Pages 49 to 60 of W. Daelemans, A. van den Bosch, and A. Weijters (Editors), Workshop Notes of the ECML/MLnet Workshop on Empirical Learning of Natural Language Processing Tasks, April 26, 1997, Prague, Czech Republic

Learning Slovene Declensions with FOIDL

Sašo Džeroski, Tomaž Erjavec

Saso.Dzeroski@ijs.si, Tomaz.Erjavec@ijs.si
Department for Intelligent Systems, Jožef Stefan Institute
Jamova 39, 1111 Ljubljana, Slovenia

Abstract

The paper presents results of using FOIDL to learn certain inflectional forms of Slovene nouns. FOIDL learns first-order decision lists, defined as ordered list of clauses; it has been previously tested on the problem of inducing rules for forming the past tense of English verbs. Slovene, unlike English, has rich inflectional morphology, and the paper reports the result of applying FOIDL over a large lexicon of Slovene word-forms to induce rules for the generation of the singular genitive forms of Slovene nouns of the masculine, feminine and neuter gender.

1 Introduction

The Slovene language belongs to the South-Slavic family of languages; with the other Slavic languages it shares a rich system of inflections. Nouns in Slovene exhibit lexically defined syntactic gender and inflect for number and case. The gender distinguished are masculine, feminine and neuter, numbers are singular, plural and dual, and the cases nominative, genitive, dative, accusative, locative and instrumental. This gives us 18 morphologically distinct forms comprising the paradigm of a noun. The situation is complicated, much as in Latin, by nouns belonging to various paradigm classes, of which we have three for the masculine gender, three for feminine and two for neuter. Finally, each paradigm class exhibits alternations, some of which are determined by the morphosyntactic properties of the noun (e.g., animacy), some by the morphophonological makeup of the noun, and some being idiosyncratic to the noun in question.

In the scope of the MULTEXT-East project [1] a lexicon for the Slovene language was developed which comprises all the inflectional word-forms of approximately 15.000 lemmas appearing in the corpus of the project, giving a total of more than 500.000

golob	=	Ncmsn
goloba	golob	Ncmda
goloba	golob	Ncmdn
goloba	golob	Ncmsa
goloba	golob	Ncmsg
golobe	golob	Ncmpa
golobi	golob	Ncmpi
golobi	golob	Ncmpn
golobih	golob	Ncmdl
golobih	golob	Ncmpl
golobom	golob	Ncmpd
golobom	golob	Ncmsi
goloboma	golob	Ncmdd
goloboma	golob	Ncmdi
golobov	golob	Ncmdg
golobov	golob	Ncmpg
golobu	golob	Ncmsd
golobu	golob	Ncmsl

Table 1: Paradigm of 'pidgeon'

word-forms. Each lexicon entry is composed of three fields, as can be seen in Table 1, which gives the complete paradigm of the lemma `golob` ('pigeon').

The first field is the word-form, the second the lemma (base form), and the third the morphosyntactic description of the word-form, in the first case above expanding to Noun, common, masculine, singular, nominative.

This paper is concerned with the problem of learning morphological rules for forming particular inflectional forms of nouns given the lemma. The particular form of the singular genitive was chosen as it usually exhibits the greatest number of alternations, and is given in dictionaries as the second leading from, i.e., the nominative singular together with the singular genitive is sufficient to determine the correct full paradigm for almost all nouns. Section 2 describes in more detail the data used in our experiments.

FOIDL [4] is used to learn rules for generating the genitive forms of nouns. FOIDL is an inductive logic programming (ILP) [3] system that learns first-order decision lists, i.e., ordered lists of clauses. A brief description of FOIDL and its application to learning past tense is given in Section 3.

Section 4 describes the experiments with FOIDL. The induced rules are discussed, as well as their performance on unseen cases. Section 5 concludes with a discussion and some directions for further work.

2 The data

In our setting, examples are pairs of lemmas and word-forms that have the same morphosyntactic description. A particular morphosyntactic description can thus be considered a concept. E.g., the fact `ncmsg([g,o,l,o,b],[g,o,l,o,b,a])` is a positive example for the concept `Ncmsg` (cf. Table 1), represented by the predicate `ncmsg`. Note that an orthographic representation is used.

For our experiments, three 'concepts' were selected from the lexicon of the Slovene language [1], namely the singular genitive forms of (proper and common) masculine, feminine and neuter nouns, i.e., `N*msg` (2893 cases), `N*fsg` (2755 cases) and `N*nsg` (1323 cases). These are represented by the predicates `nxmsg`, `nxfsg` and `nxnsg`.

A simple way to arrive from the base form to an oblique form is to subtract a suffix from the base form and then append a suffix to the resulting 'stem', thus getting rules of the form *-suff$_{base}$/+suff$_{oblique}$*, e.g., `golob -/+a = goloba`. To cover the singular genitive data set we need 24 such rules for the most complicated case of the masculine, 10 for the feminine, and 11 for neuter. To illustrate the morphological processes involved we give for the `N*fsg` case in Table 2 the complete set of rules, together with their coverage.

a)	2103	-a/+e
b)	522	-/+i
c)	113	-ev/+ve
d)	9	-en/+ni
e)	2	-i/+ere
f)	2	-el/+li
g)	1	-i/+vi
h)	1	-er/+ri
i)	1	-em/+mi
j)	1	-an/+ni

Table 2: Rules for feminine singular genitive form

The two rules under a) and b) belong, respectively, to the nouns of the canonical first and second feminine declensions, e.g., *miz-a/miz-e; perut-0/perut-i* ('table'; 'wing'). Case c) belongs to a relatively common alternation affecting all feminine nouns of the first declension ending in *-ev*, e.g., *klet<u>e</u>v-)/kletv-e* ('curse'). Cases d), f), h), and i) all exhibit a common phonological alternation in Slovene, whereby a schwa in the last syllable is deleted in word-forms with a non-null ending, e.g., *ljubez<u>e</u>n/ljubezn-i* ('love'). The e) case is an idiosyncratic alternation affecting only two nouns of Slovene, namely *hč-i/hč-ere; mat-i/mat-ere* ('daughter'; 'mother'). Finally, g) and j) are, again, forms of idiosyncratic nouns: *kr<u>i</u>-0/kr<u>v</u>-i; rav<u>a</u>n-0/ravn-i* ('blood'; 'plain').

As regards Slovene morphology, it should be noted that the morphological stem, especially in its orthographic form, does not in all cases contain enough information to correctly predict the correct forms that it can give rise to. In a stem lexicon we would typically need to accompany the stem with morphological information i.e., declension, morphosyntactic information, e.g., animacy, and phonological information (e.g., that the -*e*- in the final syllable is indeed a schwa and not a stressed *e*). However, in a lexicon such as the one for the MULTEXT-East project, the problem is alleviated, to a certain extent, as it contains not stems, but the base form of the words. For example, the stem of 'table' is *miz*-, while the base form is *miza*, and, as can be seen above, the nominative singular ending -*a* serves to uniquely identify the word as belonging to the first feminine declension and thus to correctly determine the genitive ending. Still, the $-suff_{base}/+suff_{oblique}$ type rules will in general not have enough discriminatory power to apply only to the correct base forms. The simplest extension to such rules is to take into account more orthographic material from the base, and this is where FOIDL induction comes into play.

3 FOIDL and learning past tense

FOIDL [4] is an inductive logic programming (ILP) [3] system. Unlike most other ILP approaches that learn unordered sets of Horn clauses, FOIDL learns first-order decision lists, i.e., ordered lists of clauses. First-order decision lists seem to be a very appropriate formalism for representing linguistic knowledge, as they allow for an elegant way of representing exceptions to general rules. The exceptions are placed before the general rules in a decision list: the first applicable rule from the list is used when treating new cases. The data structure produced by FOIDL thus implements a version of the Elsewhere Condition, well known in morphological theory [2].

Another important feature of FOIDL is its ability to learn decision lists from *positive examples only*. Most other ILP systems rely on negative examples to avoid overly general hypotheses. FOIDL on the other hand, uses an output completeness assumption. A mode declaration for the target predicates is provided by the user, specifying for each argument whether it is an input or an output argument. The output completeness assumption states that the training set contains all of the corresponding output patterns for every unique input pattern that it contains. The ability to learn from positive data only is also very important for almost all linguistic applications of ILP.

Like all ILP systems, FOIDL can use background knowledge defining predicates relevant for learning the target predicate. While some ILP systems, e.g., FOIL [5], can only use extensional background knowledge, consisting of ground facts only, FOIDL can use intensional background knowledge consisting of PROLOG clauses. Predicates for list processing, relevant for learning linguistic concepts, can thus be represented concisely and efficiently.

FOIDL has been successfully applied to the problem of learning rules for forming the past tense of English verbs [4]. Three different variants of the problem have been addressed: phonetic representation, phonetic representation of regular verbs only, and orthographic representation. For illustration, `past([s,l,e,e,p],[s,l,e,p,t])` is a positive example for the last case.

In all three cases, the list processing predicate `split(A,B,C)`, which splits a list `A` into two nonempty lists `B` and `C`, was used as background knowledge. This predicate is defined as:

```
split([X,Y|Z],[X],[Y|Z]).
split([X|Y],[X|Z],W) :- split(Y,Z,W).
```

The first argument (present tense of the verb) of the target predicate `past` is an input argument and the second (past tense) is an output argument. FOIDL was given the opportunity to use constant prefixes and suffixes in its rules. An excerpt from a first-order decision list learned by FOIDL from 250 examples is given below.

```
past(A,B) :- split(A,C,[e,p]), split(B,C,[p,t]),!.
...
past(A,B) :- split(B,A,[d]), split(A,C,[e]), !.
past(A,B) :- split(B,A,[e,d]).
```

Note the `!` (cuts) in the clauses: these indicate that only the first applicable clause from the list should be used when treating new examples. The FOIDL output, however, does not list the cuts. The FOIDL output in the next Section is shown in its original form without cuts. When treating new cases, it is interpreted as a decision list, i.e., as if the cuts were there.

Given 1392 examples, FOIDL used up to 500 examples for learning and the remainder for testing. Given 500 examples for learning, FOIDL achieves accuracy of approximately 85% on the testing set. The running time on a SUN SPARC 10 was on the order of 8 hours.

4 Experiments and results

Our decision to use FOIDL for learning Slovene declensions was based on its properties listed in the previous Section and its success on the English past tense learning problem. The training sets sizes were chosen with regard to FOIDLś computational efficiency limits, i.e., not to exceed 500 examples. FOIDL was run three times, once for each target concepts, which in turn correspond to the three genders of Slovene nouns in their singular genitive form (`nxfsg`, `nxmsg`, and `nxnsg`).

The sizes of the entire datasets were 2755, 2893 and 1323 facts, for the female, male, and neuter gender, respectively. respectively. For training, one sixth of the `nxfsg` and `nxmsg` sets and one third of the `nxnsg` set were randomly chosen. The remainder of the data were used to test the performance of the rules induced by FOIDL. The rules reported on in this section were generated in this fashion.

We also performed a series of five FOIDL runs for each gender, where the training sets were half the size of the above, comprising between 220 and 250 examples. In each of the five runs, the training examples were chosen randomly (a different choice for each of the five runs). The average and standard deviation of the accuracies (over the five runs) on the testing examples were measured.

The general set-up for the experiments was exactly as for the orthographic past tense learning experiment of Mooney and Califf, i.e., the training data was encoded as PROLOG facts of the form:[1]

```
nxfsg([a,b,e,c,e,d,a],[a,b,e,c,e,d,e]).
nxfsg([a,g,o,n,i,j,a],[a,g,o,n,i,j,e]).
nxfsg([b,o,l,e,cx,i,n,a],[b,o,l,e,cx,i,n,e]).
...
```

The first argument of each target predicates (the lemma) is an input argument and the second is an output argument. The predicate `split` is used as background knowledge. Constant prefixes and suffixes are allowed in the rules.

The programs generated for the three concepts show varying degrees of success in capturing the relevant morphological generalizations. First, it should be noted that due to the random sampling of the dataset some low-frequency alternations were not included in the training set. The rules to generate such forms were obviously not discovered by FOIDL. Second, as has been already mentioned, FOIDL works only with the orthographic representation of the words, which does not contain enough information to predict the correct rule for generating the singular genitive form in all cases.

In brief, the rules induced from the larger training sets achieve accuracy on unseen (testing) examples of 99.1% for feminine, 95.9% for neuter, and 86.4% for the masculine gender. The average accuracies on unseen examples for the five runs with smaller training sets are 99.3 (0.1)%, 94.4 (3.3)%, and 85.0 (2.4)%, respectively (the numbers in brackets are the standard deviations). These accuracies are as would be expected from the number of 'rules' as discussed in Section 2, i.e., 10 for the feminine, 11 for neuter, and 24 for masculine. We examine first the simplest case of the feminine gender, and then the most complicated one of the masculine gender.

[1]The Slovene characters č, š, ž are encoded as `cx`, `sx`, and `zx` respectively.

```
nxfsg(A,B) :- split(A,C,[v]), split(A,D,[e,v]), split(B,D,[v,e]).
nxfsg(A,B) :- split(A,C,[a]), split(B,C,[e]).
nxfsg(A,B) :- split(B,A,[i]).
```

Table 3: FOIDL rules for feminine

4.1 Feminine Singular Genitive

The feminine case is the simplest of the three, and the rules that FOIDL generates to cover it are given in Table 3.

It can be seen that FOIDL correctly induced the three most frequent rules of Table 2. As the base form of a noun for these three cases gives an unambiguous cue for to the rule to be applied, and as these three cases cover the vast majority of feminine nouns, the accuracy of 99.1% comes as no surprise. the errors are all due to the rare alternations and idiosyncratic nouns not covered in the training set.

```
nxmsg([zx,i,g,a],[zx,i,g,e]).
nxmsg([d,o,s,t,o,j,e,v,s,k,i],[d,o,s,t,o,j,e,v,s,k,e,g,a]).
...
nxmsg([z,i,d],[z,i,d,u]).
nxmsg([n,o,s],[n,o,s,u]).
nxmsg([c,v,e,t],[c,v,e,t,u]).
...
nxmsg([b,l,a,g,o,r],[b,l,a,g,r,a]).
nxmsg([d,a,n],[d,n,e]).
...
nxmsg([t,e,d,e,n],[t,e,d,n,a]).
nxmsg([o,v,e,n],[o,v,n,a]).
nxmsg([p,e,c,e,l,j],[p,e,c,l,j,a]).
...
nxmsg([b,r,e,v,i,r],[b,r,e,v,i,r,j,a]).
nxmsg([t,u,m,o,r],[t,u,m,o,r,j,a]).
nxmsg([o,k,u,p,a,t,o,r],[o,k,u,p,a,t,o,r,j,a]).
...
```

Table 4: FOIDL rules for masculine: exceptions

However, it should be noted that the first rule redundantly tests whether the suffix of the base form is *-v*, as it then performs a specialization of this test, namely whether the suffix is *-ev*. This type of redundancy occurs in the other two concepts as well: it is due to the greedy search of FOIDL and can be removed by post-processing similar to the one used in later versions of FOIL [5].

```
nxmsg(A,B) :- split(A,C,[n,e,k]), split(B,C,[n,k,a]).
nxmsg(A,B) :- split(A,C,[e,k]),   split(B,C,[k,a]), split(A,D,[t,e,k]).
nxmsg(A,B) :- split(A,C,[e,k]),   split(B,C,[k,a]), split(A,D,[v,e,k]).
nxmsg(A,B) :- split(A,C,[e,k]),   split(B,C,[k,a]), split(A,D,[sx,e,k]).
nxmsg(A,B) :- split(A,C,[e,k]),   split(B,C,[k,a]), split(A,D,[d,e,k]).
nxmsg(A,B) :- split(A,C,[e,k]),   split(B,C,[k,a]), split(A,D,[zx,e,k]).
nxmsg(A,B) :- split(A,C,[e,k]),   split(B,C,[k,a]), split(A,D,[cx,e,k]).
nxmsg(A,B) :- split(A,C,[e,k]),   split(B,C,[k,a]), split(A,[z],D).
nxmsg(A,B) :- split(A,C,[e,r]),   split(B,C,[r,a]), split(A,D,[b,e,r]).
nxmsg(A,B) :- split(A,C,[e,r]),   split(B,C,[r,a]), split(A,D,[e,t,e,r]).
nxmsg(A,B) :- split(A,C,[e,c]),   split(B,C,[c,a]).
nxmsg(A,B) :- split(A,C,[e,n,j]), split(B,C,[n,j,a]).
nxmsg(A,B) :- split(A,C,[z,e,m]), split(B,C,[z,m,a]).

nxmsg(A,B) :- split(B,A,[j,a]),   split(A,C,[i]).
nxmsg(A,B) :- split(B,A,[j,a]),   split(A,C,[r]),
              split(A,D,[a,r]),   split(A,E,[v,a,r]).
nxmsg(A,B) :- split(B,A,[j,a]),   split(A,C,[r]),   split(A,D,[e,r]).
nxmsg(A,B) :- split(B,A,[j,a]),   split(A,C,[r]),
              split(A,D,[a,r]),   split(A,[s],E).
nxmsg(A,B) :- split(B,A,[j,a]),   split(A,C,[r]),
              split(A,D,[a,r]),   split(A,E,[k,a,r]).

nxmsg(A,B) :- split(A,C,[a]),     split(B,C,[e]),   split(A,[v,o],D).
nxmsg(A,B) :- split(A,C,[a]),     split(B,C,[a]).

nxmsg(A,B) :- split(B,A,[t,a]),   split(A,C,[e]).

nxmsg(A,B) :- split(A,C,[o]),     split(B,C,[a]).
nxmsg(A,B) :- split(B,A,[a]).
```

Table 5: FOIDL rules for masculine: generalisations

4.2 Masculine Singular Genitive

This is the most varied case, as 24 rules are needed to cover the complete data set. These cover four different paradigm classes, as well as numerous alternations and idiosyncratic nouns. A number of cases (30) were treated by FOIDL as exceptions,

and some examples are given in Table 4. The first two cases are, respectively, an example of the second and third masculine paradigms. Here FOIDL is correct in postulating them as exceptions, as insufficient information is available in the base form to correctly predict the genitive. The next three cases are examples of an ending alternation which affects some short nouns; instead of the usual *-a* ending, the *-u* ending appears in the genitive case. As there is no rule that would predict which nouns fall in this class, FOIDL is correct in treating them as exceptions. The same holds for the next group, which encompasses the rare and unpredictable case of *-o-* and *-e-* elision. The last two groups are, however, different. Both contain examples of a productive alternation of (*-e-* elision and stem lengthening with *-j-*), which are, furthermore also encompassed in the generalization rules that follow.

The generalization rules are given in Table 5. For illustrative purposes they have been reordered to make natural groups, taking care, however, that the procedural semantics of the program remains the same. The first group tries to model the *-e-* elision alternation, by listing all the possible end-strings of the base forms where it occurs. The second group models stem lengthening by *-j-*. The third group covers masculine nouns ending in *-a*, i.e., belonging to the second masculine paradigm.[2] The penultimate group covers a common alternation whereby nouns ending in *-e* in the base form lengthen their stem with *-t-* with non-null endings. Finally, the last group takes nouns ending in *-o* or *-0* and appends to them the canonical first declension ending *-a*.

4.3 Neuter Singular Genitive

The neuter case needs 11 rules to generate all the genitive forms from the data set. From the 441 training examples, FOIDL generates a decision list with 18 clauses and 5 exceptions. The accuracy of this decision list on the remaining 882 cases is 95.9%.

The clause `nxnsg(A,B) :- split(A,C,[o]), split(B,C,[a])` (fourth from the bottom) corresponds to the second most common rule `-o/+a` represented by 315 instances in the complete dataset. The bottom clause corresponds to the rule `-/+sa`, represented by 2 instances, the next clause corresponds to the rule `-/+na`, represented by 9 instances, and the second clause from the top to the rule `-/+ga`, represented by 2 instances. The remaining clauses all try to approximate the rule `-e/+a`, represented by 969 instances in the entire dataset.

[2]Such masculine nouns actually fall outside the scope of the FOIDL induction algorithm, which presupposes that the mapping is a function: Slovene nouns ending in *-a* can be declined either according to the first or according to the second masculine paradigm. In other words, they exhibit free variation in the oblique forms.

```
nxnsg(A,B) :- split(A,C,[l,j,e]), split(B,C,[l,j,a]), split(A,[o],D).
nxnsg(A,B) :- split(A,C,[k,o]), split(B,C,[k,e,g,a]).
nxnsg(A,B) :- split(A,C,[j,e]), split(B,C,[j,a]), split(A,D,[b,j,e]).
nxnsg(A,B) :- split(A,C,[j,e]), split(B,C,[j,a]), split(A,D,[d,j,e]).
nxnsg(A,B) :- split(A,C,[j,e]), split(B,C,[j,a]), split(A,[r,a,z],D).
nxnsg(A,B) :- split(A,C,[j,e]), split(B,C,[j,a]), split(A,D,[s,j,e]).
nxnsg(A,B) :- split(A,C,[e]),   split(B,C,[a]),   split(A,D,[c,e]).
nxnsg(A,B) :- split(A,C,[e]),   split(B,C,[a]),
              split(A,D,[j,e]), split(A,E,[cx,j,e]).
nxnsg(A,B) :- split(A,C,[e]),   split(B,C,[a]),
              split(A,D,[j,e]), split(A,[p,o],E).
nxnsg(A,B) :- split(A,C,[e]),   split(B,C,[a]),
              split(A,D,[j,e]), split(A,[v],E).
nxnsg(A,B) :- split(A,C,[e]),   split(B,C,[a]),
              split(A,D,[j,e]), split(A,E,[i,l,j,e]).
nxnsg(A,B) :- split(A,C,[e]),   split(B,C,[a]),
              split(A,D,[j,e]), split(A,E,[v,j,e]).
nxnsg(A,B) :- split(A,C,[e]),   split(B,C,[a]),
              split(A,D,[j,e]), split(A,E,[zx,j,e]).
nxnsg(A,B) :- split(A,C,[e]),   split(B,C,[a]),
              split(A,D,[j,e]), split(A,E,[t,j,e]).
nxnsg(A,B) :- split(A,C,[e]),   split(B,C,[a]),   split(A,D,[cx,e]).
nxnsg(A,B) :- split(A,C,[o]),   split(B,C,[a]).
nxnsg(A,B) :- split(A,C,[e]),   split(B,C,[a]),   split(A,D,[n,j,e]).
nxnsg(A,B) :- split(B,A,[n,a]), split(A,C,[m,e]).
nxnsg(A,B) :- split(B,A,[s,a]).
```

Table 6: FOIDL rules for neuter: generalisations

5 Conclusions

We have presented the results of an initial attempt to learn rules for generating inflectional forms of Slovene nouns. In particular, FOIDL was applied to learn rules for the genitive form of proper and common nouns in singular, for each of the three genders separately. Taking into account that FOIDL was given very limited background knowledge, the results obtained are quite satisfactory: the accuracy of the induced rules for the genitive singular is approximately 99% for the feminine, 95% for the neuter, and 85% for the masculine gender. The errors can be traced to two causes, both having to do with insufficient information available to FOIDL.

First, while Slovene orthography is much closer to the phonological form of a word than it is in English, it is nevertheless sometimes not enough to predict whether a certain phonologically determined alternation should take place or not. For example, the *-e-* elision should take place only where the *e* is a schwa and this cannot be pre-

dicted from the spelling alone — compare e.g., *angel/angela* and *triangel/triangla*. To cover such cases, a phonological representation has to be substituted for, or added to the orthographic one. Furthermore, the background knowledge of FOIDL could then be extended to take phonological regularities into account, by e.g., distinguishing vowels from consonants etc., thus leading to better generalizations.

Second, FOIDL makes use only of the form of the lemma. For Slovene, this is insufficient, as additional background knowledge on the content of the lemma is necessary in order to correctly generate word-forms. In particular, lexical morphosyntactic information must be incorporated into the lexicon and made use of by the induction algorithm. The most obvious example is to add the paradigm class to the lexical entries, but other necessary information includes animacy for masculine nouns and the origin of the noun, i.e., whether it is of native or foreign origin.

Given the preliminary nature of the work presented here, many other directions for further work can be pointed out. As regards the induction methodology, at least two improvements of FOIDL seem to be needed. Efficiency seems to be a major problem, limiting the size of training sets that can be considered to approximately 500 examples. Post-processing of the induced decision lists is also needed in order to remove irrelevant literals.

On a larger scale, the work presented here should be extended to complete paradigms, i.e., rules should be learned for forming all the 17 oblique forms of Slovene nouns (from the base form) and to cover other major categories of words. An additional and useful extension would be to learn rules for morphological analysis, rather than synthesis. These would generate e.g., the nominative form of a noun from its genitive form. Finally, it would be interesting to consider the discovery of morphological phenomena in a more modular fashion: instead of simply generating/analyzing a form with one type of clauses, the morphological phenomena of morphotactics and morphophonology could be considered separately.

Acknowledgements

This work was supported in part by the ESPRIT IV project 20237 ILP2.

References

[1] T. Erjavec, N. Ide, V. Petkevič, and J. Véronis. MULTEXT-East: Multilingual text tools and corpora for Central and Eastern European languages. In *Proceedings of the First TELRI European Seminar: Language Resources for Language Technology*, pages 87–98, 1996. 15–16 September 1995, Tihany, Hungary.

[2] Paul Kiparsky. "Elsewhere" in phonology. In Steven R. Anderson, editor, *Festschrift for Morris Halle*, pages 93–106. Holt, Rinehart and Winston, New York, 1973.

[3] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, Chichester, 1994.

[4] R.J. Mooney and M.-E. Califf. Induction of first-order decision lists: Results on learning the past tense of English verbs. *Journal of Artificial Intelligence Research*, (3):1–24, 1995.

[5] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3): 239–266, 1990.