Morpholog: Constrained and Supervised Learning of Morphology

Rémi Zajac
Computing Research Laboratory, New Mexico State University
zajac@crl.nmsu.edu

Abstract

We present an automated method for building morphological analyzers using limited data elicited from a linguist. The system uses an induction procedure based on an extended edit distance algorithm to automate the discovery of morphological rules from examples. One of the requirements of the method is to minimize the size of the training data. The learning algorithm incorporates morphological language parameters, defined by the user, that constrains the space of the induction procedure. The result is a set of morphological rules that can be inspected by the user and further compiled into a morphological analyzer (or generator).

1 Introduction

We present an automated method for developing morphological analyzers for natural languages. The morphological analyzer generated by Morpholog is a component of an integrated NLP system such as a machine translation system. In such applications, the morphological analyzer is given an inflected word as input and produces the citation form of the word plus a set of morphosyntactic features which represent the morphological 'meaning' of inflectional morphemes. The morphological analyzer is dictionary-free and produces all possible analyses for a given word. The citation form is used to lookup words in a syntactic or a translation dictionary which can be maintained independently of the morphological system.

The system uses an induction procedure based on an extended edit distance algorithm to automate the discovery of morphological rules from examples. The training data consists of pairs of inflected forms and dictionary citation forms plus morphosyntactic features. Since such a training set is costly to develop by hand, one the requirements of the method is to minimize the size of the training data. The learning algorithm incorporates morphological language parameters, defined by the user, that constrains the learning space. The result of the learning procedure is a set of morphological rules in a format that can be inspected by the user and further compiled into a morphological analyzer and generator.

The Morpholog system is integrated in the Boas system for fast "ramp-up" of machine translation systems for minority languages at CRL (Nirenburg 98a; Nirenburg & Raskin 98). One assumption made for the Boas system is that there may be little or no machine readable resources for the source language (target is always English). Boas includes develop/test loop where citation forms are given to the morphological generator and results are presented to the user. When the user diagnoses erroneous forms, he correct these erroneous forms and the corrected forms are added to the learning set for the next learning iteration. The Boas system can be used by linguists with little knowledge of computational linguistics formalisms (e.g., finite-state technology). Although we expect that the set of rules induced from the training set will cover all intended phenomena, the Morpholog system allows expert users to inspect and modify the learned rules prior to compilation.

The next section reviews related work. Section 3 gives an overview of the system's architecture. Section 4 describes in more detail the induction procedure. Finally, we present some conclusions and ideas for future work.

2 Related Work

The automatic induction of morphological rules is a long standing problem that has been mentioned at least since (Garvin 67). Various authors have described, not always in great detail, approaches similar to the one presented in this paper, e.g. (Wothke 86; Brasington et als. 88). Other machine learning approaches to morphology have different goals. For example, (Goldsmith 98) presents an algorithm to infer stem and affixes from a large un-annotated corpus. However, the results do not allow to recover either the citation form of the word or associated inflectional features of word forms. (Gilloux 91) describes a method to induce a probabilistic finite-state transducer from examples. The application to grapheme-phoneme transcription seems relative straightforward. However, application to morphological analysis would need more extensive testing than the example presented in the paper. (Daelemans et als. 96) describes a system to learn phonological categories from a set of phonetic transcriptions of words, presumably for text-to-speech conversion.

The XMAS system (Zhang & Kim 90) is similar to the one described in this paper: the system is given a set of examples in the form of triples (*features, citation form, inflected form*) and learns morphological rules that can be interpreted by a morphological engine. The rule discovery procedure is not explained in detail but seems an improvement over (Wothke 86). Another similar approach is (Borin 91) who suggests to use an alignment algorithm for stem/affix identification, and phonological classes of graphemes to control generalization (as in (Zhang & Kim 90)). (Kuusik 96) describes a system for learning morphological rules for Estonian. The system distinguishes between affixation rules (so-called 'stem-end changes') and morphophonemic rules ('stem-grade changes'). The system combines ordered stem change rules (which combination seems to be equivalent to the use of an edit distance algorithm) and generalizations over morphemic classes (vowels, consonants) tailored for Estonian.

Finally, (Oflazer et als. 01) describe a a system which has been used in a previous version of the Boas system. The main differences between the approach described in this paper and (Oflazer et als. 01) is the use of a generalized edit distance which allows to identify stem and affixes in a way that can be constrained by user parameters (prefixation, infixation and suffixation).

3 Architecture

The three main components of the Morpholog system are the Morphological Knowledge Base, the Morphological Learner and the Rule Compiler. The user first populate the Morphological Knowledge Base (a set of XML files) by defining:

- The set of features and values that can appear in a morphological paradigm: part-of-speech, number, person, etc.
- The classes of the phonological segments (graphemes, composed of one or more characters) that are used to segment a word form into atomic segments. Basic classes are vowels, consonants, suprasegmentals (e.g., tone when written), and other written marks (apostrophe, etc.).
- The set of examples structured as a set of morphological paradigms. Each morphological paradigm consist of one main paradigm instance (the "primary" example) and optionally of several additional paradigm instances ("secondary" examples). Each paradigm instance, identified by the citation form, contains a set of pairs *(word form, set of features)* called paradigm members.

The Morphological Learner works on a single paradigm at a time and associate to each paradigm member a morphological rule. The Morphological Learner works as follows:

1. Each word form is aligned with the citation form to extract stem and affix candidates. The alignment procedure uses a generalized edit distance which allows for prefixation, suffixation and infixation.
2. All possible stem candidates are ranked by frequency and by weight (length of stem). At this step the user may select a stem candidate from the list suggested by Morpholog; he may also enter a stem which has not been produced.
3. Each stem candidate produces a segmentation of all instances in the paradigm. This step identifies affixes and generates segmentation rules for all instances.

4. Affixes are compared between primary and secondary examples for the same paradigm member to check for affix consistency.

5. When pure affixation does not work, morphophonemic adjustment of the stem (and possibly of affixes) are postulated. This step generate morphophonemic rules.

6. Segmentation and morphophonemic rules are combined into morphological rules associated to each paradigm member.

The rules are stored in the Morphological Knowledge Base and can be inspected (and modified) by the user. Once rules have been induced for all paradigms, the rule compiler produces rules in a format acceptable by a morphological engine. In the Boas system, we use a (reversible) unification-based morphological engine (Zajac 98).[1] This engine is integrated in the MEAT machine translation toolkit (Amtrup & Zajac 00) used to implement the Boas MT systems.

In Boas, the user (who is not assumed to know much about computational linguistic or linguistics) is carefully guided in the process of eliciting language data by a sequence of questions. The language data is then saved as XML files which are used as input by the Morpholog system. In addition, the Boas system implements an Elicit-Test-Build loop (Oflazer et als. 01) in which the resulting analyzer is tested by generating all possible forms for new words (using the morphological generation engine). The user checks, by inspecting the forms, that no gaps occur. If the user detect a gap, additional examples are provided to cover the gap (typically, morphophonological phenomena not covered by the primary examples).

4 The Morphological Discovery Procedure

Stem extraction. The stem extraction procedure operates on pairs of *(citation form, word form)* and returns a list of segmented and aligned pairs together with a weight which represent how closely a given segmentation aligns the two forms. A segmented pair contains the stem (possibly discontinuous), the list of affixes and the position of the stem in this list for all forms, and the weight associated to the segmentation.

The extraction algorithm pairs phonological segments an inflected form to phonological segments in the citation form, with the constraint that the pairs must not cross (but discontinuities between pairs are allowed). The stem is the list of paired segments. Since there will be of course many possible alignments, each aligned pair is weighted using the following heuristic, which favors the longest common continuous segment as the stem. The weight is simply the number of contiguous paired segments. To favor continuous stems, if there are 2 or more discontinuous alignments, the number of unpaired segments occurring between 2 contiguous paired sequences is subtracted.

The stem alignment algorithm uses an extended edit distance metric (Kruskal & Sankoff 99). The weights are based on consonant/vowel classifications: we prefer substitutions to indels (insertions/deletions) and we prefer substitutions in the same class (the best distance is the smallest one). We also favor consecutive indels and substitutions over independent indels and substitutions. Indels are given basic weight 3 and weight for substitutions is given by the following table:[2]

	V	S	C	M	O
V	1	2	3	4	4
S	2	1	2	4	4
C	3	2	1	4	4
M	4	4	4	1	4
O	4	4	4	4	1

The algorithm aligning a pair of *(citation form, word form)* can be visualized as a boolean matrix where rows and colons correspond to segments. For example, the alignment matrix for the Spanish *(enraizar, enraizaban)* is:[3]

```
  enraizaban
e 1000000000
n 0100000001
r 0010000000
```

[1]It should be fairly easy to produce rules in some other format –e.g., Xerox-type rules– as the rule compiler is written using XSLT.

[2]V: Vowel, S: Sonant, C: Consonant, M: Suprasegmental, O: Other.

[3]*to take root*: enraizaban is the indicative-preterit-imperfect-3rd-plu.

```
a 0001001010
i 0000100000
z 0000010000
a 0001001010
r 0010000000
```

The longest contiguous paired segments are represented by diagonals of 1s in the matrix. In this case:

```
|enraiza| r
|enraiza|   ban
```

All the other paired segments are either crossing or isolated (in this example, there are 6 other possible alignments, with diagonals reduced to one segment with weight 1), and there are no discontinuous sequences.

Stem ranking. The result of stem extraction is a list of aligned weighted of tuples *(stem, citation, word, weight)* where the citation form and the word form are segmented and the stem identified. This list is sorted by frequency of occurrence of the stem. When 2 elements have the same frequency, the elements are further ordered according to their weight. Let's assume the following paradigm instance for the citation form *torebka*:

```
torebka
torebki
torebce
torebke
torebka
torebko
torebek
torebkom
torebkami
torebkach
```

The list of stem candidates is:

Stem	Weight	Frequency
torebka	7	4
torebk	6	7
toreb	5	10
tore	4	10
tor	3	10
to	2	10
t	1	10

and the preferred stem candidate having the highest frequency and highest weight is `toreb` (which is not the correct stem in this case).

Paradigm segmentation. For a selected stem, and for each member of a paradigm instance, the pattern extraction procedure selects the segmentation that contain the stem. If there is no segmentation that contains that stem, the next most frequent stem is selected with the corresponding segmentation for that paradigm member. This step also extracts affix candidates. Suppose in the example above that the selected stem is *torebka*. The segmented paradigm identifying stem and affixes is (the + indicates that the stem from the paradigm segmentation is the same as the selected stem):

```
Stem      Weight Frequency
torebka   7      4
  + torebka
  - torebk  i
  - toreb   ce
  - torebk  e
  + torebka
  - torebk  o
  - toreb   ek
  - torebk  om
  + torebka mi
  + torebka ch
```

Affixation rules derived from this segmentation are:

```
Stem     Frequency Weight
torebka  4         7
  torebka     $1+ == $1+
  torebk  i   $1+ == $1+ @i
  toreb   ce  $1+ == $1+ @ce
  torebk  e   $1+ == $1+ @e
  torebka     $1+ == $1+
  torebk  o   $1+ == $1+ @o
  toreb   ek  $1+ == $1+ @ek
  torebk  om  $1+ == $1+ @om
  torebka mi  $1+ == $1+ @im
  torebka ch  $1+ == $1+ @ch
```

The notation for rule is as follows. Strings belonging to the stem are prefixed with $ and strings belonging to affixes are prefixed with @. Variable are numbers (e.g. `$1`), and `+` and `*` are used to express positive and free iteration. Constant strings are written as is (e.g., `@im` for the affix *im*).

Constraints on affixes. In the *torebka* example above, the selection of the most frequent stem does not lead to the correct choice. The correct segmentation should actually be based on additional evidence, i.e. on additional instances of the paradigm. In particular, the identity of affixes for two different segmented members in the same position (having with the same set of morphological features) should be a better criterion that frequency of stem alone. Identity of affixes across word forms is constitutive to the morphological definition of affixes and affixes should be identical (modulo morphophonemic changes) for all examples of a paradigm member in the same position. Therefore, when additional examples are available, they are segmented according to the procedure defined above. The number of identical (or similar) affixes is counted and this information is recorded in the segmented paradigm, and used as the first ranking criteria instead of stem frequency.

Morphophonemic rules. When there are different stems in a segmented paradigm instance, we need to account for the differences between the selected stem for the whole paradigm and the stem of a member, as it is the case in the example above (lines marked `-`).

1. If the selected stem is a sub-string of the stem member, the difference is simply reassigned to an affix candidate.
2. If the selected stem contains the stem member, we have a case of stem alternation (morphophonemic change at the boundary) and we need to produce a morphophonemic rule (see below).
3. An overlap is treated as a combination of the above
4. If the preferred stem is not comparable to another candidate (edit distance too great), we have a case of stem suppletion: either the choice of word is inadequate (and the user must provide another paradigm instance) or the suppletion is regular (and will be listed in the dictionary: no rule will be generate for this element).

In the example above, cases of stem alternations (marked `-`) must be reduced using morphophonemic rules. For example, to reconcile the segmentation `toreb|ce` (where `toreb` is the stem and `ce` is the suffix) with the selected stem `torebka`, we must posit a rule which removes `ka` at the end of the stem. These rules are built using an alignment algorithm similar to the one used for stem extraction (there is always only one morphophonemic rule generated for each paradigm member).

Since morphophonemic changes are typically triggered by specific morphophonemic contexts in stem and affixes, morphophonemic rules may include left and right contexts to restrict the application of the rule. The context is always generalized to the most specific morphophonemic class common to all segments involved when 3 or more different segments are found in the context (in the same position). If all segments are identical in a given position, no context is generated, as in the following example:

```
torebka   $1+      ==  $1+
torebk    $1+ @a   ==  $1+
toreb     $1+ @ka  ==  $1+
torebk    $1+ @a   ==  $1+
torebka   $1+      ==  $1+
torebk    $1+ @a   ==  $1+
toreb     $1+ @ka  ==  $1+
torebk    $1+ @a   ==  $1+
torebka   $1+      ==  $1+
torebka   $1+      ==  $1+
```

When secondary examples are provided, the context is specialized to be able to select a single rule based on the context.

Once morphophonemic rules are extracted, morphophonemic and affixation rules are then composed into a single rule per member:

```
$1+ $ka ==  $1+
$1+     ==  $1+ @ce
-------------------
$1+ $ka ==  $1+ @ce
```

These rules are recorded for each member and the whole paradigm pattern is:

```
Stem     Frequency Weight
torebka  4         7
  torebka    $1+     == $1+
  torebki    $1+ $a  == $1+ @i
  torebce    $1+ $ka == $1+ @ce
  torebke    $1+ $a  == $1+ @e
```

```
torebka     $1+     == $1+
torebko     $1+ $a  == $1+ @o
torebek     $1+ $ka == $1+ @ek
torebkom    $1+ $a  == $1+ @om
torebkami   $1+     == $1+ @im
torebkach   $1+     == $1+ @ch
```

Language parameters as morphological constraints. Although the algorithms for stem extraction and morphophonological induction allow almost any kind of change, we constrain the set of possible changes by filtering out phenonema which are not attested in the language. This filtering can be expressed as a set of patterns defining allowed changes. A pattern correspond to a language parameter that can be set by the user to allow/disallow classes of morphological phenomena. These include affixation (prefixation, suffixation, infixation, circumfixation) as well as morphophonemic patterns as, for example:

1. Stem alternation: one (or more) segment is changed into another segment(s). The rule is

   - $1* $3+ $2* == $1* $4+ $2*

   where $3 and $4 belong to the same colon or the same row of the vowel (or consonant) chart (they typically differ by one phonological feature).

2. Stem epenthesis: a segment is deleted or added.

   - Insertion: $1* $2* == $1* $0 $2*
   - Deletion: $1* $0 $2* == $1* $2*

   Typically, epenthesis occurs at the stem-affix boundary. Stem suffixal epenthesis is described by the following rules:

   - Insertion: $1+ @2+ == $1+ $0 @2+
   - Deletion: $1+ $0 2+ == $1+ @2+

   Stem prefixal epenthesis is described by the following rules:

   - Insertion: @1+ $2+ == @1+ $0 $2+
   - Deletion: @1+ $0 $2+ == @1+ $2+

We have similar rule for affix alternation and epenthesis. These rules cover the most frequent phenomena attested in a large set of languages and provide a good starting point for handling morphophonemic changes.

5 Conclusion

The Morpholog has been implemented and integrated in the Boas system. The user can build a morphological knowledge base (implemented as a set of XML files), select various parameters through the graphical interface, run the morphological learner (fully automatically or with an intermediate step to select a stem candidate), inspect the resulting rules, compile a morphological analyzer and run the analyzer.

Morpholog currently covers affixation (prefixes, suffixes, infixes and circumfixes), and stem alternation and stem epenthesis. The next version (which should be available soon) will cover affix alternation and epenthesis. It is currently being tested and refined on a large knowledge base for Polish and on smaller ones for Spanish and Arabic, but evaluation data is not available at the time of writing. Initial experiments show that the stem suggested by the system is correct in the majority of cases. However, convergence to a stable system depends a lot on the linguist definition of a paradigm. In particular, it is possible to define as paradigms inflectional paradigms that are only morphophonemic variants of each other, instead of conflating in a single paradigm all cases which have identical or similar affixes. Since the induction procedure works on the basis of a single paradigm, having a large number of paradigm variants will produce rules with little generalization. On the contrary, conflating morphophonemic variants in a single paradigm (where variants are defined as secondary examples) will produce rules with a higher degree of generality. Also, in this case, since there will be more examples for a single paradigm, the induction procedure is more likely to produce correct affixes.

This automated approach to the development of morphological analyzers offers several advantages over fully manual or fully automatic approaches. In comparison of fully manual approaches, our automated approach can be used by linguists who are not specialists of computational morphological formalisms. A linguist starts from a set of examples which are structured and presented in a way similar to traditional descriptions of morphology (conjugation/declension tables). Indeed, we started to develop some of our test examples by simply converting in XML published conjugation tables.

In comparison of fully automated approaches, the involvement of the user in the learning procedure requires a more transparent approach to learning. The benefits are that it allows to use smaller training sets, and also avoid the noise produced by some fully automatic schemes. The search space for string transformations is reduced to a set known morphological phenomena, and it can be further reduced by setting language specific parameters. Planned developments include support for agglutination and reduplication, as well as additional morphophonemic phenomena.

References

Amtrup, Jan W., & Rémi Zajac. 2000. "A Freely Available Toolkit for Machine Translation". *COLING-2000*, July 31-August 4 2000, Saarbrucken, Germany.

Borin, Lars. 1991. "The automatic induction of morphological regularities". Technical Report RUUL-22, Department of Linguistics, Uppsala University, Sweden.

Brasington, R., S. Jones, C. Biggs. 1988. The automatic induction of morphological rules. *Literary and Linguistic Computing* 3(2).

Daelemans, Walter, Peter Berck, Steven Gillis. 1996. "Unsupervised discovery of phonological categories through supervised learning of morphological rules". *COLING-96*, Copenhagen.

Garvin, Paul L. 1967. "The automation of discovery procedure in linguistics". *Language* 43(1), pp172-178.

Gilloux, M. 1991. "Automatic learning of word transducers from examples". *Fifth EACL*, Berlin.

Goldsmith, John. 1998."Unsupervised learning of the morphology of a natural language". *Unpublished ms.* See `http://humanities.uchicago.edu/faculty/goldsmith/`

Johnson,Mark. 1984. "A discovery procedure for creating phonological rules". *COLING-84*, pp344-347.

Koskenniemi, Kimmo. 1981. "A discovery procedure for two-level phonology". In Zampolli et als. (eds). *Computational Lexicology and Lexicography*, Vol.1. pp451-456.

Kruskal, Jospeh & David Sankoff. 1999. "An Anthology of Algorithms and Concepts for Sequence Comparison". In David Sankoff & Jospeh Kruskal (eds.) *Time Warps, String Edits, and Macromolecules. The Theory and Practice of Sequence Comparison.* CSLI Publications.

Kuusik, Evelin. 1996. "Learning morphology: algorithms for the identification of the stem changes". COLING-96.

Nirenburg, Sergei. 1998. "Project Boas: A Linguist-in-a-box as a multi-purpose language resource". *COLING-98.*

Nirenburg, Sergei, & Victor Raskin. 1998. "Universal grammar and lexis for quick ramp-up of MT systems". *LREC-98.*

Oflazer, Kemal, Sergei Nirenburg, Marjorie McShane. 2000. "Bootstrapping morphological analyzer by combining human elicitation and machine learning". *Computational Linguistics.* (to appear)

Pilato, S.F., R.C. Berwick. 1985. Reversible automata and the induction of the English auxiliary system. *23rd ACL*, Chicago.

Satta, Giorgo, & John Anderson. 1997. "String transformation Learning". *ACL/EACL-97.*

Tufis, D. 1989. "It would be much easier if Went were GOED". *4th EACL*, Manchester.

Wothke, K. 1986. "Machine learning of morphological rules by generalization and analogy". *11th COLING-86*, Bonn.

Zajac, Rémi. 1998. "Feature Structures, Unification and Finite State Transducers". *Proc. of the International Workshop on Finite State Methods in Natural Language Processing, FSMNLP-98*, Bilkent University, Ankara, Turkey. pp101-109.

Zhang, B.T., Y.-T.Kim. 1990. "Morphological analysis and synthesis by automated discovery and acquisition of linguistic rules". *13rd COLING-90*, Helsinki.