Learning Multilingual Morphology with CLOG

Suresh Manandhar,* Sašo Džeroski,† Tomaž Erjavec†

* Intelligent Systems Group, Department of Computer Science, University of York
YO10 5DD, York, U.K.
Suresh@cs.york.ac.uk

† Department for Intelligent Systems, Jožef Stefan Institute
Jamova 39, SI-1000 Ljubljana, Slovenia
Saso.Dzeroski@ijs.si, Tomaz.Erjavec@ijs.si

Abstract. The paper presents the decision list learning system CLOG and the results of using it to learn nominal inflections of English, Romanian, Czech, Slovene, and Estonian. The dataset used to induce rules for the synthesis and analysis of the inflectional paradigms of nouns and adjectives of these languages is the MULTEXT-East multilingual tagged corpus. The ILP system FOIDL is also applied to the same dataset, and this paper compares the induction methodology and results of the two systems. The experiment shows that the accuracy of the two systems is comparable when using the same training set. However, while FOIDL is, due to efficiency reasons, severely limited in the size of the training set, CLOG does not suffer from such limitations. With the increase of the training set size possible with CLOG, it significantly outperforms FOIDL and learns highly accurate morphological rules.

1 Introduction

Machine learning methods been recently applied to a variety of tasks within the area of natural language processing [2]. Inductive logic programming (ILP) systems have been applied to tasks such as learning to parse [6] and learning part-of-speech tagging [1]. Learning of morphological structure has also been attempted with the ILP system FOIDL [7], with the original experiment focused on relatively small samples of English. In subsequent work, FOIDL was used to learn the synthesis and analysis rules for Slovene nouns [3]. Here, the system was used to learn morphological rules for producing the inflectional forms of nouns given the base form (the *lemma*) as well as for deducing the lemma from these inflectional forms. Thus, rules for both morphological synthesis and analysis were learned. While these rules induced by FOIDL had a relatively high accuracy on smaller datasets, we were severely hampered by the fact that we were unable to train on larger datasets. This bottleneck is due to the training inefficiency of FOIDL.

In this paper we present a new decision list learning system, CLOG, which is similar to FOIDL but with a much greater training efficiency, enabling it to train on larger datasets, and thus achieve significantly better results. Furthermore, we extend the scope of the experiments to encompas adjectives and nouns of English, Romanian, Czech, Slovene, and Estonian.

The aim of the paper is to empirically demonstrate that it is possible to extend current ILP techniques to realistic NL tasks such as morphological rule learning that require processing of large amounts of data by the use of more efficient algorithms.

The training and testing data was taken from the MULTEXT-East tagged corpus [4], and converted to Prolog encoding, as explained in Section 2. CLOG and FOIDL are used to learn rules for synthesizing and analyzing the noun and adjective forms of the five languages. An overview of CLOG and a comparison with FOIDL is given in Section 3. Section 4 describes the MULTEXT-East corpus experiments with CLOG and FOIDL. Discussed here are the experimental setup, the induced rules of both systems for the synthesis and the analysis tasks, and their performance on unseen text. Section 5 concludes with a discussion and some directions for further work.

2 The Data

The EU MULTEXT-East project [4] developed corpora, lexica and tools for six Central and East-European languages; the project reports and samples of results are available at *http://nl.ijs.si/ME/*. The centerpiece of the corpus is the novel "*1984*" by George Orwell, in the English original and translations. For the experiments reported here, the first three parts of the "*1984*" were taken for training, and the fourth part (Appendix: "*The Principles of Newspeak*") for testing.

The "*1984*" corpus is tokenised, and its words labelled with linguistic annotations both in context disambiguated and non-disambiguated forms. The linguistic annotation of a word contains its lemmas and morphosyntactic descriptions (MSDs). So, for example, the lexical annotation of the Slovene word *članki* contains two lemma/MSD pairs: *članek*/`Ncmpi` and *članek*/`Ncmpn`.

The MSDs are structured and more detailed than is commonly assumed for part-of-speech tags; they are compact string representations of a simplified kind of feature structures — the formalism and MSD grammar for the MULTEXT-East languages is defined in [5]. The first letter of a MSD encodes the part of speech (`Noun`, `Adjective`), while the letters following give the value of the position determined attribute. Each part of speech defines its appropriate attributes and their values, acting as a kind of feature-structure type or sort. So, for example, the MSD `Ncmpi` expands to PoS:Noun, Type:common, Gender:masculine, Number:plural, Case:instrumental. It should be noted that in case a certain attribute is not appropriate (1) for a language, (2) for the particular combination of features, or (3) for the the word in question, this is marked by a hyphen in the attribute's position. Estonian nouns, for example, are not marked for gender and a 'Noun common (no gender) singular translative' is written as `Nc-s4`.

For our experiments triplets were extracted from the tagged corpus, consisting of the word-form itself, and the lexical, undisambiguated lemmas with their accompanying MSDs, thus using a setting similar to the one prior to tagging. As mentioned, we considered only the triples with noun and adjective MSDs.

Each triplet gives rise to two examples, one for synthesis and one for analysis. The examples have the form `syn_msd(lemma,orth)` and `ana_msd(orth,lemma)`. Within the learning setting of inductive logic programming, `syn_msd` and `ana_msd` are relations or predicates, that consist of all pairs (lemma, word-form), resp. (word-form, lemma) that have the same morphosyntactic description [1]. A set of rules has to be learned for each of these predicates or concepts.

Encoding-wise, the MSD's part-of-speech is decapitalised and hyphens are converted to underscores. The word-forms and lemmas are encoded as lists of characters, with non-ASCII characters encoded as SGML entities. In this way, the generated examples comply with Prolog syntax. For illustration, the orthography/lemma/MSD triplet *članki*/*članek*/`Ncmpn` gives rise to the following two examples:

```
syn_n0mpn([ccaron,l,a,n,e,k],[ccaron,l,a,n,k,i]).
ana_n0mpn([ccaron,l,a,n,k,i],[ccaron,l,a,n,e,k]).
```

Certain attributes have (almost) no effect on the inflectional behaviour of the word. We generalise over their values in the predicates, and indicate this by a `0` for the value of the vague attribute, as seen above for the collapsing of proper and common nouns (`Nc`, `Np`) to `n0`.

Below we give the numbers of these generalised MSDs for the complete noun and adjective paradigms for each of the five languages:

```
en:   1 MSD  =  1 N +  0 A
ro:  30 MSDs = 15 N + 15 A
et:  58 MSDs = 29 N + 29 A
cs:  91 MSDs = 45 N + 46 A
sl: 108 MSDs = 54 N + 54 A
```

In English the plural of nouns to account for in both the Noun and the Adjective paradigms, and only this MSD was used for the dataset. For the other languages complete paradigms were modeled, including the base forms themselves. The languages have a varying numbers of MSDs, depending on their inflectional complexity and the particular choices made in the MSD grammar. Lowest is Romanian; next comes Estonian, which has a large number of cases but does not distinguish gender. The two Slavic languages, known for their heavy inflection (on gender, number, case, animacy and definiteness) are highest.

3 CLOG and Learning of Morphological Rules

CLOG is a system for learning of first-order decision lists. CLOG shares a fair amount of similarity with FOIDL [7]. Like FOIDL, CLOG can learn first-order decision lists from positive examples only — an important consideration in NLP applications. CLOG inherits the notion of *output completeness* from FOIDL to generate implicit negative examples (see [7]). Let E be an example and Q be the corresponding *query* whose refutation should result in an answer substitution

[1] We need two different relations because the decision list representation restricts the modes of the predicates to `syn_msd(+,?)` and `ana_msd(+,?)`.

that would make $Q = E$. Let Q' be the actual result of executing Q. Then Q' is considered to be a negative example if the training set does not contain Q'. Output completeness is a form of closed world assumption which assumes that every related "variant" of an example is included in the training set.

```
Let PTC be the (positive) examples to be covered
Let CPE be the set of covered (positive) examples initially empty
Let DL be the decision list being learnt initially empty
While PTC not empty
      DO
          Let x be an (arbitrary) example in PTC
          Let GC = { (G,0,0,0,0) | G is a clause that covers x }
          For each example e in PTC
              For each (G,SP,SN,QP,QN) in GC
                  if G covers e positively
                    then (G,SP,SN,QP,QN) := (G,SP,SN,QP+1,QN)
                    else if G covers e negatively
                        then (G,SP,SN,QP,QN) := (G,SP,SN,QP,QN+1) endif
                  endif
          For each example e in CPE
              For each (G,SP,SN,QP,QN) in GC
                  if G covers e positively
                    then (G,SP,SN,QP,QN) := (G,SP+1,SN,QP,QN)
                    else if G covers e negatively
                        then (G,SP,SN,QP,QN) := (G,SP,SN+1,QP,QN) endif
                  endif
          Let Best ∈ GC be such that gain(BEST) is maximum
          For each example e in PTC
              if Best covers e positively
                  then
                    CPE := CPE ∪ {e}
                    PTC := PTC − {e}
              endif
          For each example e in CPE
              if Best covers e negatively
                  then
                    CPE := CPE − {e}
                    PTC := PTC ∪ {e}
              endif
          Add Best to top of DL
      ENDDO
```

Fig. 1. CLOG algorithm

Our experiments show that CLOG is significantly more efficient than FOIDL in the induction process. On the task of analysis of the plural of English nouns, we ran FOIDL and CLOG on subsets of the training set of size 100, 200, 300, 400, 500, 600, 700, 800, 900 and the whole set of 1063 examples. The running times for CLOG were 21, 61, 112, 204, 277, 477, 646, 941, 1198, and 1771 seconds,

respectively. For FOIDL the running times of up to 700 examples were 254, 1290, 2191, 5190, 9521, 18353, and 33915 seconds, respectively (all times on a SUN SPARC 10).

With FOIDL we were therefore forced to cut the training set example sizes (to 200 per concept — even so, learning of all 576 concepts still took about 5 days CPU time), while we were able to run CLOG on the whole dataset (on average 317 examples per concept, 751 max, 1 min). This gave us a rule set with significantly more predictive accuracy. Although a detailed analysis of the reasons behind CLOG's efficiency is beyond the scope of the current paper, we believe there are two contributing reasons.

Firstly, CLOG only considers generalisations that are relevant to an example (cf. Progol [8]). This helps in focussing the search to only those clauses that are relevant with respect to an example. In the current implementation these generalisations are supplied by a user-defined predicate `generate_clauses(+Example, -Clauses)` which takes as input an example and generates a list of all generalisations that cover that example. An alternative would have been to provide mode declarations (cf. Progol [8]) which could then be used to build clauses incrementally. However, since our primary aim was to build a system for learning morphological relations this generality was not essential and we chose the simplest approach.

Secondly, CLOG treats the set of generalisations of an example as a *generalisation set*. It then cycles every input example through the generalisation set in a single iteration checking whether a candidate generalisation covers the example positively or negatively. Once this process is complete the "best" candidate generalisation is chosen. The example set is pruned using this candidate and the cycle repeats. There are two advantages of this approach - **1.** Every example is accessed once (and used many times) and **2.** The example set is accessed sequentially. This means that the example set can be kept in a file as opposed to being kept in memory. This allows CLOG to handle large training sets.

The algorithm (see fig. 1) maintains two sets — namely examples yet to be covered and examples already covered. The examples which have already been covered can get "uncovered" by subsequent rules which results in such examples needing to be covered again. This is important since CLOG uses a hill-climbing strategy (as does FOIDL) and hence is sensitive to the example order. However, this sensitivity is avoided if generalisations from subsequent examples override a previously learnt rule by providing better *gain*. This turns out to be usually the case given enough redundancy in the training data.

The *gain* function currently used in CLOG is user-defined. For the morphology learning problem we used a simple function that maximises the number of new positive examples covered against the sum-total of implicit negatives covered:

$$gain((G, SP, SN, QP, QN)) = QP - QN - SN$$

where: *G is the clause being learnt,*
QP,QN: number of new examples covered positively and negatively
SP,SN: number of already covered examples covered positively and negatively

3.1 Problem representation

A morphological transformation rule transforms a word in one form to another form. For example, the past morphological rule in English would transform the word *sleep* (resp. *walk*) to the word *slept* (resp. *walked*). For languages that employ *concatenative* morphology such as the majority of European languages, different forms of the same word are realised by changing the *prefix* and *suffix* of words. Thus, *slept* (resp. *walk*) can be derived from *sleep* (resp. *walked*) by changing the suffix *-ep* (resp. *-∅*) to the suffix *-pt* (resp. *-ed*).

We chose the following morphological rule representation:

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

where the auxilliary predicate `mate/6` is defined by:

```
mate(W1,W2,[],[],Y1,[]):- split(W1,W2,Y1).
mate(W1,W2,[],[],[],Y2):- split(W2,W1,Y2).
mate(W1,W2,[],[],Y1,Y2):- split(W1,X,Y1),   split(W2,X,Y2).
mate(W1,W2,P1,P2,Y1,Y2):- split(W1,P1,W11), split(W2,P2,W22),
                          split(W11,X,Y1),  split(W22,X,Y2).
split([X,Y|Z],[X],[Y|Z]).
split([X|Y],[X|Z],W) :- split(Y,Z,W).
```

This representation contrasts with the representation used in [7, 3]:

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

Essentially, the predicate `mate(W1,W2,P1,P2,S1,S2)` is true if `P1` and `S1` is the prefix and the suffix of `W1` respectively and similarly for `P1`, `S1` and `W2`. Although, this representation is restrictive in that it can handle only prefixation and suffixation operations we consider it to be sufficient for our task taking advantage of the linguistic fact that the languages we are dealing with employ concatenative morphology. Our representation has the benefit that it can easily be understood by linguists.

The *generalisation set* is constructed by the following predicate which generates the set of all prefixes and suffixes of a morphologically related pair of words.

```
generate_clauses(past(U,V),Clauses):-
       bagof( ( past(X,Y) :- (mate(X,Y,P1,P2,S1,S2), !)),
              mate(U,V,P1,P2,S1,S2),
              Clauses).
```

One limitation of the current implementation is that the size of the generalisation set cannot be too large. For the morphology task the size of the generalisation set ranged from **7** to **61** per example averaging between **22-35** for a typical example. However, this limitation can be avoided by a top-down search of the hypothesis space (*cf.* Progol [8]). We hope to address this in our future implementation.

4 Experiments and Results

In our experiments we used FOIDL and CLOG to perform two sets of experiments, the first concerning synthesis and the second analysis of word-forms. For each MSD, a set of rules for the predicate `syn_msd` was induced: the induced rules generate the oblique form from a given lemma. The input and output arguments of the `syn_msd` predicate are switched for the `ana_msd` predicate: the task of FOIDL/CLOG was to learn rules that produce the base form of the word given the oblique form. Apart from exchanging the input and the output, the set-up for the synthesis and analysis experiments was identical. There were altogether 288 MSDs in the five languages.

As has been mentioned, the training sets were taken from the first three parts of "*1984*". Due to FOIDL's computational efficiency limits and the large number of relations to be induced, the 200 (most frequent) examples were chosen for training for each MSD, where more than 200 were available. CLOG, on the other hand, does not have efficiency problems, so the complete training set could be used. In order to compare CLOG with FOIDL fully, we ran CLOG twice, once on the same training set as FOIDL (i.e. cut at 200 examples), and once on the full training sets. The Appendix of the novel was used for the test set. While the whole "*1984*" has approx. 100,000 words, the Appendix has only approx. 4,000. It therefore happens that a few rare MSDs were not represented in the test set (6 for Romanian, 8 for Slovene).

The set-up for the experiment was as for the orthographic past tense learning experiment: for synthesis the training data were encoded as Prolog facts of the form `syn_msd(lemma, oblique)` and for analysis as Prolog facts of the form `ana_msd(oblique, lemma)`. In both cases, the first argument of each target predicate is an input argument and the second is an output argument. The predicate `split` was used as background knowledge with FOIDL, and `mate` with CLOG. Constant prefixes and suffixes were allowed in the rules.

4.1 Results

For a start, let us take a look at the sets of rules induced for the particular task of synthesising the genitive singular of Slovene feminine nouns. The complete training set for this concept contained 608 examples, which were cut to the most frequent 200. The testing set contained 313 examples.

Comparing FOIDL with CLOG on the same training and test set shows that FOIDL slightly outperforms CLOG on the `n0fsg` concept, with a 97.4% accuracy (8 errors) for the former vs. 96.2% (12 errors) for the latter. In all cases, the errors of FOIDL were due to a wrong lemma being proposed, while the errors of CLOG were caused by the predicate failing.

The rule set induced by FOIDL consists of five exceptions and three generalizations; the generalizations are listed below.

```
n0fsg(A,B) :- split(A,C,[e,n]), split(B,C,[n,i]).
n0fsg(A,B) :- split(A,C,[a]),   split(B,C,[e]).
n0fsg(A,B) :- split(B,A,[i]).
```

From the bottom up, the first rule describes the formation of genitive for feminine nouns of the canonical second declension where *-i* is added to the nominative singular (lemma) to obtain the genitive. The second rule deals with the canonical first declension, where the lemma ending *-a* is replaced by *-e* to obtain the genitive. Finally, the third rule deals with nouns of the second declension that exhibit a common morpho-phonological alteration in Slovene, the schwa elision. Namely, if a schwa (weak *-e-*) appears in the last syllable of the word when it has the *-0* ending, this schwa is dropped with non-null endings: *bolezen-0*, but *bolezn-i*. However, this alternation does not affect only second declension feminine nouns but practically all inflecting words of Slovene. FOIDL attempts to deal with other examples of this alternation on a case by case basis: of the five exceptions, three exhibit this alternation.

An analysis of FOIDL's test set errors on `n0fsg` reveals that all bar one are due to a noun exhibiting schwa elision, which incorrectly triggers the default second declension *+i* rule, e.g., **vrnitevi* instead of *vrnitve*, from the base form *vrnitev*.

Running CLOG on the same training set (CLOG 200), the induced rules set is somewhat larger, and consists of eleven exceptions and six generalizations, with the generalzations listed below:

```
n0fsg(A,B) :- mate(A,B,[],[],[e,d],[e,d,i]),!.
n0fsg(A,B) :- mate(A,B,[],[],[e,n],[n,i]),!.
n0fsg(A,B) :- mate(A,B,[],[],[ccaron],[ccaron,i]),!.
n0fsg(A,B) :- mate(A,B,[],[],[r],[r,i]),!.
n0fsg(A,B) :- mate(A,B,[],[],[t],[t,i]),!.
n0fsg(A,B) :- mate(A,B,[],[],[a],[e]),!.
```

All the FOIDL exceptions are also included in the exceptions of CLOG. Additionally, the first two of FOIDL's generalisations are also induced by CLOG. But where FOIDL differs from CLOG is in the second declension default discussed above; while FOIDL posits the default rule, CLOG is more conservative and attempts to model it by a series of exceptions and partial generalisations. Ultimately, neither method is sufficient to correctly predict all the cases, as neither is able to successfully model the schwa elision. While FOIDL errors are all due to schwa elision, the CLOG errors are due to a combination of schwa elision errors and second declension errors. Although the number of errors is here slightly greater, the errors of CLOG are less severe; as has been mentioned, CLOG fails on its errors, while FOIDL proposes an incorrect lemma.

We next give the average overall results on all the languages of the dataset. The 288 programs learned by FOIDL and CLOG for the synthesis and analysis concepts show varying degrees of success in capturing the relevant morphological generalizations. Table 1 gives an overview of the results obtained by testing the induced programs. Three tests were performed; one for FOIDL, with the training sets, where necessary, cut to 200 examples, then for CLOG on the same training sets, and finally for CLOG on the full training sets.

For each experiment and for each language, the results on the `syn`thesis and `ana`lysis tasks are listed, first for nouns and adjectives separately (`*n` and `*a`) (except for English, where there is only one MSD), then aggregated (`*`). The

			FOIDL 200		CLOG 200		CLOG	
	syn	*	94.07%	21/6	94.73%	19/6	98.02%	38/17
	ana	*	93.85%	22/5	93.42%	21/3	96.05%	65/9
English	*	*	93.96%	43/11	94.07%	40/9	97.03%	103/26
		a*	95.91%	91/32	95.33%	103/33	97.23%	135/41
	syn	n*	92.00%	298/99	89.87%	382/100	93.77%	500/140
		*	93.01%	389/131	91.28%	485/133	94.66%	635/181
		a*	94.89%	101/27	94.60%	115/29	96.35%	159/38
	ana	n*	86.89%	422/144	86.84%	476/120	91.24%	724/181
		*	88.95%	523/171	88.84%	591/149	92.56%	883/219
Romanian	*	*	90.98%	912/302	90.06%	1076/282	93.61%	1518/400
		a*	98.98%	144/76	98.97%	155/79	98.77%	276/112
	syn	n*	93.85%	865/326	93.22%	1087/415	95.67%	1615/562
		*	96.61%	1009/402	96.32%	1242/494	97.34%	1891/674
		a*	98.90%	178/96	98.90%	197/93	99.06%	375/110
	ana	n*	92.53%	1054/375	91.14%	1285/375	94.76%	1909/550
		*	95.96%	1232/471	95.32%	1482/468	97.08%	2284/660
Czech	*	*	96.29%	2241/873	95.82%	2724/962	97.21%	4175/1334
		n*	95.92%	819/323	96.21%	904/305	97.78%	1114/372
	syn	a*	77.49%	1675/877	82.26%	1741/729	86.79%	3983/1106
		*	85.48%	2494/1200	88.31%	2645/1034	91.56%	5097/1478
		n*	95.01%	1014/404	95.76%	1001/371	97.12%	1194/457
	ana	a*	90.19%	1530/822	92.90%	1610/635	96.81%	2043/966
		*	92.28%	2544/1226	94.14%	2611/1006	96.95%	3237/1423
Slovene	*	*	88.88%	5038/2426	91.22%	5256/2040	94.25%	8334/2901
		a*	88.5%	512/181	85.51%	639/186	88.03%	771/230
	syn	n*	73.24%	1463/396	65.44%	1915/415	81.53%	3691/981
		*	78.19%	1975/577	71.94%	2554/601	83.64%	4462/1211
		a*	87.97%	520/223	86.45%	616/207	89.13%	756/260
	ana	n*	76.32%	1235/376	74.60%	1680/382	86.29%	3084/758
		*	80.10%	1755/599	78.45%	2296/589	87.21%	3840/1018
Estonian	*	*	79.14%	3730/1176	75.20%	4850/1190	85.42%	8302/2229

Table 1. Accuracy and complexity of FOIDL and CLOG rules

induced rules for each MSD are applied to the testing cases of that MSD and the incorrectly predicted cases recorded. The number of test cases and incorrectly predicted cases are then summed over all noun MSDs, all adjective MSDs, and all MSDs. On the basis of these sums we then compute the percentage of the correctly predicted cases in the test set, which is given as the first entry in each cell of the table. The second cell entry gives the total number of clauses in the concept decision list and the number of generalisations that appear in it, e.g. 21/6. This is an estimate of the complexity of the rules induced by FOIDL and CLOG.

Comparing FOIDL 200 with CLOG 200 we can see that there is no great systematic difference between the accuracies: the difference in all cases, except Estonian, is around 1%. However, FOIDL is somewhat more compact in expressing the concepts, having smaller theory sizes, both with exceptions and generalisa-

tion. Unsurprisingly, CLOG run on the complete training set greatly outperforms FOIDL. As has been mentioned, speed of induction is the main advantage of CLOG over FOIDL: even with full training sets, CLOG needs only a fraction of the time that FOIDL does with the reduced training sets.

5 Conclusions

We have presented the ILP system CLOG and used it to learn rules for synthesizing and analyzing inflectional forms of nouns and adjectives for English, Romanian, Czech, Slovene, and Estonian. The accuracy of the obtained rule-sets was evaluated and compared to the rule-sets obtained with the FOIDL system. The two systems have comparative performance given the same training set. However, while FOIDL is limited as to the ammount of training data it can handle due to it's computational efficiency, CLOG suffers no such bottleneck. Tests on the full training set show that CLOG can outperform FOIDL significantly.

Further work will focus on improving the induced rules by using additional linguistic background knowledge, esp. for capturing (morpho-)phonological regularities and on using the improved rules to perform preliminary analysis of word forms appearing in corpora, producing input for further text processing, e.g., part-of-speech tagging.

Acknowledgements

The authors would like to thank Alan Frisch and two anonymous reviewers for useful comments. This work was supported in part by the project ESPRIT IV 20237 ILP2.

References

1. J. Cussens. Part-of-speech tagging using Progol. In *Proc. 7th Intl. Wshp. on Inductive Logic Programming*, pages 93–108. Springer, Berlin, 1997.
2. W. Daelemans, T. Weijters, and A. van der Vosch, editors. *Proc. ECML-97 Workshop on Empirical Learning of Natural Language Processing Tasks.* Prague, Czech Republic, 1997.
3. S. Džeroski and T. Erjavec. Induction of Slovene nominal paradigms. In *Proc. 7th Intl. Wshp. on Inductive Logic Programming*, pages 141–148. Springer, Berlin, 1997.
4. T. Erjavec, N. Ide, V. Petkevič, and J. Véronis. MULTEXT-East: Multilingual text tools and corpora for Central and Eastern European languages. In *Proc. 1st TELRI European Seminar*, pages 87–98. Tihany, Hungary, 1995.
5. T. Erjavec, M. Monachini (eds.). Specifications and Notation for Lexicon Encoding. MULTEXT-East Final Report D1.1F, Ljubljana, IJS, 1997.
6. R. J. Mooney. Inductive logic programming for natural language processing. In *Proc. 6th Intl. Wshp. on Inductive Logic Programming*, pages 3–22. Springer, Berlin, 1997.
7. 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.
8. S. Muggleton. Inverse entailment and Progol, *New Generation Computing*,(13):245–286, 1995.