Application of Different Learning Methods to Hungarian Part-of-speech Tagging

Tamás Horváth[1], Zoltán Alexin[2], Tibor Gyimóthy[3], and Stefan Wrobel[4,1]

[1] German National Research Center for Information Technology, GMD - AiS.KD, Schloß Birlinghoven, D-53754 Sankt Augustin, tamas.horvath@gmd.de
[2] Dept. of Applied Informatics, József Attila University, P.O.Box 652, H-6701 Szeged, alexin@inf.u-szeged.hu
[3] Research Group on Artificial Intelligence, Hungarian Academy of Sciences, Aradi vértanuk tere 1, H-6720 Szeged, gyimi@inf.u-szeged.hu
[4] Otto-von-Guericke-Universität Magdeburg, IWS, P.O.Box 4120, D-39106 Magdeburg, wrobel@iws.cs.uni-magdeburg.de

Abstract. From the point of view of computational linguistics, Hungarian is a difficult language due to its complex grammar and rich morphology. This means that even a common task such as part-of-speech tagging presents a new challenge for learning when looked at for the Hungarian language, especially given the fact that this language has fairly free word order. In this paper we therefore present a case study designed to illustrate the potential and limits of current ILP and non-ILP algorithms on the Hungarian POS-tagging task. We have selected the popular C4.5 and Progol systems as propositional and ILP representatives, adding experiments with our own methods AGLEARN, a C4.5 preprocessor based on attribute grammars, and the ILP approaches PHM and RIBL. The systems were compared on the Hungarian version of the multilingual morphosyntactically annotated MULTEXT-East TELRI corpus which consists of about 100.000 tokens. Experimental results indicate that Hungarian POS-tagging is indeed a challenging task for learning algorithms, that even simple background knowledge leads to large differences in accuracy, and that instance-based methods are promising approaches to POS tagging also for Hungarian. The paper also includes experiments with some different cascade connections of the taggers.

1 Introduction

Part-of-speech (POS) *tagging* is one of the first stages in natural language related processing (e.g., parsing, information extraction). The task of a POS tagger is, for a given text, to provide for each word in the text its contextually disambiguated part of speech *tag* representing the word's morphosyntactic category. Depending on source language grammar and application needs, a suitable tag set is chosen by linguists as basis for further analysis. Tagging is a difficult task in general because of the large number of ambiguities in natural language texts. For Finno-Ugric languages, like Hungarian in particular, this task is even more complex because they have rich morphology and fairly *free word-order*. While a reduced tag set containing 125 different tags overcomes the difficulty of the huge set of morphosyntactic categories in practice, the free-word order makes the POS tagging task for the Hungarian language a new challenging problem for learning algorithms.

In this paper, we therefore present a case study designed to illustrate the potential and limits of current ILP and non-ILP algorithms on the Hungarian POS-tagging task. We have selected the popular C4.5 and Progol systems as propositional and ILP representatives, adding experiments with our own methods AGLEARN, a C4.5 preprocessor based on attribute grammars, and the ILP approaches PHM and RIBL. For the experiments, we have used the annotated Hungarian TELRI corpus from the MULTEXT-East project which is currently the only available annotated corpus for Hungarian. The corpus contains about 100.000 tokens (which is relatively small compared to e.g. the Wall Street Journal corpus for English with its 3 million words). In the past, the MULTEXT-East corpus was already used to test Brill's tagger generator [4, 17] but there, only relatively poor per-word overall accuracy was reached, adding additional motivation to the experiments with a reduced tag set as described in the present paper.

The paper is organized as follows. In Section 2 we give a brief introduction to the POS tagging problem, describe previous work on generating taggers through learning, and then focus on issues related to the Hungarian language and the dataset that was used. In Section 3 we give our motivation for the design of our empirical case study, in particular the choice of learning algorithms. We also describe, for each chosen algorithm, how the POS tagging problem is mapped to the algorithm's internal representation. Section 4 contains the results of our experiments along with a discussion. The paper ends with a discussion and conclusions from our studies along with suggestions for further research.

2 POS Tagging and the Data Set

During language processing when a sentence is read each word is labelled by the set of its possible morphosyntactic descriptions, called *tags*. For example, the Hungarian word *'hét'* can be either a *number* ('seven') or a *noun* ('week'). In most of the cases a word's true tag is uniquely defined by its context in the sentence. Since each language has ambiguous words (i.e. that have several different taggings), a working tagger must contain a disambiguation module choosing the true tag from the assigned tag set for each word. Ideally, the task of such a module is defined as follows. *Given*

- a set S of correctly disambiguated tagged sentences,
- another correctly tagged set S_{query} of sentences containing ambiguous words,

find the prediction of the true tag for each ambiguous word from S_{query}.

2.1 Previous Work on Generating Taggers through Learning

Several methods have been developed for the automatic generation of POS taggers from annotated corpora. In [5] Cussens described a tagging system for English that has been generated by Progol (combined with a selection of the most frequent class in those cases where Progol's rules did not make a decision). Progol was trained on a corpus containing 3 million words and it generated "*removable*" rules for ambiguous tags: whenever such a rule applies to a word, the rule's tag is removed from the set of candidates. Using a simple linguistic background knowledge the generated tagger achieved 96.4% per word accuracy on the test data[1].

Daelemans & al. in [7] presented the MBT memory-based tagger generator system for different languages. They trained their system on a subset of the Wall Street Journal corpus (2 million words) for English and achieved 96.4% accuracy. For other languages they reached the following results: Dutch (95.7%), Spanish(97.8%) and Czech(93.6%).

Eineborg & Lindberg presented a tagging system for Swedish language [9]. They applied Progol and in a similar way to Cussens' approach they learned *removable* rules for ambiguous tags. The Swedish UMEÅ Corpus (1 million words) was used, and a 97% accuracy was achieved.

The learned taggers can be combined in order to get more accurate results. Halteren & al. in [13] discussed several possible combinations of four well-known tagger generator methods (HMM, Memory Based, Brill, Maximum Entropy). They showed that all combination taggers outperform their best component. Their best combined tagger named *pairwise-voting* obtained 97.92% accuracy for the LOB corpus (1 million words).

An application of the Brill's tagger generator to Hungarian is presented in [17]. The TELRI [11] corpus (100.000 words) was used but only 87.5% per word accuracy was reached. This empirical result showed that Brill's method has some difficulty with languages which are dissimilar in their characteristics from English.

2.2 The Data Set

Our experiments were based on an improved version of the TELRI corpus [11] in which several incorrect annotations had been corrected. This corpus was produced within the framework of "MULTEXT-East" project whose main goal was to prove the suitability of the tag coding convention MorphoSyntactic Description (MSD) to East-European languages. Orwell's famous novel, the "1984" was translated into six Eastern-European languages including Hungarian and annotated by MSD codes.

Words in the Hungarian language may have several thousand MSD codes. In order to reduce this huge set, linguists also propose the CTAG code convention

[1] Per word accuracy is computed on all words of the test text, i.e., including both unambiguous words (already tagged correctly by the lexicon) and ambiguous words (requiring the learned tagger module).

which leads to a smaller set containing only 125 different tags [19]. Since each CTAG code represents a set of MSD codes, the CTAG annotated version of a text contains at most as many ambiguities as its MSD annotation. Clearly, by using CTAGs we lose some linguistic information, however, as it is reported in [19], the CTAG annotation still usefully applies to a number of cases of processing Hungarian texts.

In our experiments we used the CTAG annotated Hungarian version of the TELRI corpus. From the corpus, we first created a tag lexicon by collecting for each word the list of tags that actually occurred in the corpus. This lexicon was then used to convert the sentences of the corpus into training data. For example the first sentence of the novel

"Derült, hideg áprilisi nap volt, az órák éppen tizenhármat ütöttek." [2]

was converted into the list

(asn, {asn, vmis3s}), wpunct, asn, asn, nsn, vmis3s,
wpunct, (t, {psn, t}), npn, rg, ms, vmis3p, spunct

If the tagging of a particular word is ambiguous (according to the lexicon) then the corresponding item is a pair the first element of which is the true tag (according to the annotation) and the second is the set of possible tags (according to the lexicon). Chapter 1 and 2 in the novel having been used as training data while chapter 3 and 4 served as test data. In Table 1 the number of tokens in the corpus is presented.

Category	Number of tokens		
	Training	Test	Altogether
word	59035	21673	80708
punctuation	12484	5235	17719
	71519	26908	98427

Table 1. The number of tokens in the Hungarian corpus

3 Empirical Evaluation Design

The goal of our case study was not to compare particular learning systems, but to examine the influence of the unusual and difficult properties of the Hungarian language (complex grammar, free word order) on the learning problem. We therefore selected a set of learning systems representative of different system classes that have been tried before on POS tagging tasks to see how their performance would be affected and whether phenomena observed in other experiments would carry over to Hungarian.

[2] It was a bright cold day in April, and the clocks were striking thirteen.

To this end, as reference standards we selected two learning systems representative of popular propositional and ILP learners used for tagging problems, C4.5 [20] and the C version of Progol 4.4 [18]. Since it appears that for other languages, memory-based methods are promising, we further included the instance-based ILP system RIBL [10, 3] in the comparison. To examine the strong influence of background knowledge (reported e.g. in [5]), we used AGLEARN, a learning system that adds additional attributes to the input of C4.5. These attributes are specified with the help of an attribute grammar and actually computed by a parser that is invoked on the context of the focus word. Finally, as we wanted to examine the influence of depth bounds (window sizes) on the learning problem, we included experiments with PHM [16, 14], a method that is capable of efficiently learning logic programs with colored path graph background knowledge without depth bound.

Below we sketch the particular representations used for each of these learning systems. However, let us first describe some general properties of our experimental setup. First, in contrast to experiments such as [5], where "*removable*" rules were learned, we learned "*choose*" rules, i.e., whenever such a rule applies, the rule's tag is made the chosen (i.e., predicted) tag. Second, in order to increase the specificity of the learned rules and also to reduce the high runtime complexity of the learning systems, we decided to split the learning problem by learning rules for each ambiguity class separately, and limited ourselves to those classes which occurred at least 50 times in the training set. These ambiguity classes represent 92.23% (6667 of 7229) of all ambiguity occurrences in the training set and 91.26% (2443 of 2677) of all ambiguous tokens in the test data set. In cases of C4.5, AGLEARN, and RIBL we considered a separate learning problem for each ambiguity class. For Progol and PHM a separate learning problem for $choose\_t_i\_from\_t_1\_\ldots\_t_k$ was created for each of the possible true tags for each ambiguity class $c = \{t_1, \ldots, t_k\}$.

In all *training* examples, the unique correct tags for surrounding words were used. In cases of C4.5, AGLEARN, Progol, and PHM, the learned set of rules was used for testing as follows: whenever exactly one rule applied, this was used as the learner's prediction; if none or more than one applied, the test example was taken as uncovered and was assigned, as in [5] to the most frequent candidate category. Note that a rule cannot be applied if it does not cover the focus word or it refers to at least one ambiguous token in the context of the focus word. Since the IBL paradigm is able to handle missing attribute values as well, in the case of RIBL such ambiguous tokens were considered as objects without class information.

In the rest of this section we describe briefly the different representations we employed. The results are discussed in Section 4 in details.

3.1 C4.5 and AGLEARN

In our experiments with C4.5 we represented the context of the focus words by fixed length lists of the tokens, the chosen window size being 7. Below we give an item from the C4.5 training data file:

`empty`, `empty`, ms, asn, nso, nsnx, wpunct, nsn, vmis3s, asn, npox, aso, t, asn, t

The first 7-7 columns describe the left, and right context respectively, and the last one is the true tag of the current token. If a context was shorter than 7 then it was padded with the special value `empty`. In order to extend the above description with some basic linguistic knowledge, we used the system AGLEARN [12]. AGLEARN is a learning method which extends a propositional learning table with extra columns containing "relational" information. The basic idea of AGLEARN is very similar to the ILP system LINUS [8], i.e. a relational learning problem is transformed into propositional form and the rules inferred by a propositional learner are transformed back into relational form. The difference is that AGLEARN uses *attribute grammar* formalism for the relational representation. Since AGLEARN applies C4.5, its application can be considered as an extension of the propositional language used by C4.5.

When setting up a learning task for AGLEARN a simple linguistic background knowledge was developed. The aim of this linguistic background knowledge was to recognize token groups and characteristic phrase structures called *syntagmas* in the context of the focus word. An attribute grammar was used for the computation of two new attribute values (group and syntagma). Group values were computed for the left and right contexts. The syntagma value was computed only for the right context. This value is based on the syntax of the context so it cannot be computed for the left context when the beginning of the syntactic structure is not known. AGLEARN generated training examples for C4.5 in which in addition to the CTAGs these new attribute values were appeared as new columns. Our findings showed that these new columns in several cases appeared in the learned rules [2].

3.2 Progol

As mentioned earlier, for each ambiguity class $c = \{t_1, \ldots, t_k\}$, and for each $t \in c$ a separate learning problem was considered over the target predicate `choose_t_from_c`/2. For each training token with ambiguity class c we added the positive (resp. negative) example `choose_t_from_c`(L, R) if its true tag is t (resp. is not t). Arguments L and R are the token's whole left (in reverse order) and right contexts, respectively (i.e., there are lists containing the true tags). In each Progol learning task we used the following "technical" background knowledge:

$$\bigcup_{s \in \mathcal{C}} \{s(s)\} \cup \{\mathrm{empty}([]), \mathrm{first}(T, [T|\_]), \mathrm{second}(T, [\_, T|\_]), \mathrm{third}(T, [\_, \_, T|\_])\} \ ,$$

where $\mathcal{C}$ denotes the set of CTAGs. Atoms `first`, `second`, and `third` were used to select the ith token from their second list arguments for $i = 1, 2, 3$, respectively. Note that the above background knowledge defines a window size of 3. Progol then learned rules e.g. of the form

$$\texttt{choose\_cp\_from\_cp\_rp}(L,R) : -\mathrm{first}(L1,L), \mathrm{rg}(L1), \mathrm{second}(R2,R), \mathrm{nsn}(R2) \ ,$$

meaning that a token's true tag is predicted by *cp* if its ambiguity class is $\{cp, rp\}$, its left neighbor is a token of true tag rg, and its second right neighbor has true tag nsn.

3.3 PHM

The Product Homomorphism Method (PHM) introduced in [16] is a combinatorial approach for learning simple logic programs. By using PHM, positive PAC learning results can be proved to different classes of *structured* background knowledges including the class of *colored path graphs* [16, 15, 14]. The POS problem is basically a *sequential* problem, that can be represented by *colored path graphs*. As PHM does not require any bound on the depth, in contrast to the other methods investigated, we do not need any window technique.

A colored directed graph is given by a tuple $G = (V, E, Q_1, \ldots, Q_l)$, where V denotes the set of vertices, $E \subseteq V \times V$ the set of edges, and $Q_i \subseteq V$ for $i = 1, \ldots, l$ the colors of the vertices. Thus, a vertex may have more than one or possible no color. G is a colored path graph if the directed graph (V, E) is a set of disjoint directed paths. Given a colored path graph background knowledge $\mathcal{B}$ and a set of facts $E = \{P(b_1), \ldots, P(b_t)\}$, $b_i \in V$ for $i = 1, \ldots, t$, the reduced RLGG of E with respect to $\mathcal{B}$ can be computed in time $O\left(l \cdot t \cdot L_{\max}\right)$ and space $O\left(l \cdot L_{\max}\right)$, where $L_{\max}$ is the length of the longest path in $\mathcal{B}$ [14].

In order to apply PHM, the POS problem must be represented by using colored path graph background knowledges. In our experiment with the PHM we used the next background knowledge $\mathcal{B}$ defined by the training set as follows:

- a constant identifying the jth word in sentence i was introduced for every possible i, j,
- for every consecutive words identified by a and b, respectively, a fact $R(a, b)$ was added to $\mathcal{B}$ (hence, the ground R-atoms in $\mathcal{B}$ form a set of disjoint directed paths), and
- for each constant a, a fact $t(a)$ was added to $\mathcal{B}$ where t is the true tag of the word identified by a (hence, t can be considered as a color).

For each ambiguity class $c = \{t_1, \ldots, t_k\}$ and for each possible tag $t \in c$ we considered a *separate* learning problem with background knowledge $\mathcal{B}$ and training examples $E^+(c,t) \cup E^-(c,t)$ with respect to a target predicate $\texttt{choose\_t\_from\_c}/1$, i.e.,

- $E^+(c,t)$ is the set of the atoms $\texttt{choose\_t\_from\_c}(a)$ for every training instance (token) a with ambiguity class c, and true tag t, and
- $E^-(c,t)$ is the set of the atoms $\texttt{choose\_t\_from\_c}(a)$ for every training instance a with ambiguity class c, and true tag different from t.

As the problem of finding a consistent hypothesis consisting of k clauses with the smallest k is NP-complete [14], we used a simple variant of *tabu search* [1] with tabu size 5 maximizing the *relative frequency* function n_{correct}/n, where n_{correct} is the number of training examples correctly classified, and n is the number of the covered training examples. A clause discovered was accepted if its accuracy on the covered training examples was at least 90%, and it covered at least 10% of the positive examples.

3.4 RIBL

RIBL is an instance-based ILP algorithm first introduced in [10] and further developed in [3]. Although there are a number of user-defined parameters controlling the automatic similarity measure used by RIBL, one of the basic practical problems is the choice of an adequate representation of the learning problem. In the case of the POS problem, we considered two different representations.

Representation with Lists In this *non-flattened* representation we used a ternary predicate for describing the background knowledge. Its first *input* argument was used to identify instances (i.e., tag occurrences), and its second, and third *output* arguments, both of type *list*, were used to describe the left and right neighbors (i.e., the true tags) of the instance identified by the first argument. In both directions we considered only *two* words at most (i.e., their true tags). For each ambiguity class $c = \{t_1, \ldots, t_k\}$ investigated we took the set X_c of training instances with ambiguity class c and for each $x \in X_c$ we added a target atom $t(x)$ to the set of training instances, where t is the true tag of the instance identified by x. In order to compare arguments of type list, we used list edit distance supported directly by RIBL [3]. We used trivial cost [3] on the list alphabet (i.e., the set of possible tags).

Representation with Relations In this representation we used a binary background relation R describing the chain of the words, i.e., the background knowledge contains a fact $R(a, b)$ if and only if the two instances identified by a and b, respectively, are consecutive in this order. For each training instance a with true tag t we also added the fact $\texttt{true\_tag}(a, t)$ to the background knowledge.

As in the previous representation, for each ambiguity class c we considered a separate learning problem with the same training instances as in the previous non-flattened representation.

4 Experimental Results and Discussion

In this section we discuss the experimental results of the five learning systems on the domain described in Subsection 2.2. Since AGLEARN used extra linguistic background knowledge and is based on C4.5, the results with C4.5 and AGLEARN are discussed in the same item.

C4.5 and AGLEARN C4.5 could not disambiguate only 15 test queries, in other words, it made a decision in 99.4% of the 2443 ambiguous test queries. It predicted 1973 instances correctly, which is a 80.8% test accuracy for the whole test set containing 2443 instances. We also note that this corresponds to a relative accuracy of 81.3% for the set of 2428 disambiguated test examples.

In the extended representation used by AGLEARN, the number of undecided cases was 72, i.e., it disambiguated 97.1% of the test instances. On the other hand, even for this smaller set, the number of correctly classified queries was 2045, i.e., 83.7% of the test examples. This is a 86.3% relative accuracy with respect to the decided cases.

Progol We got slightly different results with Progol. Progol could decide only in 1606 cases, that is, only in 65.7% of the test domain. A similar experimental result with Progol is also reported in [5]. On the other hand, Progol classified 1482 examples correctly, i.e., 60.7% of the 2443 test instances. This is an excellent 92.3% relative accuracy with respect to the decided 1606 cases.

PHM The rules discovered by PHM disambiguated 1965 cases from the 2443 ambiguous test queries, i.e., 80.4% of the 2443 test instances. 1776 cases, namely 72.7% of the 2443 examples, were correctly classified, which corresponds to a relative accuracy of 90.4% for the disambiguated 1965 cases.

RIBL In the non-flattened representation the optimal number of neighbors $k = 11$ was estimated by the *leave-one-out* evaluation method on the training set. Since the true tags in the window of a test instance may be unknown, we introduced a special symbol. The cost of inserting, deleting or replacing this symbol with any symbol was set to 1. In contrast to the other four systems, RIBL did not leave undecided cases, and correctly classified 84.8% (i.e., 2072 out of 2443) of the test set.

The best training (as well as test) accuracy was achieved by depth bound 3 in the second representation. In this case, RIBL correctly classified 1794 test instances from the 2443, which is a 73.4% test set accuracy. This result show that a significant improvement on the accuracy can be gained by using lists with edit distance on this domain (see also [3]).

In the remaining part of this section we summarize the above mentioned experimental results. For RIBL we take the result with the list representation. C4.5, AGLEARN, and RIBL have very good disambiguation capabilities, that is, the number of undecided test queries was small for these systems. In contrast to RIBL, C4.5, and AGLEARN, the systems Progol and PHM could not disambiguate a significant part of the test queries. There was also a difference between Progol and PHM in this sense. On the other hand, Progol and PHM had the best relative accuracy for the decided cases, and Progol was better than PHM. We also note, that although the extra linguistic background knowledge used by AGLEARN increased the number of undecided cases, it improved the results with C4.5.

In order to compare the previous results, we disambiguated the undecided test queries by applying the simple tagger based on lexicon frequencies. The accuracy of this tagger was 76.1% on the set containing the 2443 ambiguous test instances. The final results are listed in Table 2. The table also contains the per-word overall accuracies without punctuation symbols. In order to compute them, we used the lexicon frequencies for the uncovered 234 cases. The line 'ambiguous' in Table 2 illustrates the accuracies for the 2443 ambiguous test instances covered by the 27 classes investigated.

	C4.5	AGLEARN	Progol	PHM	RIBL (lists)
ambiguous (%)	81.0	84.8	82.8	84.6	84.8
overall per-word (%)	97.60	98.03	97.80	98.00	98.03

Table 2. Final test set accuracy results expressed in percentage terms.

4.1 Cascade Connection of the Taggers

As mentioned earlier, our goal was not to compare the different learning systems, but to select the candidate components of a working combined tagger. As a first step into this direction, we investigated some cascade connections of the previous taggers. These combinations first employ systems which are very accurate but have low coverage (e.g. Progol and/or PHM) and fall back to RIBL (which leaves no uncovered instances) only for the undecided tokens. The results of some combinations investigated are given in Table 3 (the accuracy is computed with respect to the 2443 ambiguous test instances considered).

	Progol $\rightarrow$ RIBL	PHM $\rightarrow$ RIBL	Progol $\rightarrow$ PHM $\rightarrow$ RIBL
accuracy (%)	85.7	86.5	86.0

Table 3. Results with different cascade connections of Progol, PHM, and RIBL.

Although the cascade connection is a simple combination in contrast e.g. [13], the results show the potential of this approach. An interesting observation is that the cascade connection of Progol with PHM did not improve the accuracy which means that PHM has weaker decision capability on the undecided tokens left by the Progol tagger.

5 Conclusion

In this paper, we have presented an initial case study with five selected ILP and non-ILP algorithms on the problem of POS tagging for the Hungarian language.

The purpose of the experiments was not to compare the learning systems, but rather to examine how the unusual and difficult properties of Hungarian (complex grammar, large tag set, free word order) affect the properties of the problem. Our experiments indicate that Hungarian is indeed a particular challenge to the learning of POS tagging rules, whether using propositional or ILP methods.

More specifically, however, some generally observed properties of the problem seem to carry over from simpler languages to Hungarian. First, in [13], it was reported that memory-based methods (MBL) are particularly suited for learning POS taggers. Given the accuracy figures reported above, our experiments indicate that this might also be the case for the POS tagging problem in Hungarian: the instance-based ILP system RIBL shows good results on this domain both in coverage and accuracy aspects. Secondly, as already emphasized e.g. by Cussens [5], the use of linguistic background knowledge is of great importance and cannot be offset by the use of more powerful learners. In our experiments, even a very simple set of linguistic background knowledge, provided by AGLEARN to C4.5, has notably improved the learning results. Thirdly, as in the experiments by Cussens for English [5], we found that also for the Hungarian POS tagging problem, both rule-inducing ILP learners have very good accuracy in those cases when they could decide, however the number of uncovered cases is relatively high for these learners, indicating some need for further generalization ability e.g. by giving additional background knowledge.

In future work on continuing this case study, we are therefore going to use additional linguistic background knowledge provided by an expert in order to improve accuracies of the three ILP systems Progol, PHM, and RIBL. For RIBL in particular, this may also mean using an expert-provided editing cost matrix instead of the trivial one used in the present experiments. In the cases of Progol and PHM, a test query remained undecided if it was covered by no clause or by two or more clauses predicting different tags. In order to reduce the number of undecided instances, we are going to apply Bayesian method for these cases. Bayesian methods for POS tagging have also been investigated in [6]. For the PHM approach, we plan improvements using *local search*. Finally, given the encouraging results of [13] on taggers that combine different learning systems, we are going to investigate other more sophisticated combinations as well.

Acknowledgements

This work was supported partly by the project ESPRIT IV 20237 ILP2. The second and third authors were also supported by the Hungarian OTKA grant T25721.

References

[1] E. H. L. Aarts and J. K. Lenstra, editors. *Local Search in Combinatorial Optimization.* Discrete Mathematics and Optimization. Wiley-Interscience, 1997.

[2] Z. Alexin, S. Zvada, and T. Gyimóthy. Application of AGLEARN on Hungarian Part-of-speech Tagging. In D. Parigot and M. Mernik, editors, *Second Workshop on Attribute Grammars and their Applications, WAGA'99*, pages 133–152, Amsterdam, The Netherlands, 1999. INRIA Rocquencourt.

[3] U. Bohnebeck, T. Horváth, and S. Wrobel. Term comparisons in first-order similarity measures. In D. Page, editor, *Proc. 8th Int. Conference on Inductive Logic Programming (ILP98)*, pages 65–79. Springer Verlag, 1998.

[4] E. Brill. Some advances in transformation-based part of speech tagging. In *Proceedings of the 12th National Conference on Artificial Intelligence. Volume 1*, pages 722–727. AAAI Press, July 31–Aug. 4 1994.

[5] J. Cussens. Part-of-speech tagging using Progol. In N. Lavrač and S. Džeroski, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297 of *LNAI*, pages 93–108. Springer, Sept. 17–20 1997.

[6] J. Cussens. Using prior probabilities and density estimation for relational classification. *Lecture Notes in Computer Science*, 1446:106–115, 1998.

[7] W. Daelemans, A. v. d. Bosch, and J. Zavrel. Rapid development of NLP modules with memory-based learning. In *Proc. of ELSNET in Wonderland, Utrecht*, pages 105–113, 1998.

[8] S. Dzeroski and N. Lavrac. Inductive learning in deductive databases. *IEEE Transactions on Knowledge and Data Engineering: Special Issue on Learning and Discovery in Knowledge-Based Databases*, 5(6):939–949, Dec. 1993.

[9] M. Eineborg and N. Lindberg. Induction of constraint grammar-rules using Progol. *Lecture Notes in Computer Science*, 1446:116–124, 1998.

[10] W. Emde and D. Wettschereck. Relational instance based learning. In L. Saitta, editor, *Machine Learning - Proceedings 13th International Conference on Machine Learning*, pages 122–130. Morgan Kaufmann Publishers, 1996.

[11] T. Erjavec, A. Lawson, and L. R. (eds.). East meets west: A compendium of multilingual resources, 1998. CD-ROM, produced and distributed by TELRI Association e.V., ISBN:3-922641-46-6.

[12] T. Gyimóthy and T. Horváth. Learning semantic functions of attribute grammars. *Nordic Journal of Computing*, 4(3):287–302, Fall 1997.

[13] H. v. Halteren, J. Zavrel, and W. Daelemans. Improving data driven wordclass tagging by system combination. In *Proc. of COLING-ACL'98, Montreal, Canada*, pages 491–497, 1998.

[14] T. Horváth. *Learning logic programs with structured background knowledge*. PhD thesis, German National Research Center for Information Technology, 1999.

[15] T. Horváth, R. H. Sloan, and G. Turán. Learning logic programs by using the product homomorphism method. In *Proceedings of the 10th Annual Conference on Computational Learning Theory*, pages 10–20. ACM Press, July 6–9 1997.

[16] T. Horváth and G. Turán. Learning logic programs with structured background knowledge. In L. D. Raedt, editor, *Advances in Inductive Logic Programming*, pages 172–191. IOS Press, 1996.

[17] B. Megyesi. Brill's rule based part-of-speech tagger for Hungarian. Master's thesis, University of Stockholm, 1998.

[18] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[19] C. Oravecz. Morfoszintaktikai annotáció a magyar nemzeti szövegtárban. Technical report, Research Institute for Linguistics, Hungarian Academy of Sciences, 1998. (in Hungarian).

[20] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.