of the phonological rules. The interface allows us to model different theoretical formalisms using the same knowledge base. E.g. the *generative phonology* formalism can be modelled at the level of the interface functions.

The morphological analysis and syllabification stages in the algorithm output a list of syllables in which internal and external word boundaries and word stress are marked. Each syllable becomes an instance of the object type syllable, which has a set of features associated with it. Figure 2 lists these features, and their value for one particular syllable.

```
##het##+gie$ren##+van##de##+herfst#+storm

        <syl 5467>
            spelling                  herfst
            closed?                   true
            stressed?                 1
            previous-syllable         <syl 5466>
            next-syllable             <syl 5468>
            external-word-boundary?   false
            internal-word-boundary?   true
            structure                 h e rfst
            onset                     /h/
            nucleus                   /e/
            coda                      /rəf/
            transcription             /herəf/
```

Figure 2. Example instance of the object type SYLLABLE and its associated feature values after transcription.

The value of some of these features is set by means of information in the input: *spelling, closed?* (true if the syllable ends in a consonant), *stressed?* (1 if the syllable carries primary stress, 2 if it carries secondary stress), *previous-syllable* and *next-syllable* (pointers to the neighbouring syllables), *external-word-boundary?* (true if an external word boundary follows), *internal-word-boundary?* (true if an internal word boundary follows). The values of these features are used by the transliteration and phonological rules. Of other features, the value must be computed: *structure* is computed on the basis of the *spelling* feature. The value of this feature reflects the internal structure of the spelling syllable in terms of onset, nucleus and coda. The features *onset, nucleus* and *coda* (this time referring to the phonological syllable) are computed by means of the transliteration and phonological rules. Their initial values are the spelling, their final values are the transcription. The rules have access to the value of these features and may change it. The feature *transcription* stands for the concatenation of the final or intermediate values of *onset, nucleus* and *coda*.

Transliteration rules are mappings from elements of syllable structure to their phonological counterpart. E.g. the syllable onset <sch> is mapped to /sX/, nucleus <ie> to /i/, and coda <x> to /ks/. Conditions can be added to make the mapping context-sensitive: onset <c> is mapped to /s/ if a front vowel follows, and to /k/ if a back vowel follows. There are about forty transliteration mappings.

The phonological rules apply to the output of the transliteration mappings (which may be regarded as some kind of *broad transcription*). They are sequentially ordered. Each rule is an instance of the object type *phonological-rule*, which has six features: *active-p, domain, application, conditions, actions* and *examples*. A rule can be made active or inactive depending on the value of *active-p*. If it is true, sending an *application* message to the rule results in checking the *conditions* on a part of the input string constrained by *domain* (which at present can be syllable, morpheme, word or sentence). If the *conditions* return true, the *actions* expression is executed. Actions may also involve the triggering of other rules. E.g. shwa-insertion triggers re-syllabification. Conditions and actions are written in a language consisting of the phonological functions and predicates mentioned earlier (they access the phonological knowledge base and features of syllables), Boolean connectors, and simple string-manipulation functions (*first, last* etc.). After successful application of a rule, the input string to which it was applied is stored in the *examples* feature. This way, interesting data about the operation of the rule is available from the rule itself. In Figure 3 some examples of rules are shown. Different notations for this rule are possible, e.g. the similarity between both rules could be exploited to merge them into one rule.

6. RELATED RESEARCH

In the *pattern recognition approach* advocated by Martin Boot (1984), it is argued that affix-stripping rules (without using a dictionary) and a set of context-sensitive pattern-matching rules suffice to phonemise spelling input. Boot states that 'there is no linguistic motivation for a phonemisation model in which syllabification plays a significant role'. We

```
REGRESSIVE ASSIMILATION
  Active?          True
  Domain?          Syllable
  Conditions
     (let ((coda-1   (last (coda SYL)))
           (onset-2  (first (onset (next SYL)))))
       (and
         (stop? onset-2)
         (voiced? onset-2)
         (obstruent? coda-1)
         (not (voiced? coda-1))))
  Actions
     (make-voiced (coda SYL))
PROGRESSIVE ASSIMILATION
  Active?          True
  Domain?          Syllable
  Conditions
     (let ((coda-1   (last (coda SYL)))
           (onset-2 (first (onset (next SYL)))))
       (and
         (obstruent? coda-1)
         (not (voiced? coda-1))
         (fricative? onset-2)
         (voiced? onset-2)))
  Actions
     (make-voiceless (onset (next SYL)))
```

Figure 3. A possible definition of voice assimilation rules in Dutch. The LET syntax is used for local variable binding, but is not strictly needed. SYL is bound to the current syllable.

In a *rule compiler approach* (e.g. Kerkhoff, Wester and Boves, 1984; Berendsen, Langeweg and van Leeuwen, 1986), rules in a particular format (most often generative phonology) are compiled into a program, thereby making a strict distinction between the linguistic and computational parts of the system. None of the existing systems incorporates a full morphological analysis. The importance of morphological boundaries is acknowledged, but actual analysis is restricted to a number of (overgenerating) pattern matching rules. Another serious disadvantage is that the user (the linguist) is restricted in a compiler approach to the particular formalism the compiler knows. I would be impossible, for instance, to incorporate theoretical insights from autosegmental and metrical phonology in a straightforward way into existing prototypes. In GRAFON, on the other hand, the phonological knowledge base can be easily extended with new objects and relations between objects, and even at the level of the function and predicate interface, some theoretical modelling can be done. This flexibility is paid, however, by higher demands on the linguist working with the system, as he should be able to write rules in a LISP-like applicative language. However, we hope to have shown from examples of rules in Figure 3 that the complexity is not insurmountable.

7. APPLICATIONS

Apart from its evident role as the linguistic part in a text-to-speech system, GRAFON has also been used in other applications.

7.1. Linguistic Tool

One advantage of computer models of linguistic phenomena is the framework they present for developing, testing and evaluating linguistic theories. To be used as a linguistic tool, a natural language processing system should at least come up to the following requirements: easy modification of rules should be possible, and traces of rule application should be made visible.

In GRAFON, rules can be easily modified both at the macro level (reordering, removing and adding rules) and the micro level (reordering, removing and adding conditions and actions). The scope (domain) of a rule can be varied as well. Possible domains at present are the syllable, the morpheme, the word and the sentence. Furthermore, the application of various rules to an input string is automatically traced and this derivation can be made visible. For each phonological rule, GRAFON keeps a list of all input strings to which the rule applies. This is advantageous when complex rule interactions must be studied. Figure 4 shows the user interface with some output by the program. Apart from the changing of rules, the derivation, and the example list for each different rule, the system also offers menu-based facilities for manipulating various parameters used in the hyphenation, parsing and conversion algorithms, and for compiling and showing statistical information on the distribution of allophones and diphones in a corpus.

7.2. Dictionary Construction

Output of GRAFON was used (after manual checking) by a Dutch lexicographic firm for the construction of the pronunciation representation of Dutch entries in a Dutch-French translation dictionary. The program turned out to be easily adaptable to the requirements by blocking rules which would lead to too much phonetic detail, and by changing the domain of others (e.g. the scope of assimilation rules was restricted to internal word boundaries). The accuracy of the program on the 100,000 word corpus was more than 99%, disregarding loan words. The phonemisation system also plays a central role in the dynamical part of the lexical database architecture we have described elsewhere /Daelemans, 1987a/.

7.3. Spelling Error Correction

A spelling error correction algorithm based on the idea that people write what they hear if they do not know the spelling of a word has been developed by Van Berkel /Van Berkel and De Smedt, 1988/. A dictionary is used in which the word forms have been transformed into phoneme representations with a simplified and adapted version of GRAFON. A possible error is transformed with the same algorithm and matched to the dictionary entries. Combined with a trigram (or rather triphone) method, this system can correct both spelling and typing errors at a reasonable speed.

8. IMPLEMENTATION AND ACCURACY

GRAFON was written in ZetaLisp and Flavors and runs on a Symbolics Lisp Machine. The lexical database is stored on a SUN Workstation and organised indexed-sequentially. Accuracy measures (on randomly chosen Dutch text) are encouraging: in a recent test on a 1000 word text, 99.26% of phonemes and 97.62% of transcribed word tokens generated by GRAFON were judged correct by an independent linguist. The main source of errors by the program was the presence of foreign words in the text (mostly of English and French origin). Only a marginal number of errors was caused by morphological analysis, syllabification or phonological rule application.

There is at present one serious restriction on the system: no syntactic analysis is available and therefore, no sophisticated intonation patterns and sentence accent can be computed. Moreover, it is impossible to experiment with the Phi (the phonological phrase, which may restrict sandhi processes in Dutch) as a domain for phonological rules. However, recently a theory has been put forward by Kager and Quené (1987) in which it is claimed that sentence accent, Phi boundaries and I (intonational phrase) boundaries can be computed *without* exhaustive syntactic analysis. The information needed is restricted to the difference between function and content words, the category of function words, and the difference between verbs and other content words. All this information is accessible in the current implementation of GRAFON through dictionary-lookup.

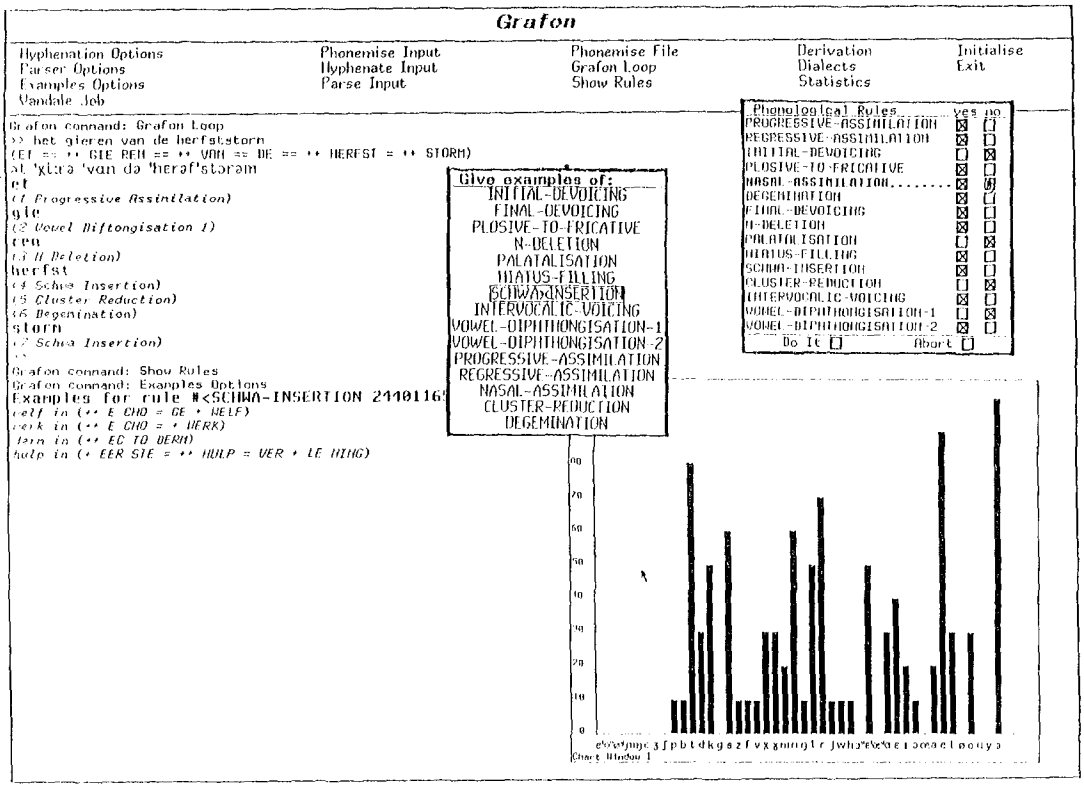

Figure 4. Snapshot of the user interface to GRAFON. Top left, the system has computed an internal representation and transcription of a sentence fragment. A derivation is also printed. In the centre of the display, a menu listing all phonological rules is shown. By clicking on a rule, the user gets a list of input phrases to which the rule has been applied (middle left). The same list of rules is also given in the top right menu. This time, the application of individual rules can be blocked, and the result of this can be studied. The chart bottom right shows the frequency distribution of phonemes for the current session.

9. CONCLUSIONS

It seems that high quality phonemisation for Dutch can be achieved only by incorporating enough linguistic knowledge (about syllable boundaries, internal word boundaries etc.). GRAFON is a first step in this direction. Although it lacks some sources of knowledge (notably about sentence accent and syntactic structure), a transcription of high quality and accuracy can already be obtained, and the system was successfully applied in practical tasks like rule testing, dictionary construction and spelling error correction.

At present, we are working on the integration of a syntactic parser into GRAFON. This would make available the phonological phrase as a domain, and would make the computation of natural intonation patterns possible (using e.g. the algorithm developed in Van Wijk and Kempen, 1987). The alternative approach to the computation of phonological phrase boundaries /Kager and Quené, 1987/ is also being explored.

Another (more trivial) extension is the addition of preprocessors for the verbal expansion of abbreviations and numbers. The specifications of a *lexical analyser* providing this functionality were provided in Daelemans (1987b). An overview of the system including the modules we are presently working on is given in Figure 5.

10. REFERENCES

Allen, J., M.S. Hunnicutt and D. Klatt. *From Text to Speech.* Cambridge, UK: C.U.P., 1987.

Berendsen, E., S. Langeweg and H. van Leeuwen. 'Computational Phonology: Merged, not Mixed.' Proceedings of COLING-86, 1986.

Berendsen, E. and J. Don. 'Morphology and stress in a rule-based grapheme-to-phoneme conversion system for Dutch.' Proceedings European Conference on Speech Technology, Edinburgh 1987.

Berkel, B. Van and K. De Smedt. 'Triphone analysis: A Combined Method for the Correction of Orthographical and Typographical Errors.' Proceedings 2nd ACL Applied Conference, 1988.

Boot. M., *Taal, tekst, computer.* Katwijk: Servire, 1984.

Daelemans, W. 'GRAFON: An Object-oriented System for Automatic Grapheme to Phoneme Transliteration and Phonological Rule Testing.' Memo, University of Nijmegen, 1985.

Daelemans, W. 'A Tool for the Automatic Creation, Extension and Updating of Lexical Knowledge Bases.' Proceedings of the Third ACL European Chapter Conference, 1987a.

Daelemans, W. *Studies in Language Technology: An Object-Oriented Computer Model of Morpho-phonological Aspects of Dutch.* Doctoral Dissertation, University of Leuven, 1987b.

Daelemans, W. 'Automatic Hyphenation: Linguistics versus Engineering.' In: F. Steurs and F.J. Heyvaert (Eds.), *Worlds behind Words*, forthcoming 1988.

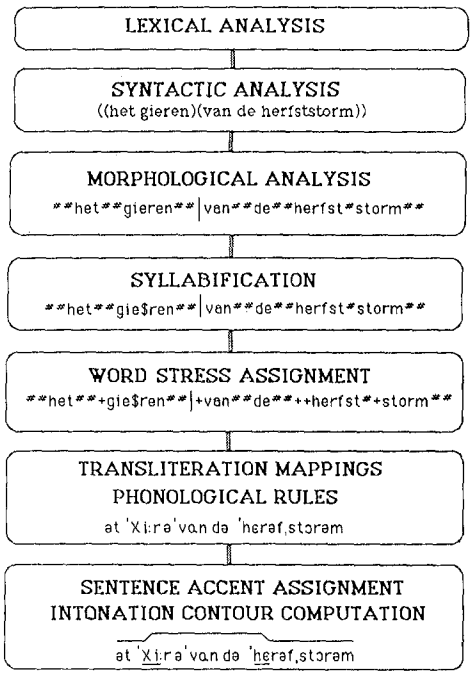

Figure 5. Processing modules in an extended version of GRAFON.

Kager, R. and H. Quené. 'Deriving prosodic sentence structure without exhaustive syntactic analysis.' Proceedings European Conference on Speech Technology, Edinburgh 1987.

Kerkhoff, J., J. Wester and L. Boves, 'A compiler for implementing the linguistic phase of a text-to-speech conversion system'. In: Bennis and Van Lessen Kloeke (eds), *Linguistics in the Netherlands*, p. 111-117, 1984.

Lammens, J.M.G. 'A Lexicon-based Grapheme-to-phoneme Conversion System.' Proceedings European Conference on Speech Technology, Edinburgh 1987.

Pounder, A. and M. Kommenda. 'Morphological Analysis for a German Text-to-speech system.' COLING '86, 1986.

Sejnowski, T.J. and C.R. Rosenberg. 'Parallel Networks that Learn to Pronounce English Text.' *Complex Systems* 1, 1987, 145-168.

Stanfill, C. and D. Waltz. 'Toward Memory-based Reasoning.' *Communications of the ACM*, 29 (12), 1986, 1213-1228.

Wijk, C. van and G. Kempen, 'From sentence structure to intonation contour'. In: B. Muller (Ed.), *Sprachsynthese*. Hidesheim: Georg Olms Verlag, 1985, p. 157-182.