Three Principled Methods of Automatic Word Form Recognition

Roland Hausser

Abstract

For the computer, word forms in an online text are simply letter sequences between blanks. A rule-based automatic language analyses presupposes, however, that the computer can *recognize* the individual word forms. This includes assigning the base form (lemmatization) and determining the morphosyntactic properties (categorization).

It is shown that there are three principled methods of automatic word form recognition, based on word forms, morphemes, and allomorphs, respectively. After describing these different methods using the notions of traditional morphology, they are compared with regards to their handling of neologisms, time efficiency, and space requirements.

1 Morphemes and allomorphs

In traditional morphology, word forms are analyzed by disassembling them into their elementary parts. These are called morphemes and defined as the smallest meaningful units of language. In contrast to the number of potential words, the number of morphemes is finite.

The notion of a morpheme is a linguistic abstraction which is manifested concretely in the form of finitely many allomorphs. The word allomorph is of Greek origin and means “alternative shape.” For example, the morpheme wolf is realized as the two allomorphs wolf and wolv.

Just as the elementary parts of the syntax are really the word forms (and not the words), the elementary parts of morphology are really the allomorphs. A morpheme may be defined as naming the set of associated allomorphs.

1.1 DEFINITION OF THE NOTION *morpheme*

morpheme $=_{def}$ {associated analyzed allomorphs}

Allomorphs are formally analyzed here as ordered triples, consisting of the surface, the category and the semantic representation. The following examples, based on the English noun wolf, are intended to demonstrate these basic concepts of morphology as simply as possible.[1]

[1] Nouns of English ending in -lf, such as calf, shelf, self, etc. form their plural in general as -lves. One might prefer for practical purposes to treat forms like wolves, calves, or shelves as elementary allomorphic forms, rather than combining an allomorphic noun stem ending in -lv with the plural allomorph es. This, however, would prevent us from explaining the interaction of concatenation and allomorphy with an example from English.

1.2 Formal analysis of the morpheme wolf

morpheme	*allomorphs*
wolf $=_{def}$	{[wolf (SN SR) wolf], [wolv (PN SR) wolf]}

The different allomorphs wolf and wolv are shown to belong to the same morpheme by the common semantic representation in the third position. As (the name of) the semantic representation we use the base form of the allomorph, i.e. wolf, which is also used as the name of the associated morpheme.

Some surfaces such as wolf can be analyzed alternatively as an allomorph, a morpheme (name), a word form, or a word (name).

1.3 Comparing morpheme and word wolf

morpheme	*allomorphs*	*word*	*word forms*
wolf $=_{def}$	{wolf,	wolf $=_{def}$	{wolf,
	wolv}		wolf/'s,
			wolv/es,
			wolv/es/'}

Other surfaces can be analyzed only as an allomorph, e.g. wolv, or only as a word form, e.g. wolves.

Besides the segmentation into morphemes or allomorphs, a word form surface may also be segmented into the units of its realization medium. Thus, written surfaces may be segmented into letters and spoken surfaces into syllables or phonemes.

1.4 Alternative forms of segmentation

allomorphs:	learn/ing
syllables:	lear/ning
phonemes:	l/e/r/n/i/n/g
letters:	l/e/a/r/n/i/n/g

The syllables lear and ning do not coincide with the allomorphs learn and ing, and similarly in the case of letters and phonemes. While, e.g., syllables are important in automatic speech recognition, morphological analysis and automatic word form recognition aim at segmenting the surface into morphemes or allomorphs, which are independent[2] of the concrete realisation in speaking or writing.

2 Irregular words

The number and variation of allomorphs of a given morpheme determine the degree of regularity of the morpheme and – in the case of a free morpheme – the associated word. An example of a regular word is the verb to learn, the morpheme of which is defined as a set containing only one allomorph.

[2] In as much as the medium of realization influences the representation of allomorphs (types), there is the distinction between allo*graphs* in written and allo*phones* in spoken language. Allographs are, e.g., happy and happi-, allophones the present vs. past tense pronunciation of read.

2.1 THE REGULAR MORPHEME learn

morpheme *allomorphs*
learn $=_{def}$ {[learn (N . . . V) learn]}

An irregular word, on the other hand, is the verb to swim, the morpheme of which has four allomorphs, namely swim, swimm,[3] swam, and swum. The change of the stem vowel may be found also in other verbs, e.g., sing, sang, sung, and is called *ablaut*.

2.2 THE IRREGULAR MORPHEME swim

morpheme *allomorphs*
swim $=_{def}$ {[swim (N . . . V1) swim],
[swimm (. . . B) swim],
[swam (N . . . V2) swim],
[swum (N . . . V) swim]}

In 2.2, the allomorph of the base form is used as the name of the morpheme. Thus, we may say that swam is an allomorph of the morpheme swim.

Cases where there is no similarity at all between the allomorphs of a given morpheme are called *suppletion*.

2.3 AN EXAMPLE OF SUPPLETION

morpheme *allomorphs*
good $=_{def}$ {[good (ADV IR) good],
[bett (CAD IR) good],
[b (SAD IR) good]}

While the regular comparation in, e.g.,
fast, fast/er, fast/est
uses only one allomorph for the stem, the irregular comparation in, e.g.,
good, bett/er, b/est
uses several allomorphs.[4] Even in a suppletive form like bett, the associated morpheme good is readily available as the third element of the ordered triple analysis.

In structuralism, the morphemes of the open and closed classes are called *free* morphemes, in contradistinction to *bound* morphemes. A morpheme is *free* if it can occcur as an independent word form, e.g. *book*. Bound morphemes, on the other hand, are affixes such as the prefixes un-, pre-, dis-, etc., and the suffixes -s, -ed, -ing, etc., which can occur only in combination with free morphemes.

The following example representes the English plural morpheme, which has been claimed to arise in such different forms as book/s, wolv/es, ox/en and sheep/#.

2.4 EXAMPLE OF A BOUND MORPHEME (hypothetical)

morpheme *allomorphs*
-s $=_{def}$ {[s (PL1) plural],
[es (PL2) plural],

[3] This allomorph is used in the progressive swimm/ing, avoiding the concatenative insertion of the gemination letter.

[4] For practical purposes, one may analyze good, better, best as basic allomorphs without concatenation.

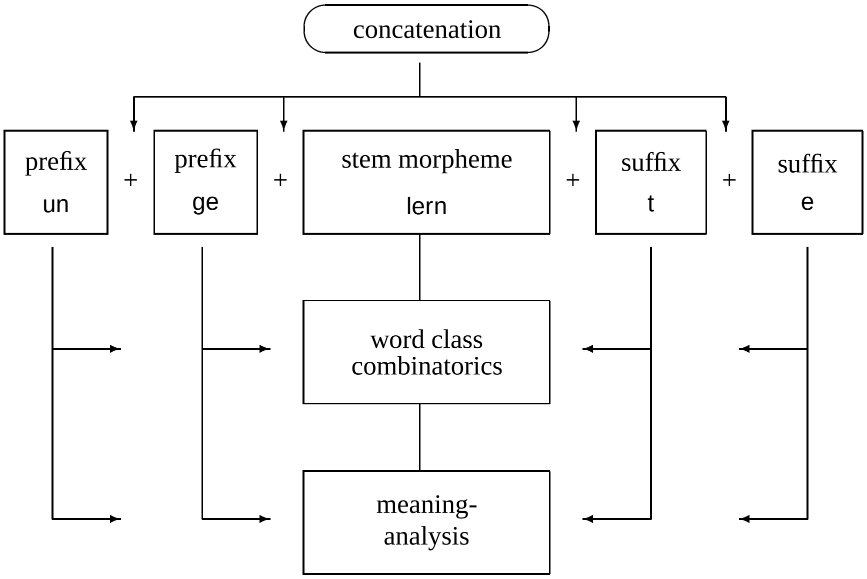
[en (PL3) plural]
[# (PL4) plural]}

In bound morphemes, the choice of the morpheme name, here -s, and the base form of the allomorph, here 'plural' is quite artificial. Also, postulating the 'zero allomorph' # is in violation of the principle of surface compositionality.

3 Categorization and lemmatization

The morphological analysis of an unknown word form surface consists in principle of the following three steps. The unanalyzed word form surface is (i) disassembled into its basic elements (segmentation), (ii) the basic elements are analyzed in terms of their grammatical definitions (lexical look-up), and (iii) the analyzed basic elements are reassembled based on rules, whereby the overall analysis of the word form is derived (concatenation). Thereby concatenation applies to the surface, the category, and the semantic representation simultaneously. Depending on the approach, the basic elements of word forms are either the allomorphs or the morphemes.

3.1 Morphological analysis of German ungelernte



For automatic word form recognition, the following components are required .

3.2 Components of word form recognition

- *On-line lexicon*

  For each element (e.g., morpheme) of the natural language there must be defined a lexical analysis which is stored electronically.

- *Recognition algorithm*

  Each unknown word form (e.g., wolves) must be characterized by the system automatically with respect to categorization and lemmatization:

  - *Categorization*
    Specifying the part of speech (e.g., noun) and the morphosyntactic properties of the surface (e.g., plural). Categorization is needed for syntactic analysis.
  - *Lemmatization*
    Specifying the correct base form (e.g., wolf). Lemmatization is needed for semantic analysis: the base form provides access to corresponding lemmata in a semantic lexicon.

The formal structure of an on-line lexicon is similar to that of a traditional dictionary. It consists of alphabetically ordered lemmata of the following basic structure:

3.3 BASIC STRUCTURE OF A LEMMA

[surface (lexical description)]

The lemmata are arranged in the alphabetical order of their surfaces. The surfaces serve as keys which are used both for the ordering of the lemmata during the building of the lexicon and for finding a certain lemma in the lexicon. The surface is followed by the associated lexical description.

Because traditional and electronic lexica are based on the same basic structure, traditional dictionaries are well-suited for lemmatization in automatic word form recognition, provided that they exist on-line and no copy-rights are violated. For example, in Webster's *New Collegiate Dictionary*, the word wolf has the following lemma:

3.4 LEMMA OF A TRADITIONAL DICTIONARY (*excerpt*)

> ^{1}**wolf** \'wu̇lf\ *n. pl* **wolves** \'wu̇lvz\ *often attributed* [ME, fr. OE *wulf*; akin to OHG *wolf*, L *lupus*, Gk *lykos*] **1** *pl also* **wolf a:** any of various large predatory mammals (genus *Canis* and exp. *C. lupus*) that resemble the related dogs, are destructive to game and livestock, and may rarely attack man esp. when in a pack – compare `COYOTE, JACKAL` **b:** the fur of a wolf …

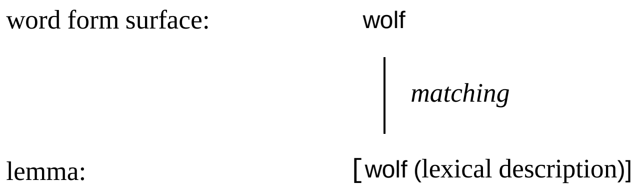
The surface is followed by the lexical description, which specifies the pronunciation, the part of speech (n), the plural form in writing and pronunciation, the etymology, and a number of semantic descriptions and pragmatic uses.

The crucial properties of a lemma like 3.4 are the quality of the information contained and the structural consistency of its coding. If these are given on-line, the information can be renamed, restructured, and reformatted automatically,[5] without losing any of the original information.

The recognition algorithm in its simplest form consists in matching the surface of the unknown word form with the corresponding key of a lemma in the on-line lexicon, thus providing access to the relevant lexical description.

[5] For this, special programming languages like AWK (Aho, Kerningham & Weinberger 1988) and PERL (Wall & Schwartz 1990) are available.

3.5 MATCHING A SURFACE ONTO A KEY



word form surface: wolf

matching

lemma: [wolf (lexical description)]

There exist several computational methods to match a given surface automatically with the proper lemma in an electronic lexicon.[6]

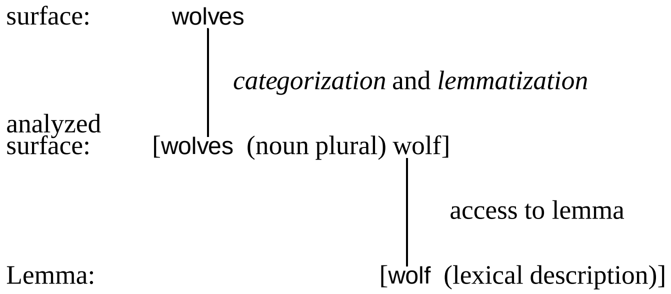
The simplest is a *linear search*, i.e. going sequentially through the list of lemmata until there is a match between the unknown surface and the key. In small lexica (containing up to 50 lemmata) this method is well-suited. Possible applications are the formal languages, where each surface must be assigned a category by way of lexical lookup.

The lexica of natural language are considerably larger, however, containing between 20 000 and 1 000 000 entries, depending on their purpose. Even more importantly, most words are related to several word *forms* which must be categorized and lemmatized. Because in the natural languages

- the number of word forms is considerably larger than the number of words, at least in inflectional and agglutinating languages, and
- the lexical lemmata normally define words rather than word forms,

it is best to handle categorization and lemmatization, on the one hand, and access to the lemma, on the other, in two separate steps.

3.6 TWO STEP PROCEDURE OF WORD FORM RECOGNITION



surface: wolves

categorization and *lemmatization*

analyzed surface: [wolves (noun plural) wolf]

access to lemma

Lemma: [wolf (lexical description)]

Automatic word form recognition takes place between the levels of the surface and the analyzed surface. It consists of categorization and lemmatization, and is based on a special analysis lexicon. Access to the lemma, containing the semantic representation common to the whole paradigm, takes place in a second step, using a traditional base form lexicon.

[6] See Aho & Ullman 1977, p. 336–341.

4 Methods of automatic word form recognition

Possible methods of automatic word form recognition may be distinguished as to whether their analysis lexicon specifies *word forms*, *morphemes*, or *allomorphs*.[7] Each of the three methods exhibits a characteristic correlation between the recognition algorithm and the associated analysis lexicon.

The *word form method*[8] uses an analysis lexicon consisting of word forms.

4.1 WORD FORM METHOD: analyzed word form as lexical lemma

[wolves (part of speech: Subst, num: Pl, case: N,D,A, base form: wolf)]

An analysis lexicon of word forms allows for the simplest recognition algorithm because the surface of the unknown word form, e.g., wolves, is simply matched whole onto the corresponding key in the analysis lexicon.

Of the three steps of morphological analysis, namely (i) segmentation, (ii) lexical lookup, and (iii) concatenation, the first and the third are here identity mappings, for which reason the word form method is a border line case of morphological analysis. Also, categorization and lemmatization are handled here solely by the lexical entry.

The word form method may be useful as a quick and dirty method for toy systems, providing lexical lookup without much programming effort. In the long run this method is costly, however, because of (i) the production, (ii) the size, and (iii) the basic finiteness of its analysis lexicon.

The third point refers to the fact that the word form method provides no possibility to recognize neologisms during runtime. Because the word form method is inherently limited to the entries provided by its analysis lexicon, the analysis of arbitrary free text would require that all *possible* word forms are provided by the analysis lexicon. This, however, is impossible because of the productivity of natural language morphology.

The *morpheme method*, on the other hand, uses the smallest possible analysis lexicon, consisting of analyzed morphemes.[9] Compared to the word form method, it has the further advantage that neologisms may be analyzed and recognized during run-time – using a rule-based segmentation and concatenation of complex word forms into their elements (morphemes). The only requirement is that the elements are lexically known and their mode of composition can be handled correctly by the rules.

The disadvantage of the morpheme method is a maximally complex recognition algorithm. The analysis of an unknown surface during run-time requires the steps of (1) segmentation into allomorphs, (2) reduction of the allomorphs to the corresponding morphemes, (3) recognition of the morphemes using an analysis lexicon, and (4) the rule-based concatenation of the morphemes to derive the analyzed word form.

In case of the word form wolves, for example, step (1) consists in the segmentation into the allomorphs wolv and es. Step (2) reduces these to the corresponding morpheme surfaces wolf and s, enabling lexical lookup as step (3). In step (4), the resulting analyzed morphemes are concatenated by means of grammatical rules which derive the morphosyntactic properties of the word form as a whole, including categorization and lemmatization.

[7] The fourth basic concept of morphology, the *word*, does not provide for a recognition method because words are not a suitable key for a multitude of word forms. See 4.4.

[8] Also known as the *full-form* method based on a *full form lexicon.*

[9] In light of the morpheme definition 1.1, a morpheme lexicon consists strictly speaking of analyzed base form allomorphs.

4.2 Schema of the morpheme method

```
surface:          wolves
                   |   |          segmentation
allomorphes:     wolv/es
                   ⇓   ⇓          reduction
morphemes:        wolf+s        base form lookup and concatenation
```

Conceptually, the morpheme method is related to transformational grammar. Allomorphs are not treated as fully analyzed grammatical entities, but exist only as the quasi-adulterated surface reflexions of the 'underlying' morphemes, which are regarded as the 'real' entities of the theory. Concatenation takes place at the level of morphemes – and not at the level of the concretely given allomorphs. For this reason, the morpheme method violates the principle of surface compositionality. Also, because the morpheme method tries to compose the morphemes as much as possible as constituents, it violates the principle of time linearity, also known as de Saussure's second law.

Mathematically and computationally, the morpheme method is of high complexity (NP complete),[10] because the system must check the surface for *all possible* phenomena of allomorphy. Faced with English valves, for example, the system would have to consider nonexisting *valf+s as a possible underlying morpheme sequence. Only after all potential allomorph-morpheme reductions have been checked for a given surface can concatenation begin.

The *allomorph* method[11] combines the respective advantages of the word form and the morpheme method by using a simple recognition algorithm with a small analysis lexicon. Based on its rule-based analysis, the allomorph method can also recognize neologisms during run-time,

Before run-time, the analysis lexicon is derived automatically from an elementary lexicon by means of allo-rules. The elementary lexicon consists in (i) the analyzed elementary base forms[12] of the open word classes, (ii) the analyzed elements of the closed word classes, and (iii) the allomorphs of the affixes[13] as needed in inflection, derivation, and composition.

During run-time, the allomorphs of the analysis lexicon are available as precomputed, fully analyzed forms[14] (e.g., 1.2, 2.1, 2.2), providing the basis for a maximally simple segmention: the unknown surface is matched from left to right with suitable allomorphs – without any reduction to morphemes.

4.3 Schema of the allomorph method

```
surface:                       wolves
                                |   |     segmention
allomorphes:                   wolv/es    allomorph lookup and concatenation
                                ⇑   ⇑    derivation of allomorphs before run-time
morphemes & allomorphs:        wolf  s
```

[10] The inherent complexity of the morpheme method is shown in detail by Barton, Berwick and Ristad 1987, p. 115–186, using the analysis of spies/*spy+s* in the KIMMO system.

[11] See Lorenz & Schueller 1994.

[12] In the case of irregular paradigms, also the suppletive forms are supplied.

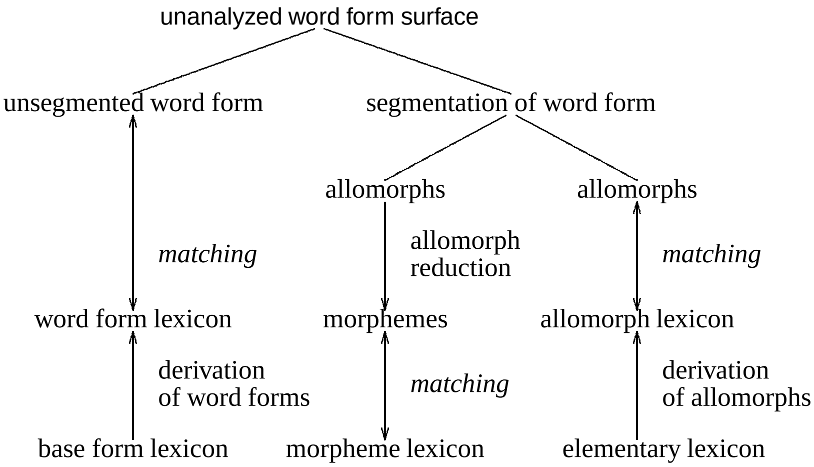
[13] Thus, no bound morphemes (cf. 2.4) are being postulated.

[14] MacWhinney 1978 demonstrates the independent status of lexical allomorphs with language acquisition data in Hungarian, Finnish, German, English, Latvian, Russian, Spanish, Arabic, and Chinese.

Concatenation takes place on the level of analyzed allomorphs by means of combi-rules. This method is in concord with the principle of surface compositionality.

In conclusion, the three basic methods of automatic word form recognition are compared schematically.

4.4 SCHEMATIC COMPARISON OF THE THREE METHODS



(1) word form method (2) morpheme method (3) allomorph method

All three methods are based on matching the input surface with a corresponding key of an analysis lexicon characteristic of the method in question. The first alternative consists in whether the word form surfaces are segmented into their elements during run-time or not. This alternative is relevant not only linguistically, but also of practical consequence for the computational implementation.

If the input surface is not segmented at all, we obtain the word form method, where the input is matched as a whole onto corresponding keys in an analysis lexicon consisting of analyzed word forms. In this case the well-known method of hash tables[15] may be used for lexical lookup, as long as the boundaries of each word form are marked, e.g., by spaces.

If the input surface is segmented, on the other hand, the concrete elements are the allomorphs. In order to automatically determine the allomorph boundaries – which are not orthographically marked – the method of trie structures is suited best. The question is, whether the allomorphs found in the surface should be reduced to morphemes prior to lexical lookup or whether lexical lookup should be defined for the allomorphs.

The morpheme method handles both, the segmentation of the surface into allomorphs and the reduction of the allomorphs into the associated (unanalyzed) morphemes during run-time. The morpheme surfaces obtained in this manner are then matched with corresponding keys in an analysis lexicon consisting of analyzed morphemes. The analyzed morphemes underlying the input are then put together again by grammatical rules to categorize and lemmatize the word form in question.

[15] See Aho & Ullman 1977, p. 336–341.

The allomorph method matches the input surface onto fully analyzed allomorphs during run-time. The analyzed allomorphs are generated automatically from an elementary lexicon by allo-rules before run-time. During run-time, the analyzed allomorphs need only be concatenated by means of rules which categorize and lemmatize the input.

Of the three methods, only the allomorph method satisfies the principles of surface compositionality and of time-linear derivation order. Furthermore, it is of low mathematical complexity (linear), describes morphological phenomena of concatenation and allomorphy in a linguistically transparent, rule-based manner, handles neologisms during run-time, may be applied easily to new languages, is computationally space and time efficient, and can be easily debugged and scaled up.

> Concluding remark: During the talk, an implementation of the allomorph method will be demonstrated.

References

Aho, A.V., B.W. Kerninghan, & P. Weinberger (1988) *The AWK Programming Language*, Addison-Wesley Publishing Company, Reading, Massachusetts.

Aho, A.V., & J.D. Ullman (1977) *Principles of Compiler Design*, Addison-Wesley Publishing Company, Reading, Massachusetts.

Barton, G., R.C. Berwick, & E.S. Ristad (1987) *Computational Complexity and Natural Language*, MIT Press, Cambridge, Massachusetts.

Koskenniemi, K. (1983) *Two-Level Morphology*, University of Helsinki, Publications, No. 11.

Lorenz, O., & G. Schüller (1994) “Präsentation von LA-MORPH,” in *LDV-Forum*, Band 11.1:39-51.

MacWhinney, B. (1978) *The Acquisition of Morphophonology*, Monographs of the Society for Research in Child Development, No. 174, Vol. 43.

Stemberger, P.J., & B. MacWhinney (1986) “Frequency and Lexical Storage of Regularly Inflected Forms,” *Memory and Cognition*, 14.1:17-26.

Saussure, F. de (1972) *Cours de linguistique générale*, Édition critique préparée par Tullio de Mauro, Éditions Payot, Paris.

Wall, L., & R.L. Schwartz (1990) *Programming Perl*, O’Reilly & Associates, Inc., Sebastopol, CA.