Pattern-driven morphological decomposition

Jeroen Reizevoort—Vincent J. van Heuven

Abstract

In this chapter we describe a technique for morphological decomposition which is based on a form of pattern recognition. This technique was then combined with insights gained in lexical morphology. This combination of technology and theory enables us to quickly split up words into morphemes and to generate the morphological information that is needed in high-quality text-to-speech (TTS) systems.

1. Morphology and text-to-speech systems

Dutch orthography uses 26 alphabetic signs to code about 50 speech sounds. Clearly, there can be no one-to-one relationship between spelling and pronunciation.

When converting text to speech, two methods can be distinguished: the lexicon-based method versus a method based on phonological rules.

The lexicon approach boils down to this: look up every text word in the lexicon and retrieve, quickly and simply, the correct sounds and stress pattern (further see Lammens, this volume). Obviously, this method will be error-prone since the vocabulary of Dutch can be extended indefinitely. Compound words will have to be split up into their constituent parts in order to look up their pronunciation.

The phonological approach generates the pronunciation of words through linguistic rules. This approach fails for words such as *huidarts* /hʌyt$ɑrts/ 'skin doctor, dermatologist' versus *heldin* /hɛl$dɪn/ 'heroine' (further see Heemskerk—van Heuven, this volume). It then transpires that pronunciation is conditioned by morphological boundaries.

It follows from the above that, when no morphological component is available, only severely limited TTS-systems can be produced. A lexicon-based system could accommodate a restricted vocabulary only, whilst the phonological rules could handle only monomorphemic words and complex words with stress-neutral affixes. However, morphological decomposition can help the lexicon-based method by allowing the look-up of word constituents. On the other hand, morphology may help phonology by indicating morpheme boundaries, which may be crucial for deriving the proper pronunciation.

2. The influence of morphology on pronunciation

As we shall expound below, the influence of morphology on pronunciation is mediated by the phonological and syntactic structure of a text utterance.

2.1. Morphology and phonological structure

Morphology influences both the stress pattern of whole words and the pronunciation of individual letters. Each affix has a specific influence on the stress pattern of the base word.

(1)	ˈafval	(defection)	afˈvall+ig	(defective)
	kampiˈoen	(champion)	kampiˈoen+schap	(championship)

These examples show that suffixation with *-ig* changes the position of the stress whilst the suffix *-schap* does not affect the stress pattern of the base word. Moreover, de-stressing a syllable generally triggers reduction of the vowel.

The effect of morphological structure on the pronunciation of individual words is demonstrated by the following examples:

(2)	spelling		pronunciation
	huid#arts	(skin doctor)	hʌyt$ɑrts
	held+in	(heroine)	hɛl$dɪn

The first word, a compound of the nouns *huid* 'skin' and *arts* 'doctor' contains an internal word boundary #. This boundary triggers devoicing of the word-final /d/ of the morpheme *huid*, just as would happen if this word were pronounced in isolation. The second word is a derivation of the noun *held* 'hero' using the female suffix *-in*. The morpheme boundary + is invisible to the syllabification rules of Dutch, so that *d* is parsed as the onset of the second syllable, and remains voiced accordingly. *Heldin* is thus pronounced as if it were a monomorphemic word such as *kade* /kaːdə/ 'quay'.

2.2. Morphology and syntax

When determining the syntactic characteristics of complex words we may use the RIGHTHAND HEAD RULE (RHR) as defined by Williams (1981: 5). This rule predicts that the rightmost part of a derivation or compound

determines (among other things) the syntactic category of the complex word. This proves valid for Dutch as well:

(3)	a.	b.	c.
	$diep_A$	de zee_N	de $diepzee_N$
	deep	the sea	the deepsea
	$vind_V$	$plaats_N$	de $vindplaats_N$
	find	place	the finding place
	de naam	het woord	het naamword
	the name	the word	the nameword (i.e. noun)
	de bal	het spel	het balspel
	the ball	the game	the ballgame

(after Trommelen—Zonneveld 1986: 149)

The examples in (3) show that, whatever the syntactic category of the first element of the compound (a), the properties of the second part (b), i.e. syntactic category and choice of article, determine the properties of the compounds (c).

3. Methods of morphological decomposition

Automatic morphological decomposition is generally achieved by using a lexicon, rules, or a combination of both. We shall now discuss some properties of lexicon-based versus rule-based systems. The properties discussed provide the background against which the third method of morphological decomposition, the pattern-based approach, will be sketched later.

3.1. The rule-based approach

A rule-based system for morphological decomposition employs linguistic, often phonological, rules, such as

(4) e,n ⇒ %,e,n / voc, $cons_0$ — <-segm> ben%en kop%en slap%en
e ⇒ e,# / voc, cons — cons,voc eike#boom
(Berendsen—Don 1987)

The advantages of such systems are that linguistic insights can be tested, and that — in principle — all possible words in the language can be analyzed.

A disadvantage of such methods is that the rule system may grow quite complex, which slows down execution time. The interactions among the many rules tend to become untractable, which makes the introduction of changes to the system a hazardous affair.

3.2. The lexicon-based approach

The lexicon approach uses an exhaustive morpheme lexicon. Polymorphemic words are split up by checking which concatenations of morphemes may cover the entire word.

The advantages of this method are that all possible words can be decomposed, since all morphemes are contained in the lexicon, and that information on syntactic category can be obtained through morpheme combination rules. A disadvantage of this type of system is that it tends to generate multiple analyses for one word form. Additional precautions, typically in the form of (ad hoc) rules, will then have to be taken in order to restrict the number of competing analyses (see Heemskerk—van Heuven, this volume).

3.3. Is there a superior method?

It seems impossible to prefer one method over the other. The ultimate choice depends on the objective of the decomposition. Relevant criteria may be speed, linguistic insight, precision, flexibility, and versatility.

A problem that faces any method, are ambiguities of the type *kwart#slagen* 'quarter beats' versus *kwarts#lagen* 'quartz layers' and *'balletje* 'little ball' versus *bal'letje* 'little ballet'. Other problems reside in spelling and typing errors, foreign words and technical vocabulary.

4. The pattern technology

4.1. Patterns and hyphenation

The pattern approach was introduced by Liang (1983) as a solution to the problem of computer hyphenation at line breaks. Liang aimed for a hyphenation algorithm that was fast, error-free, easy to adapt, memory-independent, and non language-specific.

Liang assumed that there can be no complete lexicon for any language containing all the correct hyphenation positions for all the words, and that

even an approximation to such a database would be too demanding in terms of memory space and look-up time. A rule-based system, on the other hand, would always be language-specific, and hard, if not impossible, to get error-free.

In a way Liang then opted for the best of both worlds: his pattern technology combines features of the lexicon-based and the rule-based approaches.

4.1.1. Generating patterns

The algorithm that generates the hyphenation positions is as follows:

1. Compile a dictionary with only correct hyphenations.
2. Determine for each word in this dictionary how it would be hyphenated with the present set of patterns.
3. If the word is correctly hyphenated with the present set, do nothing.
4. Else: store as many characters around the word's hyphenation position until a unique pattern is obtained that contains the correct hyphen position.

Patterns are strings of alphanumeric characters that describe the environment for hyphen insertion. The length of a pattern depends on the generality of the hyphenation rule that is captured by it. Typically, short and general patterns are generated first, and patterns grow longer as the words they apply to are more exceptional. In order to obtain an error-free set of hyphenation patterns the maximal length of the patterns will have to be such that any exception to the general patterns can still be characterized by a pattern. Patterns will, however, always be optimally compact since only those characters around a hyphen position are stored that are necessary to uniquely define it. The effectivity of a pattern, expressed as the number of words that can be hyphenated by it, will be inversely proportional to its length.

For example, it is generally possible to hyphenate words immediately before the Dutch character string *heid* (*ijdel-heid* 'vanity', *dom-heid* 'stupidity', etc.). Therefore a short pattern will be generated for this suffix. The word *afscheid* 'farewell' is an exception to this pattern, and will have to be covered by a longer pattern.

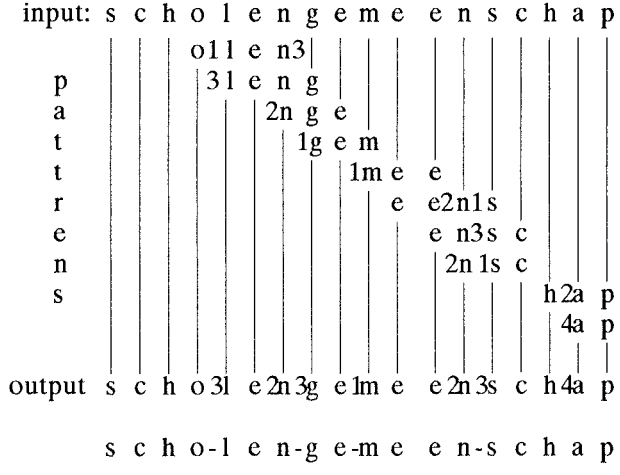
Patterns are generated in tiers. An odd-numbered tier generates patterns that indicate hyphenation positions, the subsequent even-numbered tier determines which words are incorrectly hyphenated by the earlier patterns, and then generates patterns that indicate illegal hyphen positions. For each pattern that is generated at a specific tier, the number of successful hyphenations is counted, as well as the number of errors. These counts

are made going through the entire dictionary, after which the system decides whether a pattern makes a sufficient contribution to warrant permanent incorporation.

4.1.2. Using the patterns

After patterns have been generated for the entire dictionary, these can be used to hyphenate words. For a given text word the program selects the patterns that are applicable. On the basis of the patterns selected the text word is hyphenated. An example is:

(5)



The example shows that the following patterns match the word *scholengemeenschap* 'comprehensive school': *o1len3*, *3leng*, *2nge*, *1gem*, *1mee*, *ee2n1s*, *en3sc*, *2n1sc*, *h2ap*, *4ap*. Here odd numbers within a pattern are imperative (i.e., hyphenation is mandatory in this position) whereas even numbers are prohibitive (i.e., under no circumstances may a word be hyphenated in this position). In case of overlapping patterns the highest valued (odd-numbered) pattern takes precedence. In (5) above the correct hyphenation of *scholengemeenschap* is listed under output. Whenever multiple values occur in a column between individual characters, the highest value percolates to the final result. All odd-numbered values are then taken as hyphenation positions.

4.2. Results with pattern hyphenation

The success of Liang's pattern technology, in terms of speed, accuracy, flexibility, compactness and language independence, has been demonstrated by test results for English by Liang himself, as well as for Dutch by Aerts (1986). The latter program is currently commercially used in the printing industry.

The flexibility of the pattern approach is apparent in many ways. Firstly, the method clearly works independently of any specific language. Secondly, words that are incorrectly hyphenated can be added to the dictionary, so that — after generating an updated set of patterns — these, too, will be correctly hyphenated. Crucially, the pattern technology is flexible enough to allow us to extend its applicability to other processes:

(6) The effectiveness of pattern matching suggests that this paradigm may be useful in other applications as well. Indeed more general pattern matching systems and related notions of productions systems and augmented transition networks (ATNs) are often used in artificial intelligence applications, especially natural language processing. While AI programs try to understand sentences by analyzing word patterns, we try to hyphenate words by analyzing letter patterns. (Liang 1983: 42)

4.3. Patterns and morphology

Linguistic rules underlie syllabification, hyphenation, as well as morphological decomposition. The word *rules* points to the existence of regularities, and regularities can be expressed in terms of patterns. Such patterns can be traced in a dictionary that contains word-internal morpheme boundaries, using a pattern generator of the type discussed in section 4.1.1. This would yield a method of morphological decomposition that allows us to combine the advantages of the rule-based approach (section 3.1) with those of the lexicon-based approach (section 3.2). The resulting patterns are global enough to function as rules, while, on the other hand, they may be detailed enough to correctly deal with exceptions.

The advantages of the pattern approach, as opposed to the rule-based approach, are greater speed and flexibility. Particularly, exceptions and subregularities will be processed appreciably faster. The advantages over morphological decomposition with a full-fledged morpheme lexicon are greater speed and conciseness. Processing is faster since only a single solution is sought through a single search beam. As for conciseness, we may

expect the number of patterns to be smaller than the number of morphemes in the language, and the patterns generated to be shorter than the morphemes. This is caused by the fact that patterns incorporate the graphotactic constraints that are operative in any given orthography.

As we have stated above, neither the pattern approach, nor the lexicon-based and rule-based methods, can handle semantic ambiguities. Yet, it will never be the case that the pattern generator will end up oscillating between two possible decompositions; the mere fact that there is a maximum pattern length, will prevent this from happening. Only one out of many possible decompositions will be chosen.

5. The theory, lexical morphology

5.1. Introduction

The goal of the PADMAN project (PAttern Driven Morphological ANalysis) was to design and test a module for morphological decomposition based on Liang's pattern approach.

The original hyphenation algorithm, as described in section 4.1.1, cannot be used for morphological decomposition just like that. For hyphenation purposes it is sufficient to just indicate where hyphens must be inserted. There is, in other words, only one type of boundary. As we have demonstrated earlier on, the pronunciation of complex words depends, among other things, on the type of morpheme boundary that separates the constituents within the word. In order to exploit morphology in a sensible way for text-to-speech purposes, we will therefore have to be able to insert several types of boundary. When generating the patterns, we need not only indicate where boundaries are to be inserted but also what kind of boundaries.

The word list that forms the basis for generating the patterns, will have to contain boundaries that are motivated by a morphological theory that accurately predicts the pronunciation of complex Dutch words. The most complete morphological theory for Dutch is LEXICAL MORPHOLOGY (see Heemskerk—van Heuven, this volume). This theory has been adopted for PADMAN. The next three sections sketch the theory.

5.2. The creation of lexical morphology

The influence of morphology on phonology is widely known and was first extensively commented on by Chomsky—Halle (1968) in *The sound pattern of English* (henceforth SPE). Chomsky—Halle distinguish two types of

morpheme boundary in English, with different effects on the pronunciation, specifically on the stress pattern, of polymorphemic words for each type: stress neutral (#) versus stress sensitive (+) boundaries.

This binary split of affixes in SPE prompted Siegel (1974) to devise a model in which stress rules operate in the morphological component, i.e., within the lexicon, rather than in the phonological component: Lexical Morphology was born.

Siegel observed that, in English, +-suffixes never occur closer to the word edge than #-suffixes. She then proposed that these suffixes each attach on their own level, and to formulate specific phonological rules for each of these morphological levels separately. These phonological rules operate cyclically, but only within their own level. As a consequence, morphology has (limited) access to phonological rules. This allows us to explain a number of morphological processes (see, e.g., van Beurden 1986: 13).

5.3. Lexical morphology and Dutch

Van Beurden (1986) shows that lexical morphology can also be used for Dutch, if more than two levels are distinguished (see section 5.3.2). Unlike English, native Dutch suffixes come in three types: they can be stress neutral, stress attracting, or stress bearing. We shall now discuss some of the consequences of the theory of lexical morphology for the assignment of stress (section 5.3.1) and the combinatory possibilities of Dutch morphemes (section 5.3.2).

5.4. Stress assignment

For the purpose of stress assignment four types of suffix have to be distinguished in Dutch:

(7)	a.	Nonnative/Roman	*sport*	'sport'	*sport-'ief*	'sportive'
	b.	Stress bearing	*held*	'hero'	*held-'in*	'heroine'
	c.	Stress neutral	*vrolijk*	'cheerful'	*vrolijk-'he id*	'cheerfulness'
	d.	Stress attracting	*vijand*	'enemy'	*vij'and-ig*	'enemical'

In terms of stress assignment, nonnative or Roman suffixes behave as if the derived word were monomorphemic. This is because such suffixes end in a superheavy syllable, i.e., typically contain three or more segments in the rime part (cf. Kager 1989; Neijt—van Heuven 1992). Monomorphemic words ending in a superheavy syllable, are invariably stressed on that

syllable. If we consider words with nonnative suffixes to be morphologically simplex, we correctly predict main stress on the suffix.

The three remaining types of suffix are of Germanic origin. Stress bearing suffixes are *-ˈin*, *-ˈes*, and *-ˈij*. Stress attracting suffixes shift the main stress to the nearest non-schwa syllable before the suffix (*ˈ-ig*, *ˈ-(e)lijk*, *ˈ-isch*). Stress neutral suffixes, as the name suggests, leave the stress pattern of the base word unaffected (*-schap*, *-heid*, *-dom*). All inflections fall in this latter category as well.

There are four prefixes that generally leave the stress pattern of the base word unaltered: *be-*, *ge-*, *ont-*, and *ver-*. All other prefixes, such as *aarts-* *oer-*, and *over-*, behave as the leftmost part of a compound, and are generally considered to be words rather than prefixes (cf. Langeweg 1988). Words with Roman prefixes are best considered as morphologically simplex, at least as far as stress assignment is concerned.

5.5. Morphotactic constraints

Derivation of complex words through affixation is bound to morphotactic restrictions. It is not the case that any affix can be attached to any morpheme. Van Beurden (1987) discusses the ordering relationships among the various affixation processes. Diagram (8) below summarizes van Beurden's conclusions (see also Heemskerk—van Heuven, this volume):

(8)	Underived words/ Romance derivations	⇒	main stress rule (generalization: words referring to human beings take the article *de*
	V-derivation	⇒	V-level phonology
	A-derivation	⇒	main stress on first (full) vowel before suffix
	N-derivation	⇒	N-level phonology

First, stress is assigned to morphologically simplex words and to words with nonnative affixes. In the next stage verbs are allowed to be derived; since at this stage only simplex words are available, it follows that verbs can only be derived from simplex words. Adjective formation takes place after the generation of verbs, and may therefore be derived by either attaching a stress attracting suffix to a simplex base form, or by attaching a stress attracting suffix to a derived verb. Derived nouns can never be the base for the derivation of an adjective. Finally, nouns can be derived from any type of word, be it simplex or complex, including derived nouns.

6. Linguistic theory and language technology in practice

6.1. Implementing the theory

Lexical morphology distinguishes between various levels of morpheme attachment, and — as a consequence — various types of morpheme boundary (see above). Since this organization in terms of levels is largely based on stress behavior, it is highly appropriate for use in text-to-speech (TTS) systems. The morphemic information that is being used in PADMAN is inspired by lexical morphology. In PADMAN nine types of boundary are distinguished:

(9)

Germanic	*Roman*	*Other*
Stress neutral prefix	Prefix	Compound boundary
Stress neutral suffix	Suffix	Binding grapheme
Stress attracting suffix		Improductive
Stress bearing suffix		

Any morpheme combination can now be characterized using these types of boundary. By marking the words in the training lexicon not only for position of morpheme boundaries, but also for type of boundary (with an integer between one and nine), the theory of lexical morphology is implicitly incorporated into the patterns. When the resulting patterns are applied to the task of automatically segmenting words into morphemes, information on boundary type comes available as a bonus. With the aid of this information the phonological component of the TTS system is in a position to determine the correct pronunciation of any Dutch word.

6.2. Implementation

6.2.1. Adaptation of the original algorithm

A closer study of the hyphenation pattern generator developed by Aerts (1986) revealed that this program had to be adapted substantially in order to produce the desired patterns needed for morphological decomposition. The original program has no provision for hyphenating words immediately after the first or immediately before the final letter, since this is highly unusual in the printing industry. There are, however, several single-letter morphemes in Dutch, which have to be followed or preceded by a morpheme boundary, e.g., *a-sociaal* 'non social', *tafel-s* 'tables'. The program

was adapted so as to allow single-letter morphemes in word-marginal position. Further adaptations were made in order to allow the program to recognize information on boundary type.

6.2.2. Building a training lexicon

The pattern approach crucially depends on the availability of a correctly segmented word list on the basis of which the segmentation patterns can be generated, i.e., a training lexicon. The NWO/SURF Expertise Centre for Lexical Data (CELEX, Nijmegen) supplied us with a computer readable corpus containing 123,093 morphologically segmented Dutch words. Information on type of boundary, in terms of the nine types distinguished above, was added to the corpus automatically.

6.2.3. Generating the patterns

In order to test the feasibility of the pattern approach as a means of morphological decomposition, we decided to first generate patterns for compound words and for derivations with native affixes. For this purpose a subcorpus was extracted from the CELEX word list containing 64,096 morphologically segmented words. Patterns were then generated in six tiers with a maximum length of five characters. The algorithm produced 6,482 patterns, which were able to detect 82.4 percent of the morpheme boundaries. This list of patterns could be stored in 50 kbytes of memory, whereas the source lexicon occupied 1,015 kbytes. This was a highly encouraging result.

We discovered that the source lexicon contained many inconsistencies. Such inconsistencies, e.g. *aanrecht-kast* 'sink-cupboard' versus *aan-recht-keuken* 'sink-kitchen', force the generation of additional patterns, or yield errors when the maximum string length in too short to capture the correct pattern. We expected that elimination of inconsistencies would reduce the size of the pattern inventory, reduce the maximum pattern length, and improve the percentage of correct decompositions.

After manual correction of the training lexicon, decomposition performance was much improved, indeed. Patterns were generated on eight tiers, and performance was as indicated in Table 1:

Table 1. Performance of pattern-driven morphological decomposition

Tier	Pattern length	Total number of patterns	Cumulative percent correct
1	1-2	209	29.6
2	1-2	210	29.8
3	2-3	1274	56.5
4	2-3	1294	56.5
5	3-4	6102	82.4
6	3-5	6253	82.4
7	4-6	11385	95.9
8	4-7	11979	96.0

The memory size of the pattern inventory had grown to 105 kbytes; the size of the training lexicon has remained unchanged (1,015 kbytes). Notice that at tier 6 percent correct decompositions was the same as in the case of the uncorrected training lexicon, but with a reduction in pattern inventory of 229. Unfortunately, there was no time to do an extensive error analysis of the remaining incorrect segmentations, which in turn might have allowed us to further fine-tune the pattern inventory. Also, a proper evaluation of the pattern approach can only be obtained when the training lexicon is also includes productive non-native and inflectional morphology. In spite of these caveats, however, we consider this first exploration of the possibilities of the pattern approach to morphological decomposition highly encouraging.