Morphosyntactic Tagging of Slovene Using Progol

James Cussens[1], Sašo Džeroski[2], and Tomaž Erjavec[2]

[1] University of York,
Heslington, York YO10 5DD, UK
`jc@cs.york.ac.uk`
[2] Department for Intelligent Systems, Jožef Stefan Institute,
Jamova 39, SI-1000 Ljubljana, Slovenia
`saso.dzeroski@ijs.si`, `tomaz.erjavec@ijs.si`

Abstract. We consider the task of tagging Slovene words with morphosyntactic descriptions (MSDs). MSDs contain not only part-of-speech information but also attributes such as gender and case. In the case of Slovene there are 2,083 possible MSDs. P-Progol was used to learn morphosyntactic disambiguation rules from annotated data (consisting of 161,314 examples) produced by the MULTEXT-East project. P-Progol produced 1,148 rules taking 36 hours. Using simple grammatical background knowledge, e.g. looking for case disagreement, P-Progol induced 4,094 clauses in eight parallel runs. These rules have proved effective at detecting and explaining incorrect MSD annotations in an independent test set, but have not so far produced a tagger comparable to other existing taggers in terms of accuracy.

1 Introduction

While tagging has been extensively studied for English and some other Western European languages, much less work has been done on Slavic languages. The results for English do not necessarily carry over to these languages. The tagsets for Slavic languages are typically much larger (over 1000), due to their many inflectional features; on the other hand, training corpora tend to be smaller.

In work related to this [9] a number of taggers were applied to the problem of tagging Slovene. Four different taggers were trained and tested on a hand annotated corpus of Slovene, the translation of the novel '1984' by G. Orwell. The taggers tested were the HMM tagger [6,15], Brill's Rule based tagger [3], the Maximum Entropy Tagger [14], and the Memory-based Tagger [7]. Accuracies on 'known' words were mostly a little over 90%, with the Memory-Based Tagger achieving 93.58%. Known words are those found in a lexicon that accompanies the corpus. Our goal here was to see whether ILP (specifically P-Progol) could be used to learn rules for tagging, to analyse the rules and to compare empirically with these other approaches to tagging.

2 Morphosyntactic Descriptions

The EU-funded MULTEXT-East project [8] developed corpora, lexica and tools for six Central and East-European languages. The centrepiece of the corpus is the novel "1984" by George Orwell, in the English original and translations [11]. The novel has been hand-tagged with disambiguated morphosyntactic descriptions (MSDs) and lemmas. The novel is marked up for sentences and tokens; these can be either punctuation or words. Each punctuation symbol has its own corpus tag (e.g. XMDASH), while the words are marked by their morphosyntactic descriptions.

The syntax and semantics of the MULTEXT-East MSDs are given in the *morphosyntactic specifications* of the project [10]. These specifications have been developed in the formalism and on the basis of specifications for six Western European languages of the EU MULTEXT project [1]; the MULTEXT project produced its specifications in cooperation with EAGLES (Expert Advisory Group on Language Engineering Standards) [4].

The MULTEXT-East morphosyntactic specifications contain, along with introductory matter, also:

1. the list of defined categories (parts-of-speech)
2. common tables of attribute-values
3. language particular tables

Of the MULTEXT-East *categories*, Slovene uses Noun (N), Verb (V), Adjective (A), Pronoun (P), Adverb (R), Adposition (S),[1] Conjunction (C), Numeral (M), Interjection (I), Residual (X),[2] Abbreviation (Y), and Particle (Q).

The morphosyntactic specifications provide the grammars for the MSDs of the MULTEXT-East languages. The greatest worth of these specifications is that they provide an attempt at a morphosyntactic encoding standardised across languages. In addition to already encompassing seven typologically very different languages, the structure of the specifications and of the MSDs is readily extensible to new languages.

To give an impression of the information content of the Slovene MSDs and their distribution, Table 1 gives, for each category, the number of *attributes* in the category, the total number of *values* for all attributes in the category; the number of different MSDs in the *lexicon*, and, finally, in the annotated MULTEXT-East Slovene '1984' *corpus.*

To exemplify the annotation use, Fig 1 gives the MSD-annotated first sentence of the Slovene translation of the novel: *It was a bright cold day in April, and the clocks were striking thirteen.* The `Bil/Vcps-sma` annotation shows that "Bil" is a singular masculine past participle in the active voice, and the `jasen/Afpmsnn` annotation shows that "jasen" is an adjective which is indefinite, nominative, singular, masculine, positive and qualificative.

[1] Adpositions include prepositions and postpositions; Slovene uses only prepositions.

[2] Residual is a category encompassing unknown (unanalysable) lexical items. It appears only once in the Slovene lexicon, for `2+2=5`; in our experiment we also used it to mark punctuation.

Table 1. Slovene morphosyntactic distribution

PoS	Att	Val	Lex	Cor
Noun	5	16	99	74
Verb	8	26	128	93
Adjective	7	22	279	169
Pronoun	11	36	1,335	594
Adverb	2	4	3	3
Adposition	3	8	6	6
Conjunction	2	4	3	2
Numeral	7	23	226	80
Interjection	0	0	1	1
Residual	0	0	1	1
Abbreviation	0	0	1	1
Particle	0	0	1	1
All	45	139	2,083	1,025

```
Bil/Vcps-sma je/Vcip3s--n jasen/Afpmsnn &comma;/XCOMMA
mrzel/Afpmsnn aprilski/Aopmsn dan/Ncmsn in/Ccs
ure/Ncfpn so/Vcip3p--n bile/Vmps-pfa trinajst/Mcnpnl &period;/XPERIOD
```

Fig. 1. First MSD-annotated sentence of Slovene translation of Orwell's "1984"

3 Method

Following the basic approach taken in [5,12], we used ILP to learn MSD elimination rules each of which identify a set of MSDs that cannot be correct for a word in a particular context. The context for a word is given as the MSDs of all words to the left and to the right of the word. Not using the actual words in the context simplified the learning, and is justified on the grounds that MSDs (unlike, say, Penn Treebank PoS tags) provide very specific information about the words.

The MULTEXT-East lexicon provides an *ambiguity class* for the Slovene words appearing in the corpus. For a given word, this is the set of possible MSDs for that word. Elimination rules can then be applied to reduce this ambiguity class in a particular context, ideally reducing it to a single MSDs. Note that each rule requires a word's context to be sufficiently disambiguated so that it can fire. This motivates using elimination rules in tandem with another tagger.

3.1 Examples

Each ambiguous word generated a single negative example and one or more positive examples. Each negative example is represented as a triple of left context, correct MSD and right context. The correct MSD generates a negative example for the induction of elimination rules since it identifies an MSD which it would be incorrect to eliminate. Positive examples contain MSDs which are incorrect (positive examples of MSDs to eliminate) but which are in the focus word's ambiguity class.

The left context is reversed so that the MSD immediately to the left of the focus is at the head of the list. Figure 2 show two positive and one negative example generated from a single occurrence of a word with ambiguity class `{pp3fsg__y_n,vmip3s__n,vcip3s__n}`. In this context, `vcip3s__n` is the correct MSD and hence appears in a negative example of elimination. MSDs are represented as constants for efficiency. We used exactly the same training data that had been used for training the other taggers in [9]. This data, together with other resources to reproduce our experiments, can be found at: `http://alibaba.ijs.si/et/project/LLL/tag/`. We produced 99,261 positive and 81,805 negative examples giving a total of 181,066 examples.

```
%rmv(LeftReversed,Focus,Right)
%2 POSITIVE EXAMPLES
rmv([vcps_sma],[pp3fsg__y_n],[afpmsnn,xcomma,afpmsnn,aopmsn,ncmsn,ccs,
ncfpn,vcip3p__n,vmps_pfa,mcnpnl,xperiod]).
rmv([vcps_sma],[vmip3s__n],[afpmsnn,xcomma,afpmsnn,aopmsn,ncmsn,ccs,
ncfpn,vcip3p__n,vmps_pfa,mcnpnl,xperiod]).

%1 NEGATIVE EXAMPLE
rmv([vcps_sma],[vcip3s__n],[afpmsnn,xcomma,afpmsnn,aopmsn,ncmsn,ccs,
ncfpn,vcip3p__n,vmps_pfa,mcnpnl,xperiod]).
```

Fig. 2. Examples created from a single occurrence of an ambiguous word

3.2 Background Knowledge

The use of ILP for tagging is particularly well motivated when the tags (here MSDs) have considerable structure. The background knowledge was designed to take advantage of that structure. Figure 3 shows some of the background predicates used.

Working through Figure 3, we have firstly, `msd/2` which explodes MSDs from constants into lists so that other predicates can extract the relevant structure from the MSDs; there were 1703 such `msd/2` facts in our background knowledge. Many of the background predicates consume an initial portion of a word's context (left or right) and return the remainder of the context as an output in the second argument. For example, `noun(A,B)` is true if `A` begins with a noun and is followed by `B`. We have the predicates `gender/3`, `case/3` and `number/3`, which identify gender, case and number or fail for MSDs where these are not defined. Figure 3 shows two of the `gender/2` clauses which show that the gender identifier is the 3rd attribute for noun MSDs and the 4th for pronoun. The most important predicates are `disoncase/2` `disongender/2` and `disonnumb/2` which indicate when two MSDs disagree in case, gender or number.

We also have simple phrasal definitions. Noun phrases `np/1` are defined as zero or more adjectives followed by one or more nouns. This is clearly not a full definition of noun phrase, but is included on the grounds that the simple noun phrases so defined will be useful features for the elimination rules. Finally, we have `isa/2` which identifies particular MSDs and `skip_over/2` which is used to skip over apparently unimportant tokens which do not have case, number or gender defined.

```
%EXPLODING MSD CONSTANTS
msd(afcfda, [a,f,c,f,d,a]).
msd(afcfdg, [a,f,c,f,d,g]).
msd(afcfdl, [a,f,c,f,d,l]).

%Parts of speech, always first letter
noun([M|T],T) :- msd(M,[n|_]).
verb([M|T],T) :- msd(M,[v|_]).

%GENDER
gender([M|T],Gender,T) :- msd(M,[n,_,Gender|_]).
gender([M|T],Gender,T) :- msd(M,[p,_,_,Gender|_]).

%DISAGREEMENT ON CASE, GENDER OR NUMBER
disoncase(M1,M2) :- case(M1,C1,_), case(M2,C2,_), \+ C1 = C2.
disongender(M1,M2) :- gender(M1,C1,_), gender(M2,C2,_), \+ C1 = C2.
disonnumb(M1,M2) :- numb(M1,C1,_), numb(M2,C2,_), \+ C1 = C2.

%NOUN PHRASE
np(A,B) :- adjective_star(A,C), noun_plus(C,B).
%and backwards ..
np1(A,B) :- noun_plus(A,C), adjective_star(C,B).

noun_plus(A,B) :- noun(A,B).
noun_plus(A,B) :- noun(A,C), noun_plus(C,B).

%IDENTIFYING PARTICUAR MSDS
isa([H|T],H,T).

%FOR SKIPPING TO IMPORTANT WORDS
skip_over(A,B) :-
        all_undefined_plus(A,B),
        some_defined(B).
```

Fig. 3. Excerpt of background knowledge

3.3 Splitting the Data

P-Progol is currently unable to accept 181,065 (fairly complex) examples directly. The data was therefore split according to the part-of-speech of the focus MSD (the 3rd argument of the examples). This formed the 8 data sets for Noun (n), Verb (v), Adjective (a), Pronoun (p), Adverb (r), Adposition (s), Numeral (m) and Other (o) described in Table 2. The "Other" dataset covered conjunctions (c) and particles (q) together. Although large by ILP standards, each of these datasets was sufficiently small for P-Progol 2.4.7 running on Yap Prolog 4.1.15.

Although motivated by pragmatics, this splitting had a number of beneficial effects. The split meant that all eight datasets could have been processed in parallel as eight separate Yap processes. In fact, due to a lack of suitable machines the work was spread between the first author's Viglen laptop (233 MHz Pentium, 80 MBytes RAM) and Steve Moyle's PC (266 MHz, 128 MBytes RAM). These machines are denoted Y and O respectively in Table 2. Since we had 8 rule sets induced for specific parts-of-speech we were able to index on the part-of-speech by altering the induced rules to have the relevant part-of-speech as a first argument.

In effect, we performed a single initial greedy split of the data as would be done as the first step in a decision tree inducer such as TILDE[2]. Since many of the clauses induced in earlier work on random samples of the complete data set were specific to a particular part-of-speech (e.g. `rmv(L,F,R) :- noun(L,L2) ...`), we will not have missed many good clauses as a result of our greediness.

3.4 P-Progol Parameters and Constraints

As well as limiting the amount of data input to a particular P-Progol run, we also constrained the Progol search in two major ways. The basic Progol algorithm consists of taking a 'seed' uncovered positive example, producing a most specific 'bottom clause' which covers it and then using the bottom clause to guide the search for the 'best' clause that covers the seed.

P-Progol has a number of built-in cost functions: the 'best' clause is that which minimises this cost. In this work, we choose m-estimation [13, p.179–180] to estimate the accuracy of clauses, and searched for the clause that maximised estimated accuracy. An m value of 1 was chosen. Such a low value of m might allow overfitting, so as a guard against this, only clauses which covered at least 10 positives were allowed. Such a stopping rule has the advantage of allowing the search to be pruned. If a clause dips below 10 positives then there is no point considering any specialisations of that clause, since they will also cover fewer than 10 positives. Also, only clauses with at least 97% training accuracy were allowed.

Two more constraints were required for learning to be feasible. Firstly, we restricted each Progol search to a maximum of 5000 clauses—many searches hit this threshold. Secondly, we limited clauses to a maximum of four literals, the only exception being the Numeral (m) run because of its small example set. Caching ([5]) was only used on two small runs to avoid any risk of running out of RAM.

4 Results

4.1 The Induction Process

Considerable effort was expended in tracking down bugs in earlier versions of Yap Prolog which involved indexing problems for large data sets. Ashwin Srinivasan and the first author also implemented improvements to P-Progol which considerably improved its efficiency. Despite these (productive) efforts, the large number of examples and the nature of the Progol search meant long P-Progol runs sometimes lasting a few days (see Table 2).

Table 2. Data set and induction statistics

PoS	Pos	Neg	Tot	Rules	Time (hrs)	Searches	Machine	Caching	$\|C\|$
a	29099	5709	34808	1148	36.1	1513	Y	on	4
m	1972	843	2815	223	2.6	305	Y	off	5
n	20125	14569	34694	809	54.0	2767	O	off	4
o	2279	21946	24225	36	15.8	1960	Y	on	4
p	26585	8480	35065	1291	54.5	2750	O	off	4
r	2500	4980	7480	42	5.3	1941	O	off	4
s	4865	6025	10890	90	2.3	293	Y	off	4
v	11836	19253	31089	455	25.6	1599	O	off	4
All	99261	81805	181066	4094	196.2	13128			

4.2 Structure of Induced Theories

P-Progol associates a clause label with each induced clause which gives the positive and negative cover, and the clause's 'score'. In our case the score was clause accuracy as estimated by the m-estimate. This allowed us to parameterise the induced theory, converting a clause such as

```
rmv(L,F,R) :- case(R,n,R2), disonnumb(F,R2), disonnumb(F,R).
```

from the adjective theory, to:

```
rmv(a,L,F,R,score(1283,1,0.999094)) :-
  case(R,n,R2), disonnumb(F,R2), disonnumb(F,R).
```

This allowed us to produce subsets of the complete theory by thresholding on the m-estimated accuracy ($EstAcc$). For example, filtering out all rules with $EstAcc < 0.999$ results in an under-general theory but with only ultra-reliable rules remaining. Note also the added "a" index which indicates that the rule only applies when `F` (the focus word) is an adjective.

The `isa/3` predicate appears very often in the induced theory, indicating the importance of MSD-specific rules. Also, as expected many of the rules look for disagreement between neighbouring words. In the *EstAcc* ≥ 0.999 exactly half of the 240 rules used the disagreement predicates. Many of those that did not, used two literals to specify particular disagreements.

In the complete theory, 2069 of the 4094 rules used only features of chunks (words or simple phrases) right next to the focus word. In the *EstAcc* ≥ 0.999 subtheory, 173 out of the 240 rules used only such features, showing that most highly reliable rules are quite simple, identifying anomalies between neighbouring words. Of the remaining 67 rules with *EstAcc* ≥ 0.999, all of them only looked one chunk beyond neighbouring chunks.

```
rmv(a,L,F,R,score(1130,0,0.999855)) :-
 case(F,n,D), numb(F,d,D), disonnumb(F,R).
rmv(a,L,F,R,score(748,0,0.999781)) :-
 case(R,l,R2), gender(R,f,R2), gender(F,m,_).
rmv(v,L,F,R,score(619,0,0.999001)) :-
 numb(F,p,D), gender(F,n,D), isa(L,vcip3s__n,L2).
rmv(n,L,F,R,score(600,0,0.999301)) :-
 numb(F,d,_), isa(L,spsl,L2).
rmv(n,L,F,R,score(433,0,0.999032)) :-
 case(F,a,_), isa(L,spsg,L2).
rmv(m,L,F,R,score(14,0,0.980036)) :-
 gender(F,f,_), isa(L,spsi,L2),
 numb(L2,_,L3), case(L3,_,L4).
```

Fig. 4. A subset of the induced disambiguation theory

4.3 Consistency Checking and Ambiguity Reduction

Here we tested the consistency of each test sentence with the rules. As Table 3 shows, a good half of the correct readings are deemed inadmissible by the complete theory. This is because it only takes one disambiguation rule to incorrectly fire for a whole sentence to be rejected.

Table 3. Proportion of test sentence annotations rejected

EstAcc >	0	96.0	97.0	98.0	98.5	99.0	99.5	99.7	99.9
	49.5	48.6	46.4	36.9	32.4	24.7	15.8	10.9	3.5

To measure ambiguity reduction, we selected those 263 sentences from the test set, which had fewer than 2000 possible annotations according to the ambiguity classes of the words in the sentences. Many sentences have millions of possible annotations, most of them plainly absurd, so we do not wish to receive credit for eliminating these. Table 4 shows the ambiguity reduction factor (ARF) for each subtheory: we summed the number of possible annotations for each sentences in the test set giving a total of 81634 annotations. To get the ARF we divided this number by the number of annotations consistent with the rules. We also give the rejection error rate (RER), the percentage of times that the annotation given in the test set was inconsistent with the rules.

Table 4. Per sentence ambiguity reduction factor and rejection rate, for sentences with fewer than 2000 possible annotations

EstAcc >	0	96.0	97.0	98.0	98.5	99.0	99.5	99.7	99.9
ARF	67.3	65.1	56.6	38.0	29.4	21.1	10.6	6.9	2.7
RER	25.5	24.7	22.1	17.5	13.3	8.7	3.0	2.3	0.4

The ambiguity reduction factor is good, even the *EstAcc* > 0.999 theory reduces sentence ambiguity by nearly a third. However, to use the rules to reduce ambiguity, we should be almost guaranteed not to reject the correct annotation; this means only the small theories composed only of highly reliable rules should be used.

4.4 Error Detection

The 0.4% RE for the *EstAcc* > 0.999 was due to a single test set annotation being rejected. The annotated test sentence was: "[Winston/`npmsn`] [in/`ccs`] [Julia/`npfsn`] [sta/`vcip3d__n`] [se/`px______y`] [očarana/`afpmdn`] [objela/`vmps_sfa`] [./`xperiod`]" ("Delighted, Winston and Julia embraced.") This annotation was rejected by this rule:

```
rmv(a,L,F,R,score(1130,0,0.999855)) :-
  case(F,n,D), numb(F,d,D), disonnumb(F,R).
```

on the grounds that dual nominative adjectives can not be followed by any word that does not have the same number. This rules out the "očarana/`afpmdn`, objela/`vmps_sfa`" annotation

Upon inspection we found that the rule was correct to reject this annotation: "objela" (embraced) should have been tagged as dual not singular. This lead us to use the *EstAcc* > 0.999 theory to look for other possible errors in the complete test set. To help us do this we wrote a simple Prolog interface which flagged possible errors and *explained why* they were suspected errors. Figure 5 shows how the interface flagged the "objela" error.

This demonstrates two points. Firstly our disambiguation rules can be used to detect incorrect annotations, *and provide an explanation of* why *the annotation is incorrect.* Secondly, our rules are constraints that apply not only to

```
**ERROR DETECTED**
[(Winston_BOS,npmsn),(in,ccs),(Julija,npfsn),(sta,vcip3d__n),
(se,px______y)]
ocxarana,afpmdn <= HERE
[(objela,vmps_sfa),(.,xperiod)]

Constraint number 1 confidence score(1130,0,0.999855)

because:
ocxarana,afpmdn is ambiguous, and is (apparently) an adjective

and we can not also have:
ocxarana,afpmdn with case: n
ocxarana,afpmdn with numb: d
objela,vmps_sfa and ocxarana,afpmdn disagreeing on number
**********
Enter y if this is a real error
|: |:
```

Fig. 5. Interface for annotation error flagging

the focus word. Here, "očarana", the focus word, was annotated correctly—the inconsistency was detected in its context.

However, of the 24 alleged test set errors flagged by our constraints, only 9 turned out to be actual errors. The other 15 were examples of rare atypical constructions. All the constraints which incorrectly flagged errors had $EstAcc > 0.999$. For example, one had covered 761 positives and 0 negatives in the training data. So the large number of errors is perhaps surprising and points to possible over-fitting with an inadequately large training dataset. On the other hand, as an annotation validation tool, performance is reasonable, and it would be easy to expand the explanations and allow the user to correct (real) errors as they are presented.

4.5 Tagging Accuracy

Our MSD elimination rules can not be used as a standalone tagger: they rely too heavily on disambiguated context and there is no guarantee that a single MSD will be returned for each word after incorrect MSDs have been eliminated. We propose that they can be used as a filter to reject inconsistent annotations produced by another tagger, such as those mentioned in Section 1.

Here we combine our rules with the simplest tagger—one that returns the most likely tag based on lexical statistics, without taking context into account. Our goal is to measure the degree to which accuracy increases once the rules are used to filter out incorrect annotations.

Due to problems with Sicstus Prolog, experiments were conducted on a subset of 526 of the original 650 test sentences. These were sentences with fewer than

2000 possible annotations. Choosing the most likely tag according to lexical statistics produces an accuracy of 83.3% on this test set. We combined this tagger with our rules by doing a uniform cost search i.e. (A^* with $h = 0$) for the most probable sentence annotation according to the lexical statistics, which was consistent with our rules.

Using the complete theory we achieved 86.6%. Some subtheories did a little better, for example the $EstAcc \geq 0.985$ accuracy was 87.5% So we have an improvement, albeit a modest one. Our experiments with error flagging reported in Section 4.4 indicate that a major barrier to improved performance is that our constraints frequently reject correct annotations.

It remains to be seen what improvement, if any, can be achieved when marrying our rules to more sophisticated taggers such as those mentioned in Section 1. Clearly the combination examined here has far lower performance than the taggers mentioned in Section 1.

5 Conclusions and Future Work

In this work, we have established the following positive results:

1. P-Progol can be applied directly to datasets of at least 30,000 examples. With appropriate use of sampling, it is likely that this could upper limit could be increased considerably.
2. We have induced MSD elimination rules which can be used to filter out incorrect annotations. The symbolic nature of the rules means that an *explanation* is also supplied. This makes using these rules particularly appropriate for an interactive system—we intend to use the rules induced here to check the existing MULTEXT-East corpus

We have also established the following negative result:

1. The performance of the MSD elimination rules as a standalone system or in tandem with a crude tagger based on lexical statistics is considerably worse than that of competing taggers.

Apart from checking the MULTEXT-East corpus with the rules, we also intend to use the rules to check the annotations proposed by the taggers mentioned in Section 1. By filtering out at least some incorrect annotations, the tagging accuracy should increase.

Acknowledgements

Special thanks are due to Steve Moyle for allowing us to run Yap/P-Progol on his machine—this was essential to get the work finished in time. Thanks to Ashwin Srinivasan for enhancements and bug fixes to P-Progol and to Vitor Santos Costa for allowing us to "go places no one has been before" with Yap4.1.15. Finally, thanks and apologies to Gill, Jane and Robert for putting up with the first author during the writing of this paper. The work presented here was supported by the ESPRIT IV Project 20237 ILP2.

References

1. N. Bel, Nicoletta Calzolari, and Monica Monachini (eds.). Common specifications and notation for lexicon encoding and preliminary proposal for the tagsets. MULTEXT Deliverable D1.6.1B, ILC, Pisa, 1995.
2. H. Blockeel and L. De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1–2):285–297, 1999.
3. Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.
4. Nicoleta Calzolari and John McNaught (eds.). Synopsis and Comparison of Morphosyntactic Phenomena Encoded in Lexicons and Corpora: A Common Proposal and Applications to European Languages. EAGLES Document EAG—CLWG—MORPHSYN/R, ILC, Pisa, 1996.
5. James Cussens. Part-of-speech tagging using Progol. In *Inductive Logic Programming: Proceedings of the 7th International Workshop (ILP-97). LNAI 1297*, pages 93–108. Springer, 1997.
6. D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 133–140, Trento, Italy, 1992.
7. Walter Daelemans, Jakub Zavrel, Peter Berck, and Steven Gillis. Mbt: A memory-based part of speech tagger-generator. In Eva Ejerhed and Ido Dagan, editors, *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 14–27, Copenhagen, 1996.
8. Ludmila Dimitrova, Tomaž Erjavec, Nancy Ide, Heiki-Jan Kaalep, Vladimír Petkevič, and Dan Tufiş. Multext-East: Parallel and Comparable Corpora and Lexicons for Six Central and Eastern European Languages. In *COLING-ACL '98*, pages 315–319, Montréal, Québec, Canada, 1998.
9. Sašo Džeroski, Tomaž Erjavec, and Jakub Zavrel. Morphosyntactic tagging of slovene: Evaluating pos taggers and tagsets. Technical Report IJS TechReport DP-8018, Jozef Stefan Institute, 1999.
10. Tomaž Erjavec and Monica Monachini (eds.). Specifications and notation for lexicon encoding. MULTEXT-East Final Report D1.1F, Jožef Stefan Institute, Ljubljana, December 1997. http://nl.ijs.si/ME/CD/docs/mte-d11f/.
11. Tomaž Erjavec and Nancy Ide. The MULTEXT-East corpus. In Antonio Rubio, Natividad Gallardo, Rosa Castro, and Antonio Tejada, editors, *First International Conference on Language Resources and Evaluation, LREC'98*, pages 971–974, Granada, 1998. ELRA. URL: http://ceres.ugr.es/ rubio/elra.html.
12. Nikolaj Lindberg and Martin Eineborg. Learning constraint grammar-style disambiguation rules using inductive logic programming. In *Proc. COLING/ACL98*, 1998.
13. Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.
14. Adwait Ratnaparkhi. A maximum entropy part of speech tagger. In *Proc. ACL-SIGDAT Conference on Empirical Methods in Natural Language Processing*, pages 491–497, Philadelphia, 1996.
15. Rene Steetskamp. An implementation of a probabilistic tagger. Master's thesis, TOSCA Research Group, University of Nijmegen, Nijmegen, 1995. 48 p.