Experiments with Learning Parsing Heuristics

Sylvain DELISLE
Département de mathématiques et d'informatique
Université du Québec à Trois-Rivières
Trois-Rivières, Québec, Canada, G9A 5H7
Sylvain_Delisle@uqtr.uquebec.ca

Sylvain LÉTOURNEAU, Stan MATWIN
School of Information Technology and
Engineering, University of Ottawa
Ottawa, Ontario, Canada, K1N 6N5
sletour@ai.iit.nrc.ca , stan@site.uottawa.ca

Abstract

Any large language processing software relies in its operation on heuristic decisions concerning the strategy of processing. These decisions are usually "hard-wired" into the software in the form of hand-crafted heuristic rules, independent of the nature of the processed texts. We propose an alternative, adaptive approach in which machine learning techniques learn the rules from examples of sentences in each class. We have experimented with a variety of learning techniques on a representative instance of this problem within the realm of parsing. Our approach lead to the discovery of new heuristics that perform significantly better than the current hand-crafted heuristic. We discuss the entire cycle of application of machine learning and suggest a methodology for the use of machine learning as a technique for the adaptive optimisation of language-processing software.

1 Introduction

Any language processing program—in our case, a top-down parser which outputs only the first tree it could find—must make decisions as to what processing strategy, or rule ordering, is most appropriate for the problem (i.e. string) at hand. Given the size and the intricacy of the rule-base and the goal (to optimise a parser's precision, or recall, or even its speed), this becomes a complex decision problem. Without precise knowledge of the kinds of texts that will be processed, these decisions can at best be educated guesses. In the parser we used, they were performed with the help of hand-crafted heuristic rules, which are briefly presented in section 2.

Even when the texts are available to fine-tune the parser, it is not obvious how these decisions are to be made from texts alone. Indeed, the decisions may often be expressed as rules whose representation is in terms which are not directly or easily available from the text (e.g. non-terminals of the grammar of the language in which the texts are written). Hence, any technique that may automatically or semi-automatically adapt such rules to the corpus at hand will be valuable. As it is often the case, there may be a linguistic shift in the kinds of texts that are processed, especially if the linguistic task is as general as parsing. It is then interesting to adapt the "version" of the parser to the corpus at hand.

We report on an experiment that targets this kind of adaptability. We use machine learning as an artificial intelligence technique that achieves adaptability. We cast the task described above as a classification task: *which, among the parser's top-level rules, is most appropriate to launch the parsing of the current input string?* Although we restricted ourselves to a subset of a parser, our objective is broader than just applying an existing learning system on this problem. What is interesting is: a) definition of the attributes in which examples are given, so that the attributes are both obtainable automatically from the text and lead to good rules—this is called "feature engineering"; b) selection of the most interesting learned rules; c) incorporation of the learned rules in the parser; d) evaluation of the performance of the learned rules after they have been incorporated in the parser. It is the lessons from the whole cycle that we followed in the work that we report here, and we suggest it as a methodology for an adaptive optimisation of language processing programs.

2 The existing hand-crafted heuristics

The rule-based parser we used was DIPETT [Delisle 1994]: it is a top-down, depth-first parser, augmented with a few look-ahead mechanisms, which returns the first analysis (parse tree). The fact that our parser produces only a single analysis, the "best" one according to its hand-crafted heuristics, is part of the motivation for this work. When DIPETT is given an input string, it first selects the top-level rules it is to attempt, as well as their ordering in this process. Ideally, the parser would find an optimal order that minimises parsing time and maximises parsing accuracy by first selecting the most promising rules. For example, there is no need to treat a sentence as multiply coordinated or compound when the data contains only one verb. DIPETT has three top-level rules for declarative statements: i) MULT_COOR for multiple (normally, three or more) coordinated sentences; ii) COMPOUND for compound sentences, that is, correlative and simple coordination (of, normally, two sentences); iii) NON_COMPOUND for simple and complex sentences, that is, a single main clause with zero or more subordinate clauses ([Quirk *et al.* 1985]). To illustrate the data that we worked with and the classes for which we needed the rules, here are two sentences (from the Brown corpus) used in our experiments: "And know, while all this went on, that there was no real reason to suppose that the murderer had been a guest in either hotel." is a non-compound sentence, and "Even I can remember nothing but ruined cellars and tumbled pillars, and nobody has lived there in the memory of any living man." is a compound sentence.

The current hand-crafted heuristic ([Delisle 1994]) is based on three parameters, obtained after (non-disambiguating) lexical analysis and before parsing: 1) the number of potential verbs[1] in the data, 2) the presence of potential coordinators in the data, and 3) verb density (roughly speaking, it indicates how potential verbs are distributed). For instance, low density means that verbs are scattered throughout the input string; high density means that the verbs appear close to each other in the input string, as in a conjunction of verbs such as "$Verb_1$ and $Verb_2$ and $Verb_3$". Given the input string's features we have just discussed, DIPETT's algorithm for top-level rule selection returns an ordered list of up to 3 of the rules COMPOUND, NON_COMPOUND, and MULT_COOR to be attempted when parsing this string. For the purposes of our experiment, we simplified the situation by neglecting the MULT_COOR rule since it was rarely needed when parsing real-life text. Thus, the original problem went from a 3-class to a 2-class classification problem: COMPOUND or NON_COMPOUND.

[1] A "potential" verb may actually turn out to be, say, a noun, but only parsing can tell us how such a lexical ambiguity has been resolved. If the input were pre-processed by a tagger, the ambiguity might disappear.

3 Learning rules from sentences

As any heuristic, the top-level rule selection mechanism just described is not perfect. Among the principal difficulties, the most important are: i) the accuracy of the heuristic is limited and ii) the internal choices are relatively complex and somewhat obscure from a linguist's viewpoint. The aim of this research was to use classification systems as a tool to help developing *new* knowledge for improving the parsing process. To preserve the broad applicability of DIPETT, we have emphasised the generality of the results and did not use any kind of domain knowledge. The sentences used to build the classifiers and evaluate the performance have been randomly selected from five unrelated real corpora.

Typical classification systems (e.g. decision trees, neural networks, instance based learning) require the data to be represented by feature vectors. Developing such a representation for the task considered here is difficult. Since the top-level rule selection heuristic is one of the first steps in the parsing process, very little information for making this decision is available at the early stage of parsing. All the information available at this phase is provided by the (non-disambiguating) lexical analysis that is performed before parsing. This preliminary analysis provides four features: 1) number of potential verbs in the sentence, 2) presence of potential coordinators, 3) verb density, and 4) number of potential auxiliaries. As mentioned above, only the first three features are actually used by the current hand-crafted heuristic. However, preliminary experiments have shown that no interesting knowledge can be inferred by using only these four features. We then decided to improve our representation by the use of DIPETT's

fragmentary parser: an optional parsing mode in which DIPETT does not attempt to produce a single structure for the current input string but, rather, analyses a string as a sequence of major constituents (i.e. noun, verb, prepositional and adverbial phrases). The new features obtained from fragmentary parsing are: the number of fragments, the number of "verbal" fragments (fragments that contain at least one verb), number of tokens skipped, and the total percentage of the input recognised by the fragmentary parser. The fragmentary parser is a cost-effective solution to obtain a better representation of sentences because it is very fast—on average, less than one second of CPU time for any sentence—in comparison to full parsing.

Moreover, the information obtained from the fragmentary parser is adequate for the task at hand because it represents well the complexity of the sentence to be parsed. In addition to the features obtained from the lexical analysis and those obtained from the fragmentary parser, we use the string length (number of tokens in the sentence) to describe each sentence. The attribute used to classify the sentences, provided by a human expert, is called **rule-to-attempt** and it can take two values: compound or non-compound, according to the type of the sentence. To summarise, we used the ten following features to represent each sentence: 1) **string-length**: number of tokens (integer); 2) **num-potential-verbs:** number of potential verbs (integer); 3) **num-potential-auxiliary:** number of potential auxiliaries (integer); 4) **verb-density:** a flag that indicates if all potential verbs are separated by coordinators (boolean); 5) **nbr-potential-coordinators:** number of potential coordinators (integer); 6) **num-fragments:** number of fragments used by the fragmentary parser (integer); 7) **num-verbal-fragments:** number of fragments that contain at least one potential verb (integer); 8) **num-tokens-skip:** number of tokens not considered by the fragmentary parser (integer); 9) **%-input-recognized:** percentage of the sentence recognized, i.e. not skipped (real); 10) **rule-to-attempt**: type of the sentence (COMPOUND or NON-COMPOUND).

We built the first data set by randomly selecting 300 sentences from four real texts: a software user manual, a tax guide, a junior science textbook on weather phenomena, and the Brown corpus. Each sentence was described in terms of the above features, which are of course acquired automatically by the lexical analyser and the fragmentary parser, except for **rule-to-attempt** as mentioned above. After a preliminary analysis of these 300 sentences, we realised that we had unbalanced numbers of examples of compound and non-compound sentences: non-compounds are approximately five times more frequent than compounds. However, it is a well-known fact in machine learning that such unbalanced training sets are not suitable for inductive learning. For this reason, we have re-sampled our texts to obtain roughly an equal number of non-compound and compound sentences (55 compounds and 56 non-compounds).

Our experiment consisted in running a variety of attribute classification systems: IMAFO ([Famili & Turney 1991]), C4.5 ([Quinlan 1993]), and different learning algorithms from MLC++ ([Kohavi *et al.* 1994]). IMAFO includes an enhanced version of ID3 and an interface to C4.5 (we used both engines in our experimentation). MLC++ is a machine learning library developed in C++. We experimented with many algorithms included in MLC++.

We concentrated mainly on learning algorithms that generate results in the form of rules. For this project, rules are more interesting than other form of results because they are relatively easy to integrate in a rule-based parser and because they can be evaluated by experts in the domain. However, for accuracy comparison, we have also used learning systems that do not generate rules in terms of the initial representation: neural networks and instance-based systems. We randomly divided our data set into the training set (2/3 of the examples, or 74 instances) and the testing set (1/3 of the examples, or 37 instances). Table 1 summarises the results obtained from different systems in terms of the error rates on the testing set. All systems gave results with an error rate below 20%.

SYSTEM	Type of system	Error rate
ID3	decision rules	16.2%
C4.5	decision rules	18.9%
IMAFO	decision rules	16.5%
oneR	decision rule (one)	15.6%
IB	instance-based	10.8%
aha-ib	instance-based	18.9%
naive-bayes	belief networks	16.2%
perceptron	neural networks	13.5%

Table 1. Global results from learning.

The error rates presented in Table 1 for the first four systems (decision rules systems) represent the average rates for all rules generated by these systems. However, not all rules were particularly interesting. We kept only some of them for further evaluation and integration in the parser. Our selection criteria were: 1) the estimated error rate, 2) the "reasonability" (only rules that made sense for a computational linguist were kept), 3) the readability (simple rules are preferred), and 4) the novelty (we discarded rules that are already in the parser). Tables 2 and 3 present rules that satisfy all the above the criteria: Table 2 focuses on rules to identify compound sentences while Table 3 presents rules to identify non-compound sentences. The error rate for each rule is also given. These error rates were obtained by a 10 fold cross-validation test.

Rules to identify COMPOUND sentences	Error rate (%)
num-potential-verbs <= 3 AND num-potential-coordinators > 0 AND num-verbal-fragments > 1	10.5
num-fragments > 7	9.4
num-fragments > 5 AND num-verbal-fragments <= 2	23.9
string-length <= 17 AND num-potential-coordinators > 0 AND num-verbal-fragments > 1	5.4
num-potential-verbs > 1 AND num-potential-verbs <= 3 AND num-potential-coordinators > 0 AND num-fragments > 4	4.2
num-potential-coordinators > 0 AND num-fragments >= 7	4.3
num-potential-coordinators > 0 AND num-verbal-fragments > 1	16.8
num-potential-coordinators > 0 AND num-fragments < 7 AND string-length < 18	4.7

Table 2. Rules to identify COMPOUND sentences

The error rates that we have obtained are quite respectable for a two-class learning problem given the volume of available examples. Moreover, the rules are justified and make sense. They are also very compact in comparison with the original hand-crafted heuristics. We will see in section 4 how these rules behave on unseen data from a totally different text.

Rules to identify NON-COMPOUND sentences	Error rate (%)
num-potential-verbs <= 3 AND num-verbal-fragments <= 1	8.3
string-length > 10 AND num-potential-verbs <= 3 AND num-fragments <= 4	6.7
string-length <= 21 AND num-potential-coordinators = 0	5.6
num-potential-coordinators = 0 AND num-fragments <= 7	9.7

Table 3. Rules to identify NON-COMPOUND sentences

Attribute classification systems such as those used during the experiment reported here are highly sensitive to the adequacy of the features used to represent the instances. For our task (parsing), these features were difficult to find and we had only a rough idea about their appropriateness. For this reason, we felt that better results could be obtained by transforming the original instance space into a more adequate space by creating new attributes. In machine learning research, this process is referred as constructive learning, or constructive induction ([Wnek & Michalski 1994]). We even attempted to use principal component analysis (PCA) ([Johnson & Wichern 1992]) as a technique of choice for simple constructive learning but we did not get very impressive results. We see two reasons for this. The primary reason is that the ratio between the number of examples and the number of attributes is not high enough for PCA to derive high-quality new attributes. The second reason is that the original attributes are already highly non-redundant. It is important to note that these rules do not satisfy the reasonability criteria applied to the original representation. In fact, losing the understandability of the attributes is the usual consequence of almost all approaches that change the representation of instances.

4 Evaluation of the new rules

We explained in section 3 how we derived new parsing heuristics with the help of machine learning techniques. The next step was to evaluate how well would the new rules perform if we replaced the parser's current hand-crafted heuristics with the new ones. In particular, we wanted to evaluate the accuracy of the heuristics in correctly identifying the appropriate rule, COMPOUND or NON_COMPOUND, that should first be attempted by

the parser. This goal was prompted by an earlier evaluation of DIPETT in which it was noted that a good proportion of questionable parses (i.e. either bad parses or correct but too time-consuming parses) were caused by a bad first attempt, such as attempting COMPOUND instead of NON_COMPOUND.

4.1 From new rules to new parsers

Our machine learning experiments lead us to two classes of rules obtained from a variety of classifiers and concerned only with the notion of compoundness: 1) those predicting a COMPOUND sentence, and 2) those predicting a NON_COMPOUND. The problem was then to decide what should be done with the set of new rules. More precisely, before actually implementing the new rules and including them in the parser, we first had to decide on an appropriate strategy for exploiting the set of new rules. We now describe the three implementations that we realised and evaluated.

The first implements only the rules for the COMPOUND class—one big rule which is a disjunct of all the learned rules for that class. And since there are only two alternatives, either COMPOUND or NON_COMPOUND, if none of the COMPOUND rules applies, the NON_COMPOUND class is predicted. This first implementation is referred to as C-Imp. The second implementation, referred to as NC-Imp, does exactly the opposite: i.e. it implements only the rules predicting the NON_COMPOUND class.

The third implementation, referred to as NC_C-Imp, benefits from the first two implementations. The class of a new sentence is determined by combining the output from C-Imp and NC-Imp. The combination of the output is done according to the following decision table in Table 4.

C-Imp	NC-Imp	Output of NC_C-Imp
C	C	C
NC	NC	NC
NC	C	NC
C	NC	NC

Table 4. Decision table used in the NC_C implementation.

The first two lines of this decision table are obvious since the outputs from both implementations are consistent. When the two implementations disagree, the NC_C-Imp implementation predicts the non-compound. This prediction is justified by a bayesian argumentation. In the absence of any additional knowledge, we are forced to assign an equal probability of success to each of the two sets of rules and the most probable class becomes the one with the highest frequency. Thus, in general, non-compound sentences are more frequent then compound ones. One obvious way to improve this third implementation would be to precisely evaluate the accuracies of the two sets of rules and then incorporate these accuracies in the decision process.

4.2 The results

To perform the evaluation, we randomly sampled 200 sentences from a new corpus on mechanics ([Atkinson 1990]): *note that this text had not been used to sample the sentences used for learning*. Out of these 200 sentences, 10 were discarded since they were not representative (e.g. one-word "sentences"). We ran the original implementation of DIPETT plus the three new implementations described in the previous section on the remaining 190 test sentences. Table 5 presents the results. The error-rate, the standard deviation of the error-rate and the p-value are listed for each implementation. The p-value gives the probability that DIPETT's original hand-crafted heuristics are better than the new heuristics. In other words, a small p-value means an increase in performance with a high probability.

Implementation	Err-rate (%)	Std. dev.	p-value
Original heur.	25.268	±3.2	---
C-Imp	20.526	±2.9	0.126
NC-Imp	22.105	±3.0	0.229
NC_C-Imp	16.316	±2.7	0.009

Table 5. Performances of the new implementations versus DIPETT's original heuristics.

We observe that all new automatically-derived heuristics did beat DIPETT's hand-crafted heuristics and quite clearly. The results from the third implementation (i.e. NC_C-Imp) are especially remarkable: *with a confidence of over 99%, we can*

affirm that the NC_C-Implementation will outperform DIPETT's original heuristic. We also note that the error rate drops by 35% of its value for the original heuristic. Similarly, with a confidence of 87.4%, we can affirm that the implementation that uses only the C-rules (i.e. C-Imp) will perform better then DIPETT's current heuristics.

These very good results are also amplified by the fact that *the testing described in this evaluation was done on sentences totally independent from the ones used for training.* Usually, in machine learning research, the training and the testing sets are sampled from the same original data set, and the kind of "out-of-sample" testing that we perform here has only recently come to the attention of the learning community ([Ezawa *et al.* 1996]). Our experiments have shown that it is possible to infer rules that perform very well and are highly meaningful in the eyes of an expert even if the training set is relatively small. This indicates that the representation of sentences that we chose for the problem was adequate. Finally, an other important output of our research is the identification of the most significant attributes to distinguish non-compound sentences from compound ones. This alone is valuable information to a computational linguist. Only five out of ten original attributes are used by the learned rules, and all of them are cheap to compute: two attributes are derived by fragmentary parsing (number of verbal fragments and number of fragments), and three are lexical (number of potential verbs, length of the input string, and presence of potential coordinators).

5 Related Work

There have been successful attempts at using machine learning in search of a solution for linguistic tasks, e.g. discriminating between discourse and sentential senses of cues ([Litman 1996]) or resolution of coreferences in texts ([McCarthy & Lehnert 1995]). Like our work, these problems are cast as classification problems, and then machine learning (mainly C4.5) techniques are used to induce classifiers for each class. What makes these applications different from ours is that they have worked on surface linguistic or mixed surface linguistic and intonational representation, and that the classes are relatively balanced, while in our case the class of compound sentences is much less numerous than the class of non-composite sentences. Such unbalanced classes create problems for the majority of inductive learning systems.

A distinctive feature of our work is the fact that we used machine learning techniques to improve an existing rule-based natural language processor *from the inside.* This contrasts with approaches where there are essentially no explicit rules, such as neural networks (e.g. [Buo 1996]), or approaches where the machine learning algorithms attempt to infer—via deduction (e.g. [Samuelsson 1994]), induction (e.g. [Theeramunkong *et al.* 1997]; [Zelle & Mooney 1994]) under user cooperation (e.g. [Simmons & Yu 1992]; [Hermjakob & Mooney 1997]), transformation-based error-driven learning (e.g. [Brill 1993]), or even decision trees (e.g. [Magerman 1995])—a grammar from raw or preprocessed data. In our work, we do not wish to acquire a grammar: we have one and want to devise a mechanism to make some of its parts adaptable to the corpus at hand or, to improve some aspect of its performance. Other researchers, such as [Lawrence *et al.* 1996], have compared neural networks and machine learning methods at the task of sentence classification. In this task, the system must classify a string as either grammatical or not. We do not content ourselves with results based on a grammatical/ungrammatical dichotomy. We are looking for heuristics, using relevant features, that will do better than the current ones and improve the overall performance of a natural language processor: this is a very difficult problem (see, e.g., [Huyck & Lytinen 1993]). One could also look at this problem as one of optimisation of a rule-based system.

Work somewhat related to ours was conducted by [Samuelsson 1994] who used explanation-based generalisation to extract a subset of a grammar that would parse a given corpus faster than the original, larger grammar—[Neumann 1997] also used EBL but for a generation task. In our case, we are not looking for a subset of the existing rules but, rather, we are looking for brand new rules that would replace and outperform the existing rules. We should also mention the work of [Soderland 1997] who also worked on the comparison of automatically learned and hand-crafted rules for text analysis.

6 Conclusion

We have presented an experiment which demonstrates that machine learning may be used as a technique to optimise in an adaptive manner the high-level decisions that any parser must make in the presence of incomplete information about the properties of the text it analyses. The results show clearly that simple and understandable rules learned by machine learning techniques can surpass the performance of heuristics supplied by an experienced computational linguist. Moreover, these very encouraging results indicate that the representation that we chose and discuss was an adequate one for this problem. We feel that a methodology is at hand to extend and deepen this approach to language processing programs in general. The methodology consists of three main steps: 1) feature engineering, 2) learning, using several different available learners, 3) evaluation, with the recommendation of using the "out-of-sample" approach to testing. Future work will focus on improvements to constructive learning; on new ways of integrating the rules acquired by different learners in the parser; and on the identification of criteria for selecting parser rules that have the best potential to benefit from the generalisation of our results.

Acknowledgements

The work described here was supported by the Natural Sciences and Engineering Research Council of Canada.

References

Atkinson, H.F. (1990) *Mechanics of Small Engines*. New York: Gregg Division, McGraw-Hill.

Brill E. (1993) "Automatic Grammar Induction and Parsing Free Text: A Transformation-Based Approach", *Proc. of the 31st Annual Meeting of the ACL*, pp.259-265.

Buo F.D. (1996) "FeasPar—A Feature Structure Parser Learning to Parse Spontaneous Speech", Ph.D. Thesis, Fakultät für Informatik, Univ. Karlsruhe, Germany.

Delisle S. (1994) "Text Processing without a priori Domain Knowledge: Semi-Automatic Linguistic for Incremental Knowledge Acquisition", Ph.D. Thesis, Dept. of Computer Science, Univ. of Ottawa. Published as technical report TR-94-02.

Ezawa K., Singh M. & Norton S. (1996) "Learning Goal Oriented Bayesian Networks for Telecommunications Risk Management", *Proc. of the 13th International Conf. on Machine Learning*, pp.139-147.

Famili A. & Turney P. (1991) "Intelligently Helping the Human Planner in Industrial Process Planing", *AI EDAM - AI for Engineering Design Analysis and Manufacturing*, 5(2), pp.109-124.

Hermjakob U. & Mooney R.J. (1997) "Learning Parse and Translation Decisions From Examples With Rich Context", *Proc. of ACL-EACL Conf.*, pp.482-489.

Huyck C.R. & Lytinen S.L. (1993) "Efficient Heuristic Natural Language Parsing", *Proc. of the 11th National Conf. on AI*, pp.386-391.

Johnson R.A. & Wichern D.W. (1992) *Applied Multivariate Statistical Analysis*, Prentice Hall.

Kohavi R., John G., Long R., Manley D. & Pleger K. (1994) "MLC++: A machine learning library in C++", *Tools with AI*, IEEE Computer Society Press, pp.740-743.

Lawrence S., Fong S. & Lee Giles C. (1996) "Natural Language Grammatical Inference: A Comparison of Recurrent Neural Networks and Machine Learning Methods", in S. Wermter, E. Riloff and G. Scheler (eds.), *Symbolic, Connectionnist, and Statistical Approaches to Learning for Natural Language Processing*, Lectures Notes in AI, Springer-Verlag, pp.33-47.

Litman D. (1996) "Cue Phrase Classification Using Machine Learning", *Journal of AI Research*, 5, pp.53-95.

Magerman D. (1995) "Statistical Decision-Tree Models for Parsing", *Proc. of the 33rd Annual Meeting of the ACL*, 276-283.

McCarthy J. & Lehnert W.G. (1995) "Using Decision Trees for Coreference Resolution", *Proc. of IJCAI-95*, pp.1050-1055.

Neumann G. (1997) "Applying Explanation-based Learning to Control and Speeding-up Natural Language Generation", *Proc. of ACL-EACL Conf.*, pp.214-221.

Quinlan J.R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Quirk R., Greenbaum S., Leech G. & Svartvik J. (1985) *A Comprehensive Grammar of the English Language*, Longman.

Samuelsson C. (1994) "Grammar Specialization Through Entropy Thresholds", *Proc. of the 32nd Annual Meeting of the ACL*, pp.188-195.

Simmons F.S. & Yu Y.H. (1992) "The Acquisition and Use of Context-dependent Grammars for English", *Computational Linguistics*, 18(4), pp.392-418.

Soderland S.G. (1997) "Learning Text Analysis Rules for Domain-Specific Natural Language Processing", Ph.D. Thesis, Dept. of Computer Science, Univ. of Massachusetts.

Theeramunkong T., Kawaguchi Y. & Okumura (1997) "Exploiting Contextual Information in Hypothesis Selection for Grammar Refinement", *Proc. of the CEGDLE Workshop at ACL-EACL'97*, pp.78-83.

Wnek J. & Michalski R.S. (1994) "Hypothesis-driven constructive induction in AQ17-HCI: a method and experiments", *Machine Learning*, 14(2), pp.139-168.

Zelle J.M. & Mooney R.J. (1994) "Inducing Deterministic Prolog Parsers from Treebanks: A Machine Learning Approach", *Proc. of the 12th National Conf. on AI*, pp.748-753.