Bootstrapping Morphological Analyzers by Combining Human Elicitation and Machine Learning

Kemal Oflazer
Department of Computer Engineering
Bilkent University
Bilkent, Ankara, 06533, Turkey
ko@cs.bilkent.edu.tr

Sergei Nirenburg and Marjorie McShane
Computing Research Laboratory
New Mexico State University
Las Cruces, NM, 88003
{sergei,marge}@crl.nmsu.edu

January 21, 2000

Abstract

This paper presents a semi-automatic technique for developing broad-coverage finite-state morphological analyzers for use in natural language processing applications. It consists of three components – elicitation of linguistic information from humans, a machine learning bootstrapping scheme and a testing environment. The three components are applied iteratively until a threshold of output quality is attained. The initial application of this technique is for morphology of low-density languages in the context of the Expedition project at NMSU Computing Research Laboratory. This elicit-build-test technique compiles lexical and inflectional information elicited from a human into a finite state transducer lexicon and combines this with a sequence of morphographemic rewrite rules that is induced using transformation-based learning from the elicited examples. The resulting morphological analyzer is then tested against a test suite, and any corrections are fed back into the learning procedure that builds an improved analyzer.

1 Introduction

The Expedition project at NMSU Computing Research Laboratory is devoted to fast "ramp-up" of machine translation systems from less studied, so-called "low-density" languages into English. One of the components that must be acquired and built during this process is a morphological analyzer for the source low-density language. Since language informants are not expected or required to be well-versed in computational linguistics in general, or in recent approaches to building morphological analyzers (e.g., [Koskenniemi, 1983, Antworth, 1990, Karttunen *et al.*, 1992, Karttunen, 1994]) and the operation of state-of-the-art finite state tools (e.g., [Karttunen, 1993, Karttunen and Beesley, 1992, Karttunen *et al.*, 1996]) in particular, the generation of the morphological analyzer component has to be accomplished almost semi-automatically. The user will be guided through a knowledge elicitation procedure using the elicitation component of Expedition, the Boas system. As this task is not easy, we expect that the development of the morphological analyzer will be an iterative process, whereby the human informant will revise and/or refine the information previously elicited based on the feedback from test runs of the nascent analyzer.

The work reported in this paper describes the process of building and refining morphological analyzers using data elicited from human informants and machine learning. The main use of machine learning in our current approach is in the automatic learning of formal rewrite or replace rules for morphographemic changes derived from the examples provided by the informant. The subtask of accounting for morphographemic changes is perhaps one of the more complicated aspects of building an analyzer; by automating it we expect to gain a certain improvement in productivity.

This paper is organized as follows: After a review of related work, we very briefly describe the Boas project of which this work is a part. Subsequent sections describe the details of the approach, the morphological analyzer architecture, the elicited descriptive data and the computational processes on this data, including segmentation and the the induction of morphographemic rules. We then provide a very detailed example of applying this approach to developing a morphological analyzer for Polish. Finally, we provide some conclusions and ideas for future work.

2 Related Work

Machine learning techniques are widely employed in many aspects of language processing. The availability of large annotated corpora has fuelled a significant amount of work in the application of machine learning techniques to language processing problems, such as part-of-speech tagging, grammar induction, sense disambiguation, etc., as witnessed by recent workshops and journal issues dedicated to this topic.[1] The current work attempts to contribute to this literature by describing the human-supervised machine learning approach to the induction of morphological analyzers - a problem that, surprisingly, has received little attention.

There have been a number of studies on inducing morphographemic rules from a list of inflected words and a root word list. Johnson [1984] presents a scheme for inducing phonological rules from surface data, mainly in the context of studying certain aspects of language acquisition. The

[1] For instance, the CoNLL (Computational Natural Language Learning) Workshops, recent special issues of Machine Learning Journal (Vol 34 Issue 1/3, Feb 1999.), AI Magazine (Vol. 18, No. 4, 1997).

premise is that languages have a finite number of alternations to be handled by morphographemic rules and a fixed number of contexts in which they appear; so if there is enough data, phonological rewrite rules can be generated to account for the data. Rules are ordered by some notion of "surfaciness", and at each stage the most surfacy rule – the rule with the most transparent context – is selected. Golding and Thompson [1985] describe an approach for inducing rules of English word formation from a corpus of root forms and the corresponding inflected forms. The procedure described there generates a sequence of transformation rules,[2] each specifying how to perform a particular inflection.

More recently, Theron and Cloete [1997] have presented a scheme for obtaining two-level morphology rules from a set of aligned segmented and surface pairs. They use the notion of string edit sequences, assuming that only insertions and deletions are applied to a root form to get the inflected form. They determine the root form associated with an inflected form (and consequently the suffixes and prefixes) by exhaustively matching the inflected form against all root words. The motivation is that "real" suffixes will appear frequently in the corpus of inflected forms. Once common suffixes and prefixes are identified, the segmentation for an inflected word can be determined by choosing the segmentation with the most frequently occurring affix segments; the remainder is then considered the root. While this procedure seems to be reasonable for a small root word list, the potential for "noisy" or incorrect alignments is quite high when the corpus of inflected forms is large and the procedure is not given any prior knowledge of possible segmentations. As a result, automatically selecting the "correct" segmentation becomes quite nontrivial. An additional complication is that allomorphs show up as distinct affixes and their counts in segmentations are not accumulated, which might lead to actual segmentations being missed due to fragmentation. The rule induction is not via a learning scheme: aligned pairs are compressed into a special data structure and traversals over this data structure generate morphographemic rules. Theron and Cloete have experimented with pluralization in Afrikaans, and the resulting system has shown about 94% accuracy on unseen words.

Goldsmith [1998] has used an unsupervised learning method based on the minimum description length principle to learn the "morphology" of a number of languages. What is learned is a set of "root" words and affixes, and common inflectional pattern classes. The system requires just a corpus of words in a language. In the absence of any root word list to use as a scaffolding, the shortest forms that appear frequently are assumed to be roots, and observed surface forms are then either generated by the concatenative affixation of suffixes or by rewrite rules.[3] Since the system has no notion of what the roots and their part of speech values really are, and what morphological information is encoded by the affixes, these need to be retrofitted manually by a human who has to weed through a large number of noisy rules. We feel that this approach, while quite novel, can be used to build real-world morphological analyzers only after substantial

[2] Not in the sense it is used in transformation-based learning [Brill, 1995].

[3] Some of these rules may not make sense, but they are necessary to account for the data: for instance, a rule like *insert a word final y after the root "eas"*, is used to generate *easy*.

modifications are made.

3 The BOAS Project

Boas [Nirenburg, 1998, Nirenburg and Raskin, 1998] is a semi-automatic knowledge elicitation system that guides a team of two people (a language informant and a programmer) through the process of developing the static knowledge sources required to produce a moderate-quality, broad-coverage MT system from any "low-density" language into English. Boas contains knowledge about human language and means of realization of its phenomena in a number of specific languages as well as extensive pedagogical support, making the system a kind of "linguist in the box", intended to help non-professional acquirers with the task. In the spirit of the goal-driven, "demand-side" approach to computational applications of language processing [Nirenburg, 1996], the process of acquiring this knowledge has been split into two steps: (i) acquiring the descriptive, declarative knowledge about a language and (ii) deriving operational knowledge (content for the processing engines) from this descriptive knowledge.

An important goal that we strive to achieve regarding these descriptive and operational pieces of information, be they elicited from human informants or acquired via machine learning, is that they be *transparent*, *human readable*, and, where necessary, *human maintainable and extendable,* contrary to opaque and uninterpretable representations acquired by various statistical learning paradigms.

Before proceeding any further, we would also like to state the aims and limitations of our approach. Our main goal is to significantly expedite the development of a morphological analyzer. It is clear that for inflectional languages where each root word can be associated with a finite number of word forms, one can, with a lot of work, generate a list of word forms with associated morphological features encoded and use this as a look-up table to analyze word forms in input texts. This is, however, something we would like to avoid, as it is time consuming, expensive and error-prone. We would prefer to capture general morphophonological and morphographemic phenomena using sample paradigms that will be the basis of lexical abstractions. This will reduce the acquisition process to assigning *citation forms*[4] to one of the established paradigms; the automatic generation process described below will do the rest of the work. This process will still be imperfect, as we expect human informants to err in making their paradigm abstractions and to overlook details and exceptions. So, the whole process will be an iterative one, with convergence to a wide-coverage analyzer coming slowly at the beginning (where morphological phenomena and lexicon abstractions are being defined and tested), but significantly speeding up once wholesale root form acquisition starts. Since the generation of the operational content

[4] We use the term *citation form* to refer to the word form that is used to look up a given form in a dictionary. It may be the root or stem form that affixation is applied to, or it may have additional morphological markers to indicate its citation-form status.

(data files to be used by the morphological analyzer engine) from the elicited descriptions is expected to take only a few minutes, feedback on operational performance can be provided very fast.

Human languages have many diverse morphological phenomena and it is not our intent at this point to have a universal architecture that can accommodate any and all phenomena. Rather, we propose an extensible approach that can accommodate additional functionality in future incarnations of Boas. We also intend to limit the morphological processing to processing single tokens and to deal with multi-token phenomena, such as partial or full word reduplications, with additional machinery that we do not discuss here.

4 The Elicit-Build-Test Loop

In this paper we concentrate on operational content in the context of building a morphological analyzer. To determine this content, we integrate the information provided by the informant with automatically derived information. The whole process is an iterative one, as illustrated in Figure 1: the elicited information is transformed into the operational data required by the generic morphological analyzer engine;[5] the resulting analyzer is then tested on a test corpus.[6] Any discrepancies between the output of the analyzer and the test corpus are then analyzed and potential sources of errors are given as feedback to the elicitation process. Currently, this feedback is limited to morphographemic processes.

The box in Figure 1 labeled *Morphological Analyzer Generation* is the main component, which takes in the elicited information and generates a series of regular expressions for describing the morphological lexicon and morphographemic rules. The morphographemic rules describing changes in spelling as a result of affixation operations are induced from the examples provided by using transformation-based learning [Brill, 1995, Satta and Henderson, 1997]. The result is an ordered set of contextual replace or rewrite rules, much like those used in phonology.

4.1 Morphological Analyzer Architecture

We adopt the general approach advocated by Karttunen [1994] and build the morphological analyzer as the combination of several finite state transducers, some of which are constructed directly from the elicited information, and others of which are constructed from the output of the machine learning stage. Since the combination of the transducers is computed at compile time, there are no run time overheads. The basic architecture of the morphological analyzer is

[5] We currently use XRCE finite state tools as our target environment [Karttunen *et al.*, 1996].

[6] The test corpus is either elicited from the human informant or compiled from on-line resources for the language in question.

depicted in Figure 2. The components of this generic architecture are as follows. The analyzer consists of the union of transducers, each of which implements the morphological analysis process for one paradigm. Each such transducer is the composition of a number of components. These components are (from bottom to top) described below:

1. The bottom component is an ordered sequence of morphographemic rules that are learned via transformation-based learning from the sample inflectional paradigms provided by the human informant. These rules are then composed into one finite state transducer [Kaplan and Kay, 1994].

2. The *root and affix lexicon* contains the citation forms and the affixes. We currently assume that all affixation is concatenative and that the lexicon is described by a regular expression of the sort `[ Prefixes ]* [ Roots ] [ Suffixes ]*`.[7]

3. The *morpheme to surfacy feature mapping* essentially maps morphemes to feature names but retains some encoding of the surface morpheme. Thus, allomorphs that encode the same feature would be mapped to different "surfacy" features.

4. The *lexical and surfacy constraints* specify any conditions to constrain the possibly overgenerating morphotactics of the root and morpheme lexicon. These constraints can be encoded using the root morphemes and the surfacy features generated by the previous mapping. The use of surfacy features enables reference to zero morphemes, which otherwise could not have been used. For instance, if in some paradigm a certain prefix does not co-occur with a certain suffix, or always occurs with some other suffix, or if a certain root/lemma of that paradigm has exceptional behavior with respect to one or more of the affixes, or if the affixal allomorph that goes with a certain root depends on the properties of the root, these are encoded at this level as finite state constraints.

5. The *surfacy feature to feature mapping* module maps the surfacy representation of the affixes to symbolic feature names; as a result, no surface information remains except for the citation form. Thus, for instance, allomorphs that encode the same feature and map to different surfacy features now map to the same feature symbol.

6. The *feature constraints* specify constraints among the symbolic features. They are a different means of constraining morphotactics than the one provided by lexical and surfacy constraints. At this level, one refers to and constrains symbolic morphosyntactic features as opposed to surfacy features. This may provide a more natural or convenient abstraction, especially for languages with long-distance morphotactic constraints.

[7] We currently assume that we have at most one prefix and at most one suffix, but this is not a fundamental limitation. The elicitation of morphotactics for an agglutinating language like Turkish or Finnish requires a more sophisticated elicitation machinery.

These six finite state transducers are composed to yield a transducer for the paradigm. The union of the transducers for all paradigms produces one (possibly large) transducer for morphological analysis, where surface strings applied at the lower end produce all possible analyses at the upper end.

4.2 Information Elicited from Human Informants

The Boas environment guides the language informant through a series of questions leading up to paradigm delineation. The informant indicates the parameters for which a given part-of-speech inflects (e.g., Case, Number), the relevant values for those parameters (e.g., Nominative, Accusative; Singular, Plural), and the licit combinations of parameter values (e.g., Nominative Singular, Nominative Plural). Then he posits any number of paradigms, whose members are expected to show similar patterns of inflection. It is assumed that all citation forms that belong to the same paradigm take essentially the same set of inflectional affixes (perhaps subject to morphophonological variations). It is expected that the roots and/or the affixes may undergo systematic or idiosyncratic morphographemic changes. It is also assumed that certain lemmas in a given paradigm may behave in some exceptional way (for instance, contrary to all other lemmas, a given lemma may not have one of the inflected forms.) A paradigm description provides the full inflectional pattern for one characteristic or distinguished lemma and additional examples for any other lemmas whose inflectional forms undergo nonstandard morphographemic changes. If necessary, any lexical and feature constraints can be encoded. Currently the provisions we have for such constraints are limited to writing regular expressions (albeit at a much higher level than standard regular expressions); however, capturing such constraints using a more natural language (e.g., [Ranta, 1998]) can be stipulated for future versions.

4.3 Elicited Descriptive Data

Figure 3 presents the encoding of the information elicited for one paradigm of a Polish morphological analyzer, which will be covered in detail later.[8]

The data elicited using the user interface component of Boas is converted into a description text file with various components delineated by SGML-like tags. The components in the description are as follows:

- The `<LANGUAGE-DESCRIPTION...>` component lists information about the language and a

[8] Our actual system works using unicode character representation. But unicode input and output are not yet supported, hence we employ an ASCII internal representation for the unicode characters used for offline testing purposes. In the following examples, however, we have opted to represent the actual characters as they should appear in text.

specifies its vowels, consonants, and other orthographic symbols that do not fall into those two groups.

- A paradigm description starts with the tag `<PARADIGM NAME=...>`, which lists the name of the paradigm, its part of speech, and any additional morphosyntactic features that are common to all citation forms in this paradigm. In the example in Figure 3, the paradigm is for masculine nouns. Everything up to the `</PARADIGM>` tag is part of the descriptive data for the paradigm. This descriptive data consists of a primary example, a series of zero or more additional examples and the lexicon.
- The primary example is given between the `<PRIMARY-EXAMPLE>` and `</PRIMARY-EXAMPLE>` tags. The description is given as a sequence of one or more inflection groups between `<INF-GROUP>` `</INF-GROUP>` tags. In some instances, a given lexical item can use different stems (here called ”citation forms”) in different inflectional forms. For example, one stem might be used in the present tense and another in the past tense; or one might be used with multi-syllable affixes and another with single-syllable affixes. Thus, a given lexical item can have multiple citation forms, each of which gets associated with a mutually exclusive subset of inflectional forms. All the citation forms for a given lexical item plus all its inflectional forms are represented in an "inflection group". If the association of citation forms with inflectional forms is predictable (as indicted by the language informant), the subsets of inflectional forms are processed separately; if not, we assume that all citation forms can be used in all inflectional forms and hence overgenerate. Manual constraints can later be added, if necessary, to constrain this overgeneration.
- Additional examples are provided within `<EXAMPLE>` and `</EXAMPLE>` tags. Examples contain new citation forms plus any inflectional forms that are not predictable based on the primary example. Each example is considered an inflectional group and is enclosed within the corresponding tags.
- The citation forms given in the primary example and any additional examples are considered to be a part of the root lexicon of the paradigm definition. Any additional citation forms in this paradigm are listed between the `<LEXICON>` and `</LEXICON>` tags.

5 Generating the Morphological Analyzer

The morphological analyzer is a finite state transducer that is actually the union of the transducers for each paradigm definition in the description provided. Thus, the elicited data is processed one paradigm at a time. For each paradigm we proceed as follows:

1. The elicited primary citation form and associated inflected forms are processed to find the "best" segmentation of the forms into a *stem* and affixes. The stem is considered to be that part of the citation form onto which affixes are attached and has no function except for determining the affix strings. Although we allow for inflectional forms to have both a prefix and a suffix (one of each), we expect only suffixation to be employed by the inflecting languages with which we are dealing [Sproat, 1992].

2. Once the suffixes are determined, we segment the inflected forms for the primary example and any additional examples provided, and pair them with the corresponding surface forms. The segmented forms are now based on the citation form plus the affixes (not the stem). The reason is that we expect the morphological analyzer to generate the citation form as the lemma for further access to lexical databases to be used in the applications. The resulting segmented form - surface form pairs make up the example base of the paradigm.

3. The citation forms given in the primary example, additional examples and explicitly in the lexicon definition of the elicited data, and the mapping from suffix strings to the corresponding morphosyntactic features are compiled (by our morphological analyzer generating system) into suitable regular expressions (expressed using the regular expression language of the XRCE Finite State Tools [Karttunen *et al.*, 1996]).

4. The example base of the paradigm generated in step 2 is then used by a learning algorithm to generate a sequence of morphographemic rules [Kaplan and Kay, 1994] that handle the morphographemic phenomena.

5. The regular expressions for the lexicon in step 3 and the regular expressions for the morphographemic rules induced in step 4 are then compiled into finite state transducers and combined by composition to generate the finite state morphological analyzer for the paradigm.

The resulting finite state transducers for each paradigm are then unioned to give the transducer for the complete set of paradigms.

5.1 Determining Segmentation and Affixes

The suffixes and prefixes in a paradigm are determined by segmenting the inflected forms provided for the primary example. This process is complicated by the fact that the citation form may not correspond to the stem – it may contain a morphological indication that it is the citation form. Furthermore, since the language informant provides only a small number of examples, statistically motivated approaches like the one suggested by Theron and Cleoete [1997] are not applicable. We have experimented with a number of approaches and have found that the following approach works quite well.

Using the notion of description length [Rissanen, 1989], we try to find a stem and a set of affixes that account for all the inflected forms of the primary example. Let $C =< c_1, c_2, \ldots, c_c >$ be the character string for the citation form in the primary example (c_i are symbols in the alphabet of the language). Let $S_k =< c_1, c_2, \ldots, c_k >, 1 \leq k \leq c$ be a (string) prefix of C length k. We assume that the stem onto which morphological affixes are attached is S_k for some k.[9] The set of inflectional forms given in the primary example are $\{F_1, F_2, \ldots, F_f\}$, with each $F_j =< f_1^j, f_2^j, \ldots f_{l_j}^j >$ (f_i^j are symbols in the alphabet of the language and l_j is the length of the j^{th} form). The function $ed(v,w)$ (ed for edit distance), where v and w are strings, measures the minimum number of symbol insertions and deletions (but not substitutions) that can be applied to v to obtain w [Damerau, 1964].[10] We define

$$d(S_k) = k + \sum_{j=1}^{j=f} ed(S_k, F_j)$$

as a measure of the information needed to account for all the inflected forms. The first term above, k, is the length of the stem. The second term, the summation, measures how many symbols must be inserted and deleted to obtain the inflected form. The S_k with the minimum $d(S_k)$ is then chosen as the stem S. Creating segmentations based on stem S proceeds as follows: To determine the affixes in each inflected form $F_j =< f_1^j, f_2^j, \ldots f_{l_j}^j >$, we compute the projection of the stem $P_j =< f_b^j, \ldots f_e^j >$ in F_j, as that substring of F_j whose alignment with S provides the minimum edit distance, that is,

$$P_j = \underset{< f_{b'}^j, \ldots, f_{e'}^j >, 1 \leq b' < e' \leq l_j}{\operatorname{argmin}} ed(S, < f_{b'}^j, \ldots, f_{e'}^j >)$$

Then we select the substring $< f_1^j, \ldots, f_{b-1}^j >$ of F_j (if it exists), as the prefix, and $< f_{e+1}^j, \ldots, f_{l_j}^j >$ (if it exists) as the suffix. If there are multiple substrings of F_j that give the same (minimum) edit distance when aligned with S, we prefer the longer substring. We then create

$$(< f_1^j, \ldots, f_{b-1}^j + C + < f_{e+1}^j, \ldots, f_{l_j}^j >, F_j)$$

as an aligned segmented-surface pair and add it to the example base that we will use in the learning stage. Note that we now use the citation form C, and not the stem S, as a part of the segmented form.

Thus, at the end of the process we generate pairs of inflected forms and their corresponding segmented forms to be used in the derivation of the morphographemic rules. These pairs come

[9] The stem can also be an arbitrary substring of C, not just some initial prefix. Our approach can certainly extend to that.

[10] $ed(\ldots)$ assumes that vowels only align with other vowels or are elided, and consonants only align with consonants or are elided.

from both the inflected forms given in the primary example and from any additional examples given.

For example, suppose we have the following primary example

```
<PRIMARY-EXAMPLE>
<INF-GROUP>
             <PRIMARY-CIT-FORM FORM = "strona">
             <INF-FORM FORM = "strona" FEATURE = "Nom.Sg.">
             <INF-FORM FORM = "stronę" FEATURE = "Acc.Sg.">
             <INF-FORM FORM = "strony" FEATURE = "Gen.Sg.">
             <INF-FORM FORM = "stronie" FEATURE = "Dat.Sg.">
             <INF-FORM FORM = "stronie" FEATURE = "Loc.Sg.">
             <INF-FORM FORM = "stroną" FEATURE = "Instr.Sg.">
             <INF-FORM FORM = "strony" FEATURE = "Nom.Pl.">
             <INF-FORM FORM = "strony" FEATURE = "Acc.Pl.">
             <INF-FORM FORM = "stron" FEATURE = "Gen.Pl.">
             <INF-FORM FORM = "stronom" FEATURE = "Dat.Pl.">
             <INF-FORM FORM = "stronach" FEATURE = "Loc.Pl.">
             <INF-FORM FORM = "stronami" FEATURE = "Instr.Pl.">
</INF-GROUP>
</PRIMARY-EXAMPLE>
```

For this example, stems S_k: *s*, *st*, *str*, *stro*, *stron*, *strona*, are considered. Table 1 tabulates $d(S_k)$ considering all the unique inflected forms above. It can be seen that the value of $d(S_5)$ is minimum for $S_5 = S = stron$. We then determine suffixes based on this stem selection. The suffixes are given in this table under $k = 5$, where the stem $S = stron$ perfectly aligns with the initial substring *stron* in each inflected form F_j, with 0 edit distance.

The segmented - surface pairs in Table 2 are then generated from the alignment of the stem with each surface form.

5.2 Learning Segmentation and Morphographemic Rules

The lemma and suffix information elicited and extracted by the process described above are used to construct regular expressions for the lexicon component of each paradigm.[11] The example segmentations are fed into the learning module to induce morphographemic rules.

[11] The result of this process is a script for the XRCE finite state tool *xfst*. Large scale lexicons can be more efficiently compiled by the XRCE tool *lexc*. We currently do not generate *lexc* scripts, but it is trivial to do so.

Table 1: Stems S_k and the corresponding $d(S_k)$

	Stems Considered, S_k $ed(S_k, F_j)$						
	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$		$k = 6$
Form F_j	s	st	str	stro	**stron**	Suffix	strona
strona	5	4	3	2	1	-*a*	0
stronę	5	4	3	2	1	-*ę*	2
strony	5	4	3	2	1	-*y*	2
stronie	6	5	4	3	2	-*ie*	3
stroną	5	4	3	2	1	-*ą*	2
stron	4	3	2	1	0		1
stronom	6	5	4	3	2	-*om*	3
stronach	7	6	5	4	3	-*ach*	2
stronami	7	6	5	4	3	-*ami*	2
$d(S_k)$	51	43	35	27	**19**		23

Table 2: The segmented and surface pair examples obtained

Segmented	**Surface**
strona+a	strona
strona+ę	stronę
strona+y	strony
strona+ie	stronie
strona+ą	stroną
strona+	stron
strona+om	stronom
strona+ach	stronach
strona+ami	stronami

5.2.1 Generating Candidate Rules from Examples

The preprocessing stage yields a list of pairs of *segmented lexical forms* and *surface forms*. The segmented forms contain the citation forms and affixes; the affix boundaries are marked by the `+` symbol. This list is then processed by a transformation-based learning paradigm [Brill, 1995, Satta and Henderson, 1997], as illustrated in Figure 4. The basic idea is that we consider the list of segmented words as our input and find transformation rules (expressed as contextual rewrite rules) to incrementally transform it into the list of surface forms. The transformation we choose at every iteration is the one that makes the list of segmented forms closest to the list of surface forms.

The first step in the learning process is an initial alignment of pairs using a standard dynamic programming scheme. The only constraints in the alignment are (i) a + in the segmented lexical form is always aligned with an empty string on the surface side, notated by `0`; (ii) a consonant on one side is always aligned with a consonant or `0` on the other side, and likewise for vowels; (iii) the alignment must correspond to the minimum edit distance between the original lexical and surface forms.[12] From this point on, we will use a simple example from English to clarify our points.

We assume that we have the pairs `(un+happy+est, unhappiest)` and `(shop+ed, shopped)` in our example base. We align these and determine the total number of "errors" in the segmented forms that we have to fix to make all segmented forms match the corresponding surface forms. The initial alignment produces the aligned pairs:

$\texttt{un+happy+est} \qquad \texttt{shop0+ed}$

$\texttt{un}\underline{\texttt{0}}\texttt{happ}\underline{\texttt{i0}}\texttt{est} \qquad \texttt{shop}\underline{\texttt{p0}}\texttt{ed}$

with a total of 5 errors. From each segmented pair we generate rewrite rules of the sort[13]

```
u -> l || LeftContext _ RightContext ;
```

where `u`(pper) is a symbol in the segmented form, `l`(ower) is a symbol in the surface form. Rules are generated only from those aligned symbol pairs that are different. `LeftContext` and `RightContext` are simple regular expressions describing contexts in the segmented side (up to some small length), taking into account also the word boundaries. For instance, from the first aligned-pair example, this procedure would generate rules such as (depending on the amount of

[12] We arbitrarily choose one if there are multiple legitimate alignments.

[13] We use the XRCE Finite State Tools regular expression syntax [Karttunen *et al.*, 1996]. For the sake of readability, we will ignore the escape symbol (`%`) that should precede any special characters (e.g., `+`) used in these rules.

left and right context allowed)

```
y -> i ||  p _                        y -> i ||    p _ + e
y -> i ||  p _ + e s                  y -> i ||    p _ + e s t
y -> i ||  p _ + e s t #              y -> i || p  p _ + e
. . .                                 . . .
+ -> 0 ||  # u n  _                   + -> 0 || # u n  _  h a p
+ -> 0 ||         _  e s t
. . .
+ -> 0 ||  _  e s t #   . . .
+ -> 0 ||  p p y  _  e s t #
```

The # symbol denotes a word boundary and is intended capture any word-initial and word-final phenomena. The segmentation rules (+ -> 0) require at least some minimal left or right context (usually longer than the minimal context for other rules in order to produce more accurate segmentation decisions). We disallow contexts that consist only of a morpheme boundary, as such contexts are usually not informative. It should be noted that these rules transform a segmented form into a surface form (contrary to what may be expected for analysis). This lets us capture situations where multiple segmented forms map to the same surface form, which occurs when the language has morphological ambiguity. Thus, in a reverse look-up, a given surface form may be interpreted in multiple ways, if applicable.

Since we have many examples of aligned pairs in our example base, it is likely that a given rule will be generated from many pairs. For instance, if the pairs (stop+ed, stopped) and (trip+ed, tripped) were also in the list, the gemination rule 0 -> p || p _ + e d (along with certain others) will also be generated from these examples. We count how many times a rule is generated and associate this number with the rule as its *promise*, meaning that it promises to fix this many "errors" if it is selected to apply to the current list of segmented forms.

5.2.2 Generalizing Rules

The candidate rules generated by the processes described above refer to specific strings of symbols as left and right contexts. It is, however, possible to obtain more generalized rules by classifying the symbols in the alphabet into phonologically relevant groups, like vowels and consonants. The benefit of this approach is that the number of rules thus induced is typically smaller, and more unseen cases can be covered.

For instance, in addition to a rule like 0 -> p || p _ + e, the rules

```
0 -> p  ||   CONSONANTS _ + e
```

```
0 -> p  ||   p _ + VOWELS
. . .
0 -> p  ||   CONSONANTS _ + VOWELS
```

can be generated, where symbols such as CONSONANTS and VOWELS stand for regular expressions denoting the union of relevant symbols in the alphabet. The promise scores of the generalized rules are found by adding the promise scores of the original rules generating them. Generalization substantially increases the number of candidate rules to be considered during each iteration, but this is not a very serious issue, as the number of examples per paradigm is expected to be quite small. The rules thus learned would be *the most general set of rules that do not conflict with the evidence in the examples.* It is possible to use a more refined set of classes that correspond to subclasses of vowels (e.g., high-vowels) and consonants (e.g., fricatives) but these will substantially increase the number of candidate rules at every iteration and will have an impact on the iteration time.

5.2.3 Selecting Rules

At each iteration all the rules along with their promise scores are generated from the current state of the example pairs. The rules generated are then ranked based on their promise scores, with the top rule having the highest promise. Among rules with the same promise score, we rank more general rules higher, with generality being based on context subsumption (i.e., preference goes to rules using shorter contexts and/or referring to classes of symbols, like vowels or consonants). All segmentation rules go to the bottom of the list, though within this group, rules are still ranked based on decreasing promise and context generality. The reasoning for treating the segmentation rules separately and later in the process is that affixation boundaries constitute contexts for all morphographemic changes; therefore they should not be eliminated if there are any (more) morphographemic phenomena to process.

Starting with the top ranked rule, we test each rule on the segmented component of the pairs. A finite state engine emulates the replace rules to see how much the segmented forms are "fixed". The first rule that fixes as many "errors" as it promises to fix, and does not generate an interim example base with generation ambiguity, is selected.[14] The issue of generation ambiguity refers to cases where the same segmented forms are paired up with distinct surface forms.[15] In such cases, finding a rule that fixes both pairs is not possible, so in choosing rules, we avoid any rules

[14] Note that a rule may actually introduce unintended errors in other pairs, since context checking is done only on the segmented form side; therefore what a rule delivers may be different than what it promises, as promise scores also depend on the surface side.

[15] Consider a state of the example base where some segmented lexical form L is paired up with different surface forms S_1 and S_2, that is, we have pairs (L, S_1) and (L, S_2) in our example base. Any rule that will bring L closer to S_1 will also change L of the second pair and possibly make it impossible to bring it closer to S_2.

whose tentative application generates an interim example base with such ambiguities. In this way we can account for all the discrepancies between the surface and segmented forms without falling into a local minima. Although we do not have formal proof that this simple heuristic avoids such local minima situations, in our experimentation with a large number cases we have never seen such an instance.

The complete procedure for rule learning can now be given as follows:

```
- Align surface and segmented forms in the example base;

- Compute total Error;

- while(Error > 0) {

   -Generate all possible rewrite rules subject to context size limits;


   -Rank Rules;

   -while (there are more rules and a rule has not yet been selected) {

          - Select the next rule;

          - Tentatively apply rule to  all the segmented forms;

          - Re-align the resulting segmented forms with the
            corresponding surface forms to see how many
            ``errors'' have been fixed;

          - If the number of errors fixed is equal to what the rule promised
            to fix AND the result does not have generation ambiguity,
            select this rule;
      }

    -Commit the changes performed by the rule on the segmented forms
     to the example base;

    -Reduce Error by the promise score of the selected rule;
  }
```

This procedure eventually generates an ordered sequence of two ordered groups of rewrite rules. The first group of rules is for any morphographemic phenomena in the given set of examples, and the second group of rules handles segmentation. All these rules are composed in the order in which they are generated to construct the *Morphographemic Rules* transducer at the bottom of each paradigm (see Figure 2).

5.3 Identifying Errors and Providing Feedback

Once the *Morphographemic Rules* transducers are compiled and composed with the lexicon transducer that is generated automatically from the elicited information, we obtain an analyzer for the paradigm. The analyzer for the paradigm can be tested by using the *xfst* environment of the XRCE finite state tools. This environment provides machinery for testing the output of the analyzer by generating all forms involving a specific citation form, a specific morphosyntactic feature, etc. This kind of testing has proved quite sufficient for our purposes.

When the full analyzer is generated by unioning all the analyzers for each paradigm, one can do a more comprehensive test against a test corpus to see what surface forms in the test corpus are not recognized by the generated analyzer. Apart from revealing obvious deficiencies in coverage (e.g., missing citation forms in the lexicon), such testing provides feedback about minor human errors – the failure to cover certain morphographemic phenomena, the incorrect assignment of lemmas to paradigms, etc.

Our approach is as follows: we use the resulting morphological analyzer with an error-tolerant finite state recognizer engine [Oflazer, 1996]. Using this engine, we try to find words recognized by the analyzer that are (very) close to a rejected (correct) word in the test corpus, essentially performing a reverse spelling correction. If the rejection is due to a small number (1 or 2) of errors, the erroneous words recognized by the recognizer are aligned with the corresponding correct words from the test corpus. These aligned pairs can then be analyzed to see what the problems may be.

6 An Example: Bootstrapping a Polish Analyzer

This section presents a quite extensive example of bootstrapping a morphological analyzer for Polish by iteratively providing examples and testing the morphological analyzer systematically. The idea of this exercise was to have a relatively limited number of paradigms that bunched words showing slight inflectional variations.[16] For reasons of space, the exposition is limited to developing six paradigms, of which two will be covered in detail. The paradigms here cover

[16] Non-expert language informants using Boas will be encouraged to split, rather than bunch, paradigms, for the sake of simplicity.

only a subset of masculine nouns, and don't treat feminine or neuter nouns at all; however, they cover all the problems that would be found in words of those genders.

For purposes of testing to learner offline (i.e., outside the Boas environment), we tried to keep to a minimum of the number of inflected forms given for each additional citation form. This was a learner-oriented task and intended to determine how robust the learner could become with a minimum of input. However, when the language informant is using the interface, he will have the option of selectively giving inflected forms. The interface works as follows: the user gives all forms of the primary example and lists other citation forms that he thinks belong to the given paradigm. Having learned rules from the primary example, the learner generates all the inflectional forms for each citation form provided. The user then corrects all mistakes and the learner relearns is the rules. So, the user never has the opportunity to say "Well, I know the learner can't know the locative singular for this word, so, I will it overtly so I will supply it overtly from the outset." The user will just have to wait for the learner to get the given forms wrong and then correct them. Any other approach would make for a complex interface and would require a sophisticated language informant – not what we're expecting.

Polish is a highly inflectional West Slavic language that is written using extended Latin characters (6 consonants and 3 vowels have diacritics). Certain phonemes are written using combinations of letters: e.g., cz, sz and szcz all represent hushers.[17] analysis). Polish nominals inflect for seven cases: Nominative (Nom.), Accusative (Acc.), Genitive (Gen.), Dative (Dat.), Locative (Loc.), Instrumental (Instr.), Vocative (Voc.), and two numbers: Singular (Sg.), Plural (Pl.).[18] The complexity of Polish declension derives from four sources: (i) certain stem-final consonants mutate during inflection; these are called "alternating" consonants, and are contrasted with so-called "non-alternating" consonants ("alternating"/"non-alternating" is a crucial diagnostic for paradigm delineation in Polish); (ii) certain letters are spelled differently when they are word-final versus when they are word-internal (e.g., word-final *-ś* is written *-si* when followed by a vocalic ending); (iii) final-syllable vowels are added/deleted in some (not entirely predictable) words and (iv) declension is not entirely phonologically driven – semantics and idiosyncrasy affect inflectional endings.

The following practical simplifications have been made for testing purposes:

- Words that are normally capitalized (like names) are not capitalized here.
- Some inflectional form(s) that might not be semantically valid (e.g., plurals for collectives) were disregarded. Thus a bit of overgeneration still remains but can certainly be removed with some additional effort.

[17] We actually treat these as single symbols during learning. Such symbols are indicated in the description file in a special section that we have omiited in Figure 3.

[18] The vocative case was not included in these tests because it is not expected to occur widely in the journalistic prose for which the system is being built.

6.1 Paradigm 1

The process starts with the description of Paradigm 1, which describes *alternating inanimate masculine nouns with genitive singular in -u and no vowel shifts.* The following primary example for the citation form *telefon* is given in full:

	Number	
Case	**Singular**	**Plural**
Nom.	telefon	telefon**y**
Acc.	telefon	telefon**y**
Gen.	telefon**u**	telefon**ów**
Dat.	telefon**owi**	telefon**om**
Loc.	telefon**ie**	telefon**ach**
Instr.	telefon**em**	telefon**ami**

All inflectional forms in this paradigm are trivial, except:

- The *Loc.Sg.* depends on the final consonant and induces alternations for alternating consonants:

Final Consonant(s)	**Loc.Sg. Ending**	**Consonant Alternations**
b, p, f, w, m, n, s, z	-ie	
t, d, st, zm	-ie	t→c, d→dz, st→śc, zm→źm
ł, r, sł	-e	ł→l, r→rz, sł→śl
g, k, ch	-u	

- *Instr.Sg.* and *Nom.Pl.* depend upon the final consonant; two velars have an idiosyncratic ending:

Final Consonant(s)	**Instr.Sg.**	**Nom.Pl.**
	Ending	**Ending**
b, p, f, w, m, n, s, z	-em	-y
t, d, st zm, ł, r, sł, ch		
g, k	-iem	-i

The following examples were provided *in addition* to the inflectional forms of the primary example in order to show *Loc.Sg.* endings and accompanying consonant alternations:

1. *t→c*: *akcent* (*Nom.Sg.*), *akcencie* (*Loc.Sg.*)

2. *d → dz*: *wykład* (*Nom.Sg.*), *wykładzie*(*Loc.Sg.*)

3. *st* →*śc*: *most* (*Nom.Sg.*), *moście*(*Loc.Sg.*)
4. *zm*→*źm*: *komunizm* (*Nom.Sg.*), *komuniźmie* (*Loc.Sg.*)
5. *ł*→*l*: *artykuł* (*Nom.Sg.*), *artykule* (*Loc.Sg.*)
6. *r*→*rz*: *teatr* (*Nom.Sg.*), *teatrze* (*Loc.Sg.*)
7. *sł*→*sl*: *pomysł* (*Nom.Sg.*), *pomyśle* (*Loc.Sg.*)

The following additional examples were provided to show velar pecularities:

8) *g*: *pociąg* (*Nom.Sg.*), *pociągu* (*Loc.Sg.*), *pociągiem* (*Instr.Sg.*), *pociągi* (*Nom.Pl.*)

9) *k*: *bank* (*Nom.Sg.*), *banku* (*Loc.Sg.*), *bankiem* (*Instr.Sg.*), *banki* (*Nom.Pl.*)

10) *ch*: *dach* (*Nom.Sg.*), *dachu* (*Loc.Sg.*)

Table 3 summarizes the first 3 runs for this paradigm, which were sufficient to create a relatively robust set of morphological rules that required only slight amendment and further testing in 2 additional runs. For this and subsequent such tables we use the following conventions. Key 0 shows the primary citation form and additional citation forms whose inflectional patterns should be fully covered by the rules generated for the primary example. The other key numbers correspond to the additional examples given above. Boldface citation forms under the lexicon column are those for which some additional inflectional examples were given. Oblique cases refer to the genitive, dative, locative and instrumental cases.

The original assumption for paradigm 1 was that it would be sufficient to provide one unmutated form (the *Nom.Sg.*) plus the mutated form (the *Loc.Sg.*) for words ending in mutating consonants. This lead to overgeneralization of the mutation; therefore, another unmutated form had to be added as a "control". Adding the *Nom.Pl.* forms fixed most oblique forms for all the words, but it left the *Instr.Sg.* mutated. This appears to be because the inflectional ending for the *Loc.Sg.* (which mutates) and the *Instr.Sg.* (which doesn't) both begin in *-e* for the words in question. Adding the *Instr.Sg.* overtly solved the problem of overgeneralizing the mutation. The source of the velar errors is not immediately evident.

Supplementary testing was carried out after the above-mentioned words were all correct. Correct forms were produced for all new words showing consonant mutations and velar peculiarities: *samolot, przykład, pretekst, podział, kolor, dług, lek, gmach.* One error for a non-mutating word (in Key 0) occurred. This word, *herb*, ends in a different consonant than the primary example and produced the wrong *Loc.Sg.* form. This was later added overtly and more words with other non-mutating consonants (*postęp, puf, gniew, film, opis, raz*) were tested; all were covered correctly.

Table 3: Summary of Runs for Paradigm 1

Key	**Citation Forms**	**Add'l Examp.**	**Run 1 Results**	**Add'l Examp.**	**Run 2 Results**	**Add'l Examp.**	**Run 3 Results**
0	**telefon**, stron, paragraf, śpiew, sklep, tłum, adres, obraz		√				
1	**akcent**, bilet	*Nom.Sg.* *Loc.Sg.*	mutates *all* obl. forms	*Nom.Pl.*	mutates *Instr.Sg.*	*Instr.Sg.*	√
2	**wykład**, sad	*Nom.Sg.* *Loc.Sg.*	mutates *all* obl. forms	*Nom.Pl.*	mutates *Instr.Sg.*	*Instr.Sg.*	√
3	**most**, list	*Nom.Sg.* *Loc.Sg.*	mutates *all* obl. forms	*Nom.Pl.*	mutates *Instr.Sg.*	*Instr.Sg.*	√
4	**komunizm**, socjalizm	*Nom.Sg.* *Loc.Sg.*	mutates *all* obl. forms	*Nom.Pl.*	mutates *Instr.Sg.*	*Instr.Sg.*	√
5	**artykuł**, kawał	*Nom.Sg.* *Loc.Sg.*	mutates *all* obl. forms	*Nom.Pl.*	mutates *Instr.Sg.*	*Instr.Sg.*	√
6	**teatr**, numer	*Nom.Sg.* *Loc.Sg.*	mutates *all* obl. forms	*Nom.Pl.*	mutates *Instr.Sg.*	*Instr.Sg.*	√
7	**pomysł**, zmysł	*Nom.Sg.* *Loc.Sg.*	mutates *all* obl. forms	*Nom.Pl.*	mutates *Instr.Sg.*	*Instr.Sg.*	√
8	**pociąg**, brzeg	*Nom.Sg.* *Loc.Sg.* *Instr.Sg.* *Nom.Pl.*	√				
9	**bank**, krok	*Nom.Sg.* *Loc.Sg.* *Instr.Sg.* *Nom.Pl.*	missed velar-specific *Loc.Sg.* gave *krokie not kroku	*Loc.Sg.* of krok, Add błysk to lexicon for testing	√		
10	**dach**, wirch	*Nom.Sg.* *Loc.Sg.* *Nom.Pl.*	missed velar-specific *Loc.Sg.* gave *wirchie not wirchu	*Loc.Sg.* of wirch, Add śmiech to lexicon for testing	wrong *Instr.Sg.* for wirch, śmiech	add *Instr.Sg.* of wirch	√

6.2 Paradigm 2

The second paradigm that we considered was Paradigm 2: *alternating inanimate masculine nouns with genitive singular in -a and no vowel shifts.* The following primary example for the citation form *chleb* was given in full:

	Number	
Case	**Singular**	**Plural**
Nom.	chleb	chleb**y**
Acc.	chleb	chleb**y**
Gen.	chleb**a**	chleb**ów**
Dat.	chleb**owi**	chleb**om**
Loc.	chleb**ie**	chleb**ach**
Instr.	chleb**em**	chleb**ami**

This paradigm is just like Paradigm 1 except that: (i) the *Gen.Sg.* form ends in *-a*, not *-u* (an unpredictable fact) and (ii) fewer words belong to this paradigm so not all consonants arise in stem-final position.

Based on the experience of Paradigm 1, the following change in approach was made from the outset: for all words with consonant alternations, the *Nom.Sg.*, *Loc.Sg.*, and *Instr.Sg.* forms were provided explicitly during the first run (to counter overgeneration of the mutation).

The following examples were provided to show unpredictable *Loc.Sg.* endings and accompanying consonant mutations:

1. *t*→*c*: *funt* (*Nom.Sg.*), *funcie* (*Loc.Sg.*), *funtem* (*Instr.Sg.*)
2. *d*→*dz*: *listopad* (*Nom.Sg.*), *listopadzie* (*Loc.Sg.*), *listopadem* (*Instr.Sg.*)
3. *r*→*rz*: *sznur* (*Nom.Sg.*), *sznurze* (*Loc.Sg.*), *sznurem* (*Instr.Sg.*)

The following examples were provided to show velar peculiarities:

4) *g*: *pług* (*Nom.Sg.*) *pługu* (*Loc.Sg.*), *pługiem* (*Instr.Sg.*), *pługi* (*Nom.Pl.*)

5) *k*: *język* (*Nom.Sg.*), *języku* (*Loc.Sg.*), *językiem* (*Instr.Sg.*), *języki* (*Nom.Pl.*)

6) *ch*: *brzuch* (*Nom.Sg.*), *brzuchu* (*Loc.Sg.*)

The runs for Paradigm 2 proceeded as in Table 4. As mentioned earlier, the *Instr.Sg.* forms were added as controls from the outset to counter overgeneralization of the mutation. This

Table 4: Summary of Runs for Paradigm 2

Key	Citation Forms	Add'l Examp.	Run 1 Results	Add'l Examp.	Run 2 Results
0	**chleb**, czub, chlew, ogon, nos, analiz, kaktus		Incorrect *Loc.Sg.* for chub and kaktus	*Loc.Sg.* of chub and kaktus	√
1	**funt**, kąt	*Nom.Sg.*, *Loc.Sg.* *Instr.Sg.*	√		
2	**listopad**	*Nom.Sg.*, *Loc.Sg.* *Instr.Sg.*	√		
3	**sznur** buldożer	*Nom.Sg.*, *Loc.Sg.* *Instr.Sg.*,	Incorrect *Loc.Sg.* of buldożer	*Loc.Sg.* of buldożer	√
4	**pług** bumerang	*Nom.Sg.*, *Loc.Sg.* *Instr.Sg.*, *Nom.Pl.*	Incorrect *Loc.Sg.* of buldożer	*Loc.Sg.* of buldożer	√
5	**język** chodnik	*Nom.Sg.*, *Loc.Sg.* *Instr.Sg.*, *Nom.Pl.*	√		
6	**brzuch** kożuch	*Nom.Sg.*, *Loc.Sg.* *Nom.Pl.*	√		

worked well. There were some problems with the rules for the *Loc.Sg.*, which were subsequently fixed by adding a few additional *Loc.Sg.* forms overtly. Supplementary testing after the above-mentioned words were all correct showed that the analyzer worked correctly for this paradigm. The supplementary words tested were: *piorun, obrus, guz, but, akcelerator, batog, dziennik, łańcuch.*

6.3 Paradigm 3

The paradigm implemented next was Paradigm 3: *alternating inanimate masculine nouns with genitive singular in -u and vowel shifts.* The following primary example for the citation form *grób* was given in full:

	Number	
Case	**Singular**	**Plural**
Nom.	grób	grob**y**
Acc.	grób	grob**y**
Gen.	grob**u**	grob**ów**
Dat.	grob**owi**	grob**om**
Loc.	grob**ie**	grob**ach**
Instr.	grob**em**	grob**ami**

This paradigm is just like Paradigm 1, except that there are vowel shifts that are not entirely graphotactically predictable; therefore, words showing these shifts must be classed separately. The vowel shifts occur in all inflectional forms except the *Nom.Sg.* and the *Acc.Sg.*, which are identical. The following vowel shifts occurred in the cases we considered (ϕ indicates vowel deletion).

Vowel in Nom.Sg./Acc.Sg.	**Vowel in Other Forms**
ó	o
e	ϕ
ie	ϕ
a	e*

* This shift only occurs in *Loc.Sg.*

The following consonant shifts are also observed in this paradigm:

Cons. in Most forms	**Cons. in Loc.Sg.**
d	dz
dz	ździe
ł	l
r	rz

Based on the experience of paradigms 1 and 2, the *Instr.Sg.* forms for all shifting words were provided as examples at the outset to avoid the overgeneralization of the consonant mutation. The velar pecularities are still in effect and must be dealt with explicitly.

The following examples were given to exemplify vowel shifts with an unmutating consonant:

1) *e* → ϕ shift with *n*: *sen* (*Nom.Sg.*), *snie* (*Loc.Sg.*)

The following examples were employed to show vowel shifts in combination with various consonant mutations in the *Loc.Sg.* forms:

2) *ó* → *o* and *d* → *dz*: *samochód* (*Nom.Sg.*), *samochodzie* (*Loc.Sg.*), *samochodem* (*Instr.Sg.*)

3) *a* → *e* and *zd* → *źdz*: *dojazd* (*Nom.Sg.*), *dojeździe* (*Loc.Sg.*), *dojazdem* (*Instr.Sg.*)

4) *ó* → *o* and *ł* → *l*: *stół* (*Nom.Sg.*), *stole* (*Loc.Sg.*), *stołem* (*Instr.Sg.*)

5) *e* → ϕ and *r* → *rz*: *puder* (*Nom.Sg.*), *pudrze* (*Loc.Sg.*), *pudrem* (*Instr.Sg.*)

6) *ie* → ϕ and *r* → *rz*: *cukier* (*Nom.Sg.*), *cukrze* (*Loc.Sg.*), *cukrem* (*Instr.Sg.*)

Finally, the following examples were given to show velar peculiarities:

7) *e* → ϕ with *k*: *budynek* (*Nom.Sg.*), *budynku* (*Loc.Sg.*), *budynkiem* (*Instr.Sg.*), *budynki* (*Nom.Pl.*)

8) *ó* → *o* with *g*: *róg* (*Nom.Sg.*), *rogu* (*Loc.Sg.*), *rogiem* (*Instr.Sg.*), *rogi* (*Nom.Pl.*)

From now on we will summarize the results of the runs verbally instead of using detailed tables, as we believe such detail will not serve any additional purpose.

At the end of first run for this paradigm only one of the the 8 groups above was covered completely. All vowel alternations for all groups came out right. However, the *Nom.Pl.* and *Acc.Pl.* endings were incorrectly generalized as -*i* instead of -*y*, probably because two "exceptional" velar examples (in -*i*) were provided in contrast to one "regular " non-velar example (in -*y*). Adding the *Nom.Pl.* forms of 3 non-velar words fixed this error. Velars were perfect except for the loss of *z* for 10 of 12 forms of *obowiązek*. Adding the *Nom.Pl.* form *obowiązki* fixed this. For *stół* and *dół*, the consonant mutation was incorrectly extended to *Gen.Sg.* Adding the *Gen.Sg.* form of *stół* fixed this error for both words. At the end of the second run all groups were correctly learned.

Supplementary testing after the above-mentioned words were correct included the words *nawóz, dochód, pozór, rozbiór, gród, rozchód, naród, wtorek, kieruneki*; all forms were correct.

6.4 Paradigm 4

Paradigm 4 contains *alternating man nouns.* The following primary example for the citation form *pasierb* was given in full:

	Number	
Case	Singular	Plural
Nom.	pasierb	pasierb**owie**
		pasierb**i**
Acc.	pasierb**a**	pasierb**ów**
Gen.	pasierb**a**	pasierb**ów**
Dat.	pasierb**owi**	pasierb**om**
Loc.	pasierb**ie**	pasierb**ach**
Instr.	pasierb**em**	pasierb**ami**

In this paradigm all of the consonant alternations encountered above are still in effect and some word-final consonants undergo additional mutations in the *Nom.Pl.* The velar peculiarities remain in effect. One additional complication in this paradigm is that there may be multiple *Nom.Pl.* forms for a given citation form (e.g., *pasierb***owie** and *pasierb***i** are both acceptable *Nom.Pl.* forms for *pasierb*). Furthermore *-i/-y* are allomorphs in complementary distribution (i.e., the second *Nom.Pl.* form in this paradigm can be realized with *-y* for certain word-final consonants).

Stem-final Consonant	**Nom.Pl. Ending**
b, f, w, m, n, z, t	*-owie* or *-i* or both
p, ch	*-i* only
d, ł	*-owie* only
r, k, g	*-owie* or *-y* or both

Since the analyzer needs only to analyze (and not generate) forms, there is no need to split up this paradigm into 5 different ones to account for each *Nom.Pl.* possibility: *-owie, -owie/-i, -i, -owie/-y, -y.* We simply permit overgeneration, allowing each word to have two *Nom.Pl.* forms: the correct one of the *-i/-y* allomorphs and *-owie.* Further, since the analyzer has no way to predict which of the *-i/-y* allomorphs is used with a given word-final consonant, explicit examples of each word-final consonant must be provided.

These considerations lead to splitting the citation forms for this paradigm into 14 groups, which represent the primary example plus 13 inflectional groups added as supplementary examples. The *Nom.Sg.*, *Loc.Sg.* and both (or applicable) *Nom.Pl.* forms were provided for all groups apart from the primary example. After the first run, 13 out of 14 groups were correctly dealt with. The remaining group was handled correctly in two additional runs runs: 2 more inflectional forms of the example in word-final *r* had to be provided to counter overgeneralization of the *r* → *rz* mutation.

Supplementary testing after the above-mentioned words were correct included the citation forms *drab, piastun, kasztelan, faraon, wójt, mnich, biedak, norweg, włoch.* The following errors were encountered:

- *norweg* got the *Acc.Sg./Gen.Sg.* form **norweda* instead of *norwega.* Adding the correct *Acc.Sg.* form fixed this problem.
- *włoch* got the *Nom.Pl.* form **włoci* instead of *włosi.* This form was added overtly.
- *mnich* got the *Nom.Pl.* form **mnici* instead of *mnisi.* This form was added overtly.

After these final additions, *włoch* and *mnich* ended up with the *Acc.Sg./Gen.Sg.* forms **włosa* and **mnisa* instead of *włocha* and *mnicha* (i.e., the mutation was overgeneralized again). Adding the correct *Acc.Sg.* form *włocha* overtly solved this problem for both words and all forms were now correct.

6.5 Paradigm 5

Paradigm 5 covered *non-alternating inanimate masculine nouns with genitive singular in -u and no vowel shifts.* The full declension for *garaż* was provided as the primary example.

	Number	
Case	Singular	Plural
Nom.	garaż	garaż**e**
Acc.	garaż	garaż**e**
Gen.	garaż**u**	garaż**y**
Dat.	garaż**owi**	garaż**om**
Loc.	garaż**u**	garaż**ach**
Instr.	garaż**em**	garaż**ami**

In this paradigm, the genitive plural endings are *-ów* and *-i/-y* (the latter are allomorphs in complementary distribution). Although many words permit either of two *Gen.Pl.* endings (*-y/-ów* or *-i/-ów*), this test employs only the most common ending for each word-final consonant.

In addition to the primary example group, five additional groups were exemplified (one for each consonant above except *ż*). The *Nom.Sg.* and *Gen.Pl.* forms were provided for the words representing each group. The lexicon contained an additional test example for four of the six groups (no more Polish words representing the other two groups could be found)

After the first run, 2 of the 6 groups were handled fine. and For the other groups, all forms were incorrect except for the *Nom.Sg.*, *Acc.Sg.* and *Gen.Pl.* Adding the *Instr.Pl.* of the words representing each of these 4 groups sufficed to fix all incorrect forms in the second run. Supplementary testing for additional words in this paradigm (such as *metal* and *płac*) produced correct results.

6.6 Paradigm 6

Paradigm 6 was for *non-alternating inanimate masculine nouns with genitive singular in -a and no vowel shifts.* The following declension for *bicz* was provided as the primary example:

	Number	
Case	Singular	Plural
Nom.	bicz	bicz**e**
Acc.	bicz	bicz**e**
Gen.	bicz**a**	bicz**y**
Dat.	bicz**owi**	bicz**om**
Loc.	bicz**u**	bicz**ach**
Instr.	bicz**em**	bicz**ami**

A spelling rule of Polish comes into play in this paradigm: letters that take a diacritic word-finally or when followed by a consonant are spelled with no diacritic plus an *-i* when followed by a vowel. For instance: *ń+u → niu*, *ń+owi → niowi*, *ć+u → ciu*, *ć+owi → ciowi*. Some, but not all, word-final letters in this paradigm have diacritics.

This paradigm is like paradigm 5 above except that the *Gen.Sg.* form unpredictably ends in *-a*, and more word-final consonants are accounted for (namely *cz, sz, rz, ść, ń*). Further, *Gen.Sg.* endings depend on the final consonant: they can be *-ów* (for *j, ch, szcz*), *-i* (for *ł, ść, ń*) or *-y* (for *cz, sz, rz, ż*). In many instances, more than one form is possible, but this test covers only the most common form for each stem-final consonant.

The citation forms in this paradigm broke down into 10 groups based on the final consonant. The *Nom.Sg.*, *Gen.Pl.* and *Instr.Pl.* forms were provided for the 9 groups (the 10^{th} is the primary example, for which all forms were provided). 8 of the 10 groups were handled correctly after the first run. Supplementary testing included the citation forms *klawisz, bąbel, strumień, łach, cyrkularz*; all inflectional forms were produced correctly.

7 Performance Issues

Generating a morphological analyzer once the descriptive data is given can be carried out very fast. Each paradigm can be processed within tens of seconds on a fast workstation, including the few tens of iterations of rule learning from the examples. A new version of the analyzer can be generated within minutes and tested rapidly on any test data. Thus, none of the processes described in this paper constitutes a bottleneck in the elicitation process. Figure 5 provides some relevant information from the runs of the first paradigm in Polish described above. The top graph shows, for different runs, the number of distinct rules generated from the aligned segmented and surface form pairs generated from the examples provided, using a rule format with at most 5 symbols in each of the left and right contexts. The bottom graph shows, for different runs, the total number of rules generated and generalized again with the same contexts size above.

There are a few interesting things about these graphs. As expected, when some more additional examples are added, the number of rules and the number of iterations to converge usually increases. All curves have a steeper initial segment and a steeper final segment. The initial steep segments result from the initial selection of rules that fix the largest number of "errors" between the segmented and the surface forms. Once those rules are found, the curves flatten as a number of morphographemic rules are selected, each dealing with a very small number errors. Finally, when all the morphographemic changes are accounted for, the segmentation rules kick in and each such rule fixes a large number of segmentation "errors", so that a few general rules deal with all such cases.

8 Summary and Conclusions

We have presented the highlights of our approach for automatically generating finite state morphological analyzers from information elicited from human informants. Our approach uses transformation-based learning to induce morphographemic rules from examples and combines these rules with the lexicon information elicited to compile the morphological analyzer. There are other opportunities for using machine learning in this process. For instance, one of the important issues in wholesale acquisition of open class items is that of determining which paradigm a given lemma or root word belongs to. From the examples given during the acquisition phase it is possible to induce a classifier that can perform this selection to aid the language informant.

We believe that we have presented a viable approach to the automatic generation of a natural language processor. Since this approach involves a human informant working in an elicit-generate-test loop, the noise and opaqueness of other induction schemes can be avoided. Our current work involves using similar principles to induce (light) syntactic parsers in the Boas framework.

We also feel that the task of analyzing a set of incorrectly generated forms and automatically offering a diagnosis of what may have gone wrong and what additional examples can be supplied as remedies–is, in itself, an important aspect of this work. Although we have only scratched the surface of this topic here, we consider it a fruitful extension of the work described in this paper.

9 Acknowledgements

This research was supported in part by Contract MDA904-97-C-3976 from the US Department of Defense. We also thank XRCE for providing the finite state tools. Most of this work was done while the first author was visiting NMSU Computing Research Laboratory during the 1998-1999 academic year.

References

[Antworth, 1990] Evan L. Antworth. *PC-KIMMO: A two-level processor for Morphological Analysis.* Summer Institute of Linguistics, Dallas, Texas, 1990.

[Brill, 1995] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–566, December 1995.

[Damerau, 1964] F. J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the Association for Computing Machinery*, 7(3):171–176, 1964.

[Golding and Thompson, 1985] Andrew Golding and Henry S. Thompson. A morphology component for language programs. *Linguistics*, 23, 1985.

[Goldsmith, 1998] John Goldsmith. Unsupervised learning of the morphology of a natural language. Unpublished Manuscript, available at `http://humanities.uchicago.edu/ faculty/ goldsmith/index.html`, 1998.

[Johnson, 1984] Mark Johnson. A discovery procedure for certain phonological rules. In *Proceedings of 10th International Conference on Computational Linguistics–COLING'84*, 1984.

[Kaplan and Kay, 1994] Ronald M. Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378, September 1994.

[Karttunen and Beesley, 1992] Lauri Karttunen and Kenneth. R. Beesley. Two-level rule compiler. Technical Report, XEROX Palo Alto Research Center, 1992.

[Karttunen *et al.*, 1992] Lauri Karttunen, Ronald M. Kaplan, and Annie Zaenen. Two-level morphology with composition. In *Proceedings of the* 15^{th} *International Conference on Computational Linguistics*, volume 1, pages 141–148, Nantes, France, 1992. International Committee on Computational Linguistics.

[Karttunen *et al.*, 1996] Lauri Karttunen, Jean-Pierre Chanod, Gregory Grefenstette, and Anne Schiller. Regular expressions for language engineering. *Natural Language Engineering*, 2(4):305–328, 1996.

[Karttunen, 1993] Lauri Karttunen. Finite-state lexicon compiler. XEROX, Palo Alto Research Center– Technical Report, April 1993.

[Karttunen, 1994] Lauri Karttunen. Constructing lexical transducers. In *Proceedings of the* 16^{th} *International Conference on Computational Linguistics*, volume 1, pages 406–411, Kyoto, Japan, 1994. International Committee on Computational Linguistics.

[Koskenniemi, 1983] Kimmo Koskenniemi. Two-level morphology: A general computational model for word form recognition and production. Publication No: 11, Department of General Linguistics, University of Helsinki, 1983.

[Nirenburg and Raskin, 1998] Sergei Nirenburg and Victor Raskin. Universal grammar and lexis for quick ramp-up of MT systems. In *Proceedings of First International Conference on Language Resources and Evaluation*, 1998.

[Nirenburg, 1996] Sergei Nirenburg. Supply-side and demand-side lexical semantics. In *Proceedings of the Workshop on Breadth and Depth of Semantic Lexicons at the 34th Annual Meeting of the Association for Computational Linguistics*, 1996.

[Nirenburg, 1998] Sergei Nirenburg. Project Boas: "A Linguist in a Box" as a multi-purpose language resource. In *Proceedings of COLING'98*, 1998.

[Oflazer, 1996] Kemal Oflazer. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–90, March 1996.

[Ranta, 1998] Aarne Ranta. A multilingual natural language interface to regular expressions. In Lauri Karttunen and Kemal Oflazer, editors, *Proceedings of International Workshop on Finite State Methods in Natural Language Processing, FSMNLP'98*, 1998.

[Rissanen, 1989] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific Publishing, 1989.

[Satta and Henderson, 1997] Giorgio Satta and John C. Henderson. String transformation learning. In *Proceedings of ACL/EACL'97*, 1997.

[Sproat, 1992] Richard Sproat. *Morphology and Computation.* MIT Press, 1992.

[Theron and Cloete, 1997] Pieter Theron and Ian Cloete. Automatic acquisition of two-level morphological rules. In *Proceedings of 5th Conference on Applied Natural Language Processing*, 1997.

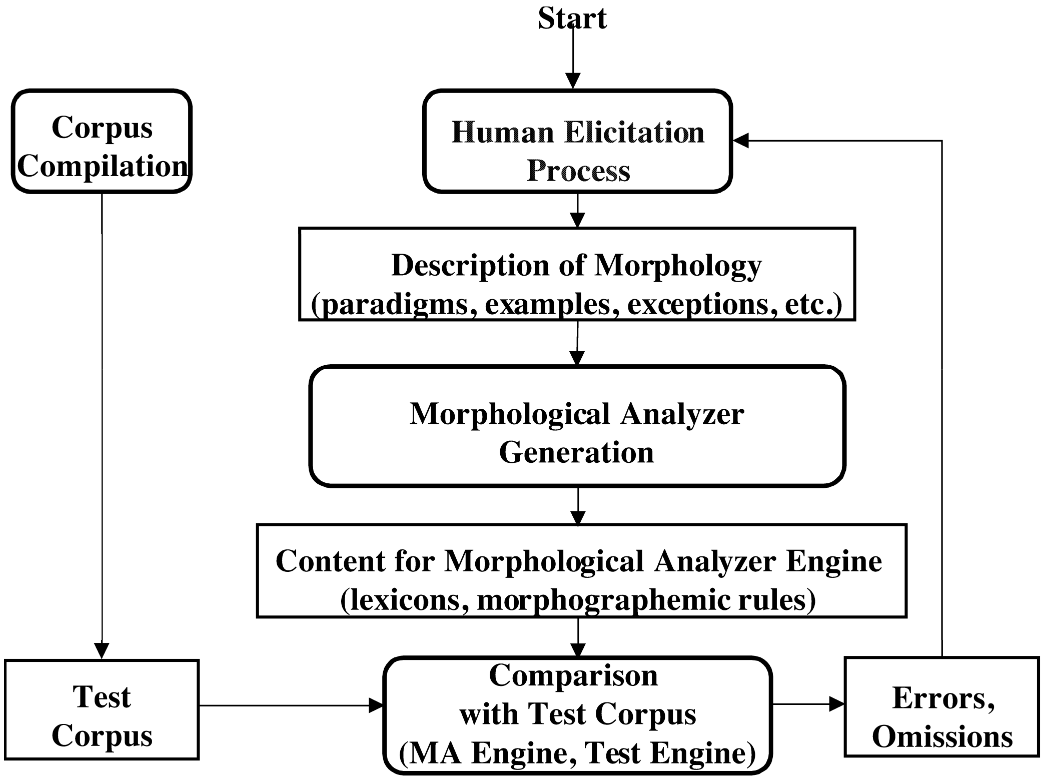
Start
Corpus Compilation
Human Elicitation Process
Description of Morphology (paradigms, examples, exceptions, etc.)
Morphological Analyzer Generation
Content for Morphological Analyzer Engine (lexicons, morphographemic rules)
Test Corpus
Comparison with Test Corpus (MA Engine, Test Engine)
Errors, Omissions

Figure 1: The Elicit-Build-Test Paradigm for Bootstrapping a Morphological Analyzer

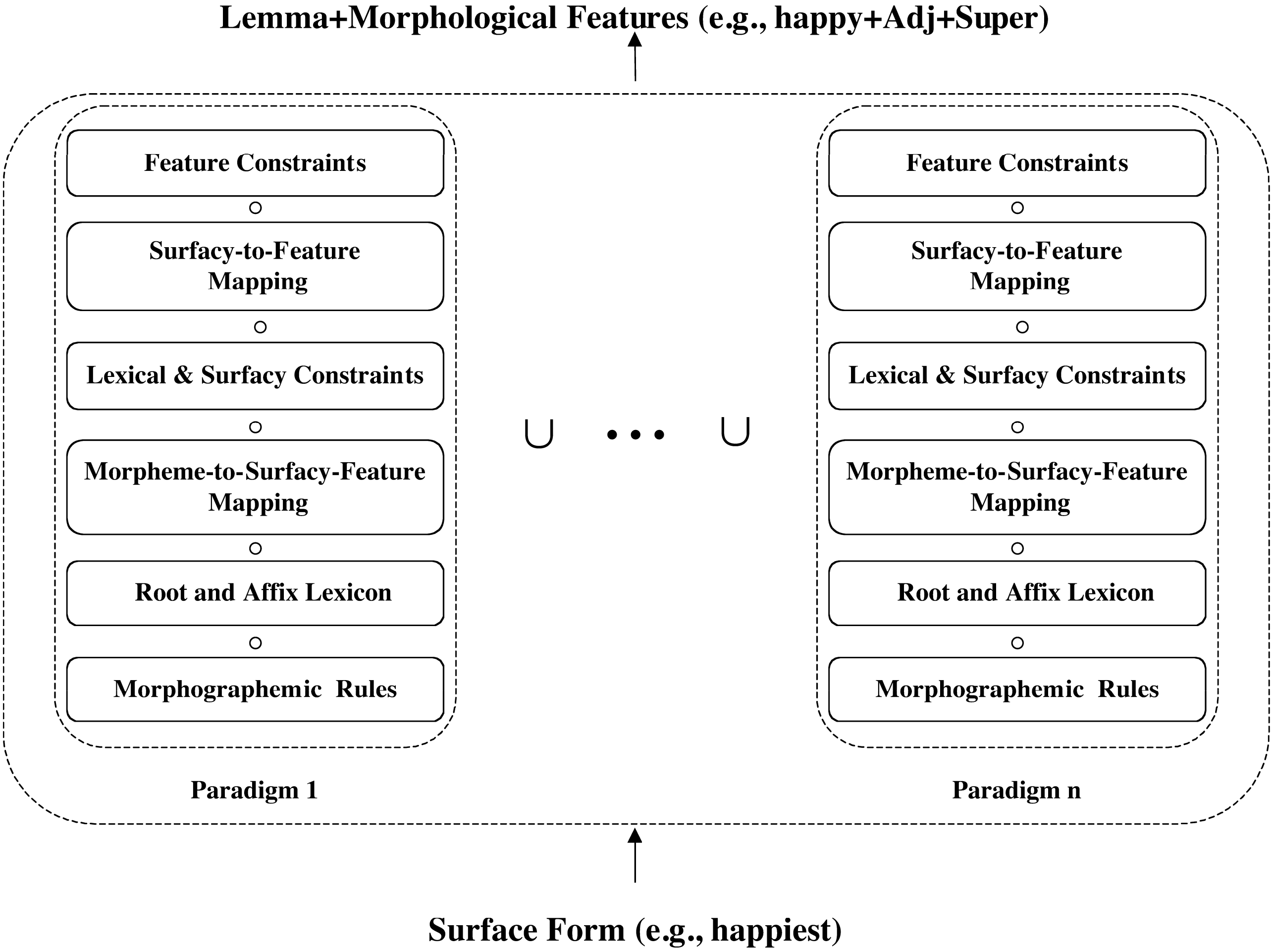
Lemma+Morphological Features (e.g., happy+Adj+Super)
Feature Constraints
∘
Surfacy-to-Feature Mapping
∘
Lexical & Surfacy Constraints
∘
Morpheme-to-Surfacy-Feature Mapping
∘
Root and Affix Lexicon
∘
Morphographemic Rules
Paradigm 1
∪ • • • ∪
Feature Constraints
∘
Surfacy-to-Feature Mapping
∘
Lexical & Surfacy Constraints
∘
Morpheme-to-Surfacy-Feature Mapping
∘
Root and Affix Lexicon
∘
Morphographemic Rules
Paradigm n
Surface Form (e.g., happiest)

Figure 2: General Architecture of the Morphological Analyzer

```
<LANGUAGE-DESCRIPTION TYPE = "morphology"
               NAME = "Polish"
               ALPHABET = "aąbcćdeęfghijklłmnńoópqrsśtuvwxyzźż"
               VOWELS = "aąeęioóuy"
               CONSONANTS= "bcćdfghjklłmnńpqrsśtvwxzźż"
               OTHER = "">
<PARADIGM NAME="MascInUStart" POS = "Noun" FEATURES="Masculine">
<PRIMARY-EXAMPLE>
<INF-GROUP>
               <PRIMARY-CIT-FORM FORM = "telefon">
               <INF-FORM FORM = "telefon" FEATURE = "Nom.Sg.">
               <INF-FORM FORM = "telefon" FEATURE = "Acc.Sg.">
               ...
               <INF-FORM FORM = "telefonach" FEATURE = "Loc.Pl.">
               <INF-FORM FORM = "telefonami" FEATURE = "Instr.Pl.">
</INF-GROUP>
</PRIMARY-EXAMPLE >
<EXAMPLE>
<INF-GROUP>
               <CIT-FORM FORM = "akcent">
               <INF-FORM FORM = "akcent" FEATURE = "Nom.Sg.">
               <INF-FORM FORM = "akcencie" FEATURE = "Loc.Sg.">
</INF-GROUP>
</EXAMPLE>
...
<LEXICON>
               <CIT-FORM FORM = "stron">
               <CIT-FORM FORM = "klub">
               <CIT-FORM FORM = "sklep">
               ...
</LEXICON>
</PARADIGM>
...
</LANGUAGE-DESCRIPTION>
```

Figure 3: Sample Paradigm Description Generated by Boas Elicitation

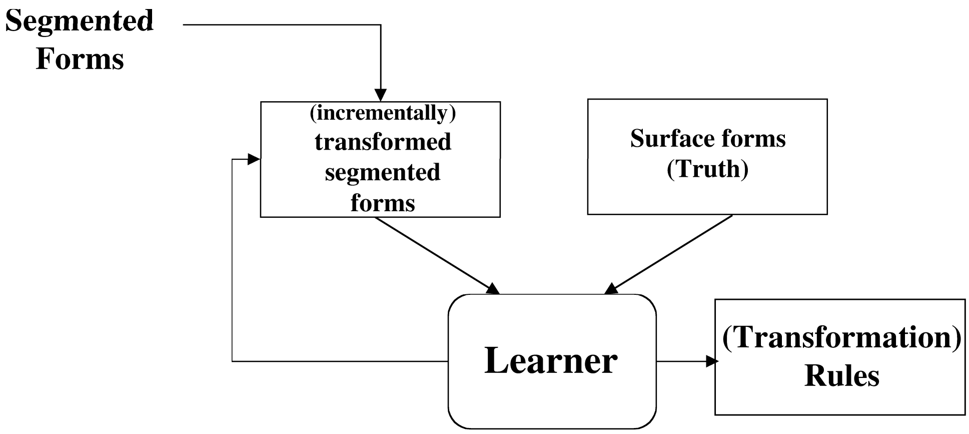

Figure 4: Transformation-based Learning of Morphographemic Rules

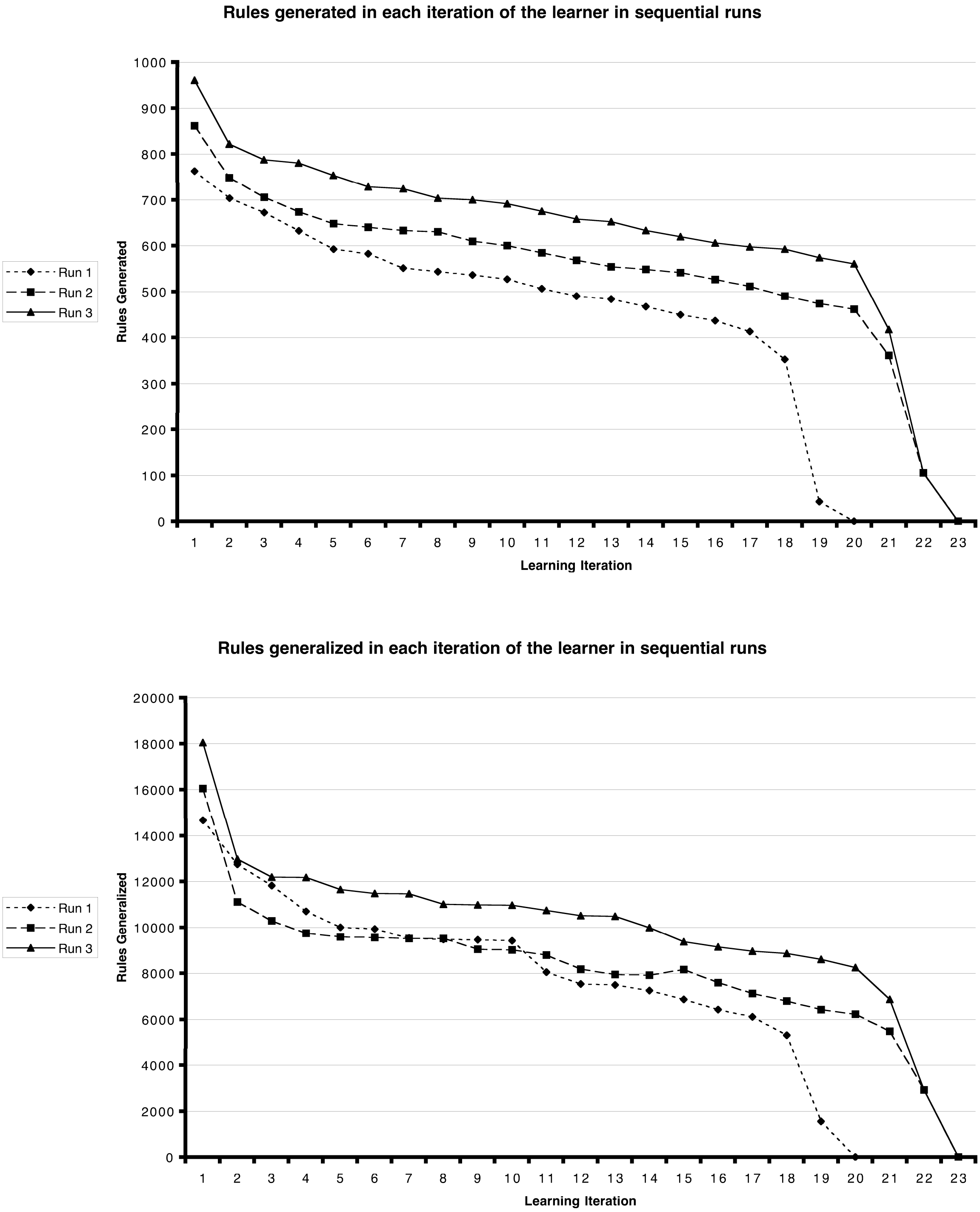

Figure 5: Rule Statistics for Processing Paradigm 1