In Wermter, S. and Riloff, E. and Scheler, Gabriele (eds.), *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, Lecture Notes in Artificial Intelligence, 315-328, Springer, 1996.

Embedded Machine Learning Systems for Natural Language Processing: A General Framework

Claire Cardie

Department of Computer Science, Cornell University, Ithaca NY 14853, USA

Abstract. This paper presents Kenmore, a general framework for knowledge acquisition for natural language processing (NLP) systems. To ease the acquisition of knowledge in new domains, Kenmore exploits an on-line corpus using robust sentence analysis and embedded symbolic machine learning techniques while requiring only minimal human intervention. By treating all problems in ambiguity resolution as classification tasks, the framework uniformly addresses a range of subproblems in sentence analysis, each of which traditionally had required a separate computational mechanism. In a series of experiments, we demonstrate the successful use of Kenmore for learning solutions to several problems in lexical and structural ambiguity resolution. We argue that the learning and knowledge acquisition components should be *embedded* components of the NLP system in that (1) learning should take place within the larger natural language understanding system as it processes text, and (2) the learning components should be evaluated in the context of practical language-processing tasks.

1 Introduction

Although current natural language processing (NLP) systems cannot yet perform in-depth text understanding, they *can* read an arbitrary text and summarize its major events provided that those events fall within a particular domain of interest (e.g., stories about natural disasters or terrorist events) [11, 17]. Thus far, among the best-performing and most robust language-processing systems for this type of limited summarization task have been knowledge-based natural language systems — NLP systems that rely heavily on domain-specific, handcrafted knowledge to handle the myriad syntactic, semantic, and pragmatic ambiguities that pervade virtually all aspects of text understanding. Not surprisingly, however, generating this knowledge for new domains is time-consuming, difficult, and error-prone, and requires the expertise of computational linguists familiar with the underlying NLP system. In addition, handcoded lexicons and knowledge bases are generally too rigid to handle the novel language and range of phenomena that occur in real-world text. This *knowledge engineering bottleneck* is one of the biggest problems in designing and building natural language systems and promises only to become worse as natural language systems attempt to understand a wider variety of texts, to produce more complex summaries of the text, and to derive knowledge structures directly from text. On the other hand, much of human knowledge is described in written documents and, as mentioned above, NLP systems can now perform limited understanding of relatively complicated texts. Machine learning techniques for inductive learning have also become increasingly available and offer powerful mechanisms for simplifying the knowledge acquisition process.

As a result of these advances, we believe that it is feasible for NLP systems to begin to bootstrap their own knowledge bases using embedded machine learning components. This paper presents and evaluates a framework within which this bootstrapping process can occur and argues that learning for NLP systems should occur within the NLP system itself as it processes text — the natural language learning components should be directly incorporated into the larger natural language understanding system and should be evaluated in the context of practical language-processing tasks.

The paper is organized as follows. In the next section, we describe the problems encountered by an NLP system as it analyzes a text and describe how standard inductive machine learning algorithms can be used in a general way to help with the associated knowledge acquisition tasks. We then present Kenmore[1], a general framework for the machine learning of natural language and summarize the results of experiments that evaluate the architecture using real-world text in a real-world text-processing application. The experiments demonstrate that the framework offers a promising approach to knowledge acquisition for natural language processing systems. In particular, we show that symbolic inductive machine learning techniques can be used to learn solutions to significant problems in sentence analysis and that these learning mechanisms can be embedded in a working NLP system that has demonstrated success in processing real-world text. The paper concludes with a discussion of the advantages of embedded learning systems for natural language processing and the advances required to make existing machine learning algorithms better equipped for natural language learning tasks.

2 Ambiguity Resolution in Language Understanding

Many of the most difficult problems in natural language understanding can be characterized as problems of *ambiguity resolution*: the NLP system reaches a word or phrase for which it has multiple interpretations, and must decide which interpretation is appropriate in the current context. As an example, consider the following sentence:

BMW plans to build a plant for large machine tools in Eisenach.

First, the sentence contains a number of *part-of-speech ambiguities* that demand adequate resolution before the sentence can be understood. "Plans," for example, can be a noun or a verb; "to" can be an infinitive, a preposition, or an

[1] **ken** (ken), *vi.* 1 [Scot.] to know.

adverb; "plant" can be a noun or a verb. Even if a word's part of speech is known, the intended meaning of the word in a particular context often requires disambiguation. "Plant," for example, exhibits this sort of *word sense ambiguity.* At a minimum, the noun form of "plant" can be used in its "factory" sense or in its "living organism" sense. In addition to these cases of *lexical* ambiguity, disambiguation is required at the constituent, or *structural,* level as well. Consider the prepositional phrase "in Eisenach." Does it modify "tools," "plant," or "build"? All three options are syntactically legal and the NLP system must access semantic knowledge of some sort to make the correct attachment decision. Other structural ambiguities encountered in sentence analysis include: pronoun resolution, compound noun analysis, adjective scoping, and understanding conjunctions and appositives.

Typically, NLP systems treat each of these ambiguities separately along a number of dimensions. First, each type of ambiguity is viewed as a different problem. As a result, each is handled by a separate module of the system. There is often a part-of-speech tagger, for example, that is solely responsible for handling part-of-speech ambiguities. In addition, each ambiguity resolution module has limited access to the system's knowledge. The part-of-speech tagger, for example, may only have access to part-of-speech information for each word in the sentence. Finally, the NLP system may handle each ambiguity using a different computational mechanism (e.g., rule-based processing, context free parsers, regular expression recognizers, Markov models, constraint satisfaction, relaxation networks), each one of which may require access to diverse types of background knowledge. Until fairly recently, for example, a common approach to the problem of ambiguity resolution was to manually design separate sets of disambiguation heuristics to handle each type of ambiguity and to manually create the lexicons on which the heuristics rely.

We take a different approach in the Kenmore framework and, instead, use symbolic machine learning algorithms to support the acquisition and application of heuristics for semantic and syntactic ambiguity resolution at both the lexical and structural levels. The solution depends critically on the following assumption:

All problems of ambiguity resolution
can be recast as classification tasks.

By viewing disambiguation as a classification task, we can apply any of a number of inductive machine learning techniques to *learn* solutions to problems in ambiguity resolution rather than handcrafting a set of disambiguation rules: we expect the inductive learning algorithms to capture automatically those regularities in the use of language that permit the resolution of ambiguity in sentence analysis. As a result of this uniform view of ambiguity, lexical, structural, syntactic, and semantic ambiguities can be handled within a single architecture. As described below, solutions to these problems can be acquired in the Kenmore framework with minimal human intervention, thus avoiding the knowledge acquisition bottleneck in natural language system design. The next section describes the framework in more detail.

3 The Kenmore Framework

The Kenmore framework for the machine learning of natural language relies on three major components: a **corpus of texts**; a **robust sentence analyzer**, or parser; and an **inductive learning module**. There are two phases to the framework: (1) a partially automated training phase, or **acquisition phase**, in which the solution to a particular problem in sentence analysis is learned, and (2) an **application phase**, in which the learned solution can be applied in novel situations.

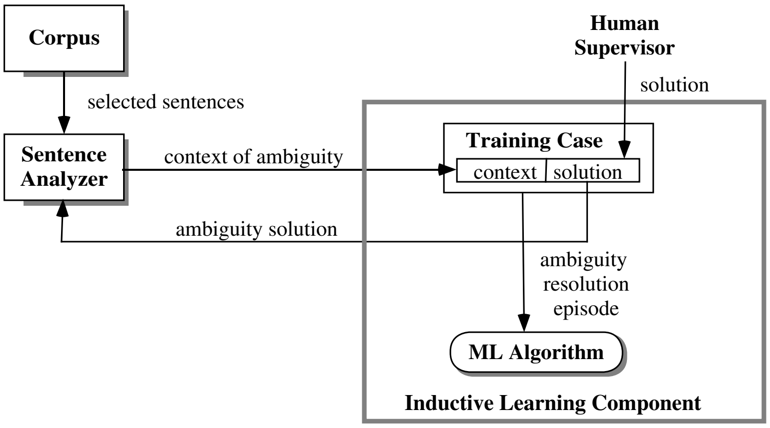

Fig. 1. Kenmore training/acquisition phase. Note that the human supervisor can be replaced by an annotated corpus.

More specifically, the goal of Kenmore's training phase (Fig. 1) is to create a set of ambiguity resolution episodes to be used as training examples for the inductive machine learning (ML) subsystem. To do this, a small set of sentences is first selected randomly from the corpus. Next, the sentence analyzer processes the training sentences and, with a human supervisor or annotated corpus, creates a training case every time an ambiguity occurs. To learn a set of heuristics to handle prepositional phrase attachment, for example, the parser would create a case whenever it recognizes a prepositional phrase. As shown in Fig. 1, each case has two parts. The *context* portion of the case encodes the context in which the ambiguity was encountered — this is an attribute-value pair representation of the state of the parser at the point of the ambiguity. The context part of the case is supplied automatically by the sentence analyzer in both the training and application phases. The *solution* portion of the case describes how the ambiguity was resolved in the current example — it encodes one or more pieces of "class" information. In the training phase, a person supplies this part of the case using a menu-driven interface. Alternatively, the necessary supervisory information can be extracted automatically from an appropriately annotated corpus. For part-of-speech disambiguation, for example, the human supervisor chooses the part of speech that is appropriate for the current word in the current context and this information is encoded as the class value for the training case. Together, the context and solution portions of the case represent a single ambiguity resolution episode. (See [8] for a general discussion of how to convert both lexical and structural ambiguity problems into classification tasks.)

To give a more concrete idea of what training cases look like, we show a portion of a sample training case for part-of-speech and word-sense disambiguation in Fig. 2. In this example, Kenmore was instantiated with the CIRCUS conceptual sentence analyzer [15] as its parser and training sentences were taken from the TIPSTER business joint ventures (JV) corpus.[2] The training case of Fig. 2 contains 15 context features and two solution features and was generated in response to the word "parts" in the sentence:

Daihatsu...has so far been in alliance with Astra Motor in production of auto **parts** *and engines...*

Once a training case is created, it is made available to the machine learning component, which uses the case to update the concept description it derives for the current ambiguity task. In addition, the solution to the ambiguity is forwarded to the parser so that it can resolve the current ambiguity, update its state, and continue processing the training sentence. At the end of the training phase, Kenmore will have created one training case for every instance of the ambiguity that appears in the training sentences.

After training, Kenmore resolves ambiguities in novel sentences from the corpus *without human intervention* (Fig. 3). Whenever the sentence analyzer encounters an ambiguity, it creates a problem case, automatically filling in its context portion based on the state of the natural language system at the point of the ambiguity. The structure of a problem case is identical to that of a training case except that the "solution" is missing. To resolve the current ambiguity, Kenmore presents the problem case to the machine learning system, which uses the concept description derived during training to determine the solution (i.e., class) of the current ambiguity. This information is then used directly by the parser to resolve the ambiguity.

4 Evaluation of the Approach within Practical Text-Processing Tasks

We evaluated Kenmore using corpora from two real-world domains in the context of a practical text summarization task: *information extraction*. Very generally, NLP systems for information extraction process a collection of texts, search for all information related to a predefined domain of interest, and then produce for

[2] The JV corpus contains over 1000 documents that describe world-wide activity in the area of joint ventures or "tie-ups" between businesses [21]. Unless otherwise noted, all examples are derived from sentences in this corpus.

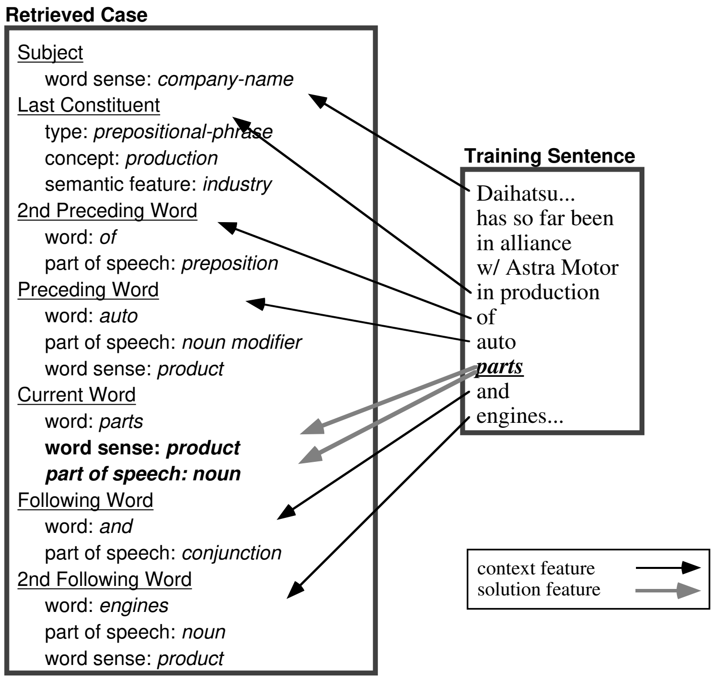

Fig. 2. Training case for "parts": part-of-speech and word-sense disambiguation. (Arrows indicate which parts of the sentence are responsible for which attribute-value pairs.)

each text a summary of this domain-relevant information in a rigid template format. In the experiments below, we tested the ability of the embedded machine learning components to handle a number of difficult problems in sentence analysis within an information extraction task: part-of-speech tagging, word-sense tagging, and relative pronoun disambiguation. Note, however, that Kenmore has been evaluated for additional language learning tasks along a number of dimensions [8] — this section summarizes just a subset of those experiments.

In all experiments, training and test sentences were drawn from the MUC/-TIPSTER corpora [20, 21] and Kenmore was instantiated with the CIRCUS parser. A variety of machine learning algorithms have been used as Kenmore's inductive learning component: a decision tree system (C4.5 [23]), a simple case-based learning algorithm (k-nearest neighbors), a conceptual clustering system (COBWEB [14]), and a hybrid inductive learning system that combines a nearest-neighbor algorithm with a decision tree system for feature selection [7]. In experiments to date, the hybrid algorithm has achieved the highest accuracies and we focus on these results below. Finally, all experiments use a 10-fold cross validation evaluation scheme. A more detailed description of the experiments and an analysis of the results can be found in [8].

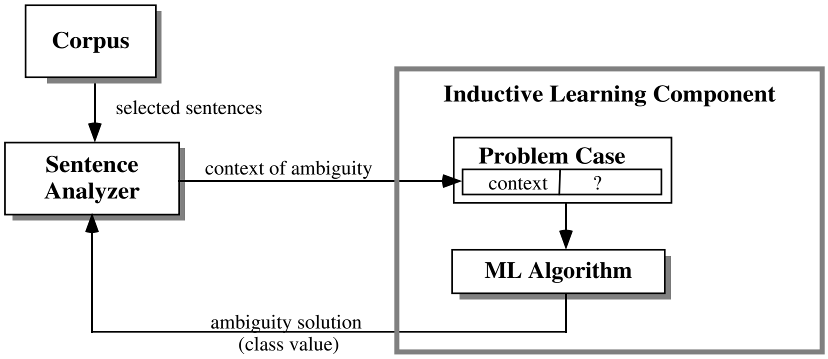

Fig. 3. Kenmore application phase.

4.1 Lexical Ambiguity

We tested Kenmore's ability to handle simultaneously two lexical ambiguity tasks: (1) part-of-speech ambiguity and (2) word sense ambiguity. There were 18 possible parts of speech including: noun, noun modifier, auxiliary, adverb, preposition, verb particle. Word senses were represented in terms of a two-level semantic feature hierarchy that included 14 general semantic classes (e.g., human, facility, joint venture entity) and 42 specific semantic classes (e.g., company name, government, factory). We assume the existence of only a small dictionary of 129 function words (e.g., prepositions, auxiliaries, determiners) — all other words are assumed to be *completely unknown* to the system in that it has no part-of-speech or semantic class information available for them. We then train the machine learning subsystem to predict the part of speech and semantic classes for all non-function words as they appear in an input text.[3] For this task, the machine learning system must deal with: (1) the consequences of making tagging errors on preceding input (as these become part of the "context" that is encoded in a problem/test case), and (2) the fact that little or no information is known about subsequent words since the parser has not yet encountered them in its left-to-right traversal of the sentence. In addition, the system currently assigns all three tags to a word simultaneously: it does not assume, for example, that the part of speech of the word is known when it assigns a semantic class value, and vice versa.

The accuracy of the machine learning subsystem is shown in Table 1 and compared to two baselines. The first baseline (labeled Random Selection) indicates the expected accuracy of a system that randomly guesses a legal tag based on the distribution of values across the training set. The second baseline (labeled Default) shows the performance of a system that always chooses the most frequent value across the training set as a default. The final two columns indicate the performance of the ML system first for the content words in the test

[3] Highly ambiguous function words are also excluded from the function word lexicon and instead are disambiguated by the machine learning component.

sentences and then across all words in the test sentences. Chi-square significance tests on the associated frequencies show that the machine learning subsystem in Kenmore performs significantly better than the baselines ($p = .01$).

Table 1. Lexical Tagging Results (% correct).

Tagging Task	Random Selection	Default	ML System	
			content words	all words
part of speech	34.3	81.5	91.0	95.0
general semantic class	17.0	25.6	65.3	80.6
specific semantic class	37.3	58.1	74.0	85.5

In practice, Kenmore performs quite well for the tested lexical ambiguity tasks. When trained on a larger set of examples (see Table 2), the learned heuristics for semantic class tagging performed well enough to be used in the UMass/Hughes CIRCUS system that participated in the the MUC-5 performance evaluation of information extraction systems [16]. In addition, the part-of-speech tagging heuristics match the accuracy of those obtained using the trigram-based tagger of the UMass/Hughes MUC-5 system and are comparable to those achievable using probabilistic taggers which generally reach accuracies in the 95–96% range [9].

In an attempt to determine the impact of the semantic feature tagging component on the overall performance of the information extraction system, we ran two variations of the UMass/Hughes MUC-5 system on test sets from the final MUC-5 performance evaluation.[4] In the first run, the system used the machine learning component to assign semantic features to all content words in the test texts. In the second run, content words were assigned a semantic type based instead on a set of handcrafted default heuristics designed to handle frequently occurring, hopefully clearcut cases (e.g., "Co." almost always indicates the name of a company). Content words that fell outside the domain of the heuristics were given a default value of *nil* as their general and specific semantic class. When compared to the learned heuristics, the handcoded default heuristics produced a 1% increase in precision and a 41% decrease in recall for the information extraction task as measured on the MUC-5 test sets. The significant drop in recall can be explained by the fact that the default heuristics were not designed to handle novel or ambiguous situations (it is too time-consuming and difficult to design such heuristics), but these "special" cases occur rather frequently in the texts. It should be emphasized that this experiment was meant only to give an indication of the extent to which the learned heuristics contributed to the information extraction system. Additional testing would be required to determine more specifically the role played by the machine learning component.

[4] Thanks to Jonathan Peterson for running these experiments.

Table 2. Semantic feature tagging using Kenmore with a larger case base (% correct).

Tagging Task	Original Results (2056 cases)	Larger Case Base (3060 cases)
general semantic class	80.6	85.7
specific semantic class	85.5	86.3

4.2 Structural Ambiguity

We have also used the embedded machine learning component to handle a single structural ambiguity task: relative pronoun disambiguation. In this task, the machine learning component acquires heuristics for locating the antecedents of relative pronouns [5, 6]. In the sentence, "I saw Mary and the boy *who* won the contest," for example, the NLP system disambiguates the relative pronoun, "who," by deciding that it refers to "the boy." Here we used texts that describe Latin American terrorist events and concentrated on locating the antecedent of "who." In all experiments, the machine learning system was also responsible for (1) deciding whether or not "who" was actually being used as a relative pronoun, and (2) recognizing antecedents that consisted of conjunctions and appositives. The experiments drew training and test cases from a base set of cases that contained all occurrences of "who" from three sets of 50 texts from the MUC terrorism corpus (241 instances). The results are summarized in Table 3. They indicate that the learned heuristics: (1) perform significantly better than a default strategy that simply chooses the single most recent phrase as the antecedent, and (2) perform as well as a set of hand-coded heuristics developed for use with UMass/MUC-3 system in the terrorism domain.

Table 3. Relative Pronoun Disambiguation Results (% correct).

ML System	Default Strategy	Hand-Coded Heuristics
84.2	74.3	80.5

4.3 Discussion of Results

Results from the experiments above indicate that symbolic inductive machine learning systems can learn solutions to a number of difficult problems in sentence analysis within the proposed Kenmore framework. There are a number of advantages of this approach over traditional methods for knowledge acquisition for natural language systems:

- The concept descriptions derived by the machine learning component implicitly define a set of disambiguation heuristics **as well as** the system lexicon (in the case of lexical ambiguities).
- The approach obviates the need for generating and maintaining explicit, handcrafted lexicons and the associated disambiguation heuristics.
- No hand-coded heuristics are required to drive the acquisition process.
- The machine learning approach to knowledge acquisition provides a flexible control structure for combining multiple sources of knowledge.

Connectionist approaches to classification could also be used as Kenmore's inductive learning component. In fact, a number of existing systems successfully combine traditional parsing techniques and distributed connectionist modules for sentence analysis (e.g., [26, 27, 10]). Symbolic machine learning systems, however, provide at least two advantages. First, there are many incremental symbolic machine learning algorithms (e.g., ID5[24], instance-based learning methods[1]) that would allow the training phase to become progressively easier for the human supervisor: Kenmore can access the existing concept description to suggest solutions to each ambiguity and rely on the supervisor only to override incorrect predictions. This type of incremental learning is generally not feasible in connectionist systems. Second, symbolic learning algorithms offer better explanations of their decisions and allow for easy incorporation of known processing biases and prior knowledge.

In the following section, we summarize the advantages of using *embedded* machine learning components in an NLP system.

5 Advantages of Embedded Machine Learning Mechanisms for NLP Systems

Corpus-based language acquisition methods recently have been the focus of much attention in the natural language processing community [4, 9, 12, 13, 28]. With very few exceptions, however, these systems treat language learning and knowledge acquisition in isolation from the actual natural language system that will use the acquired knowledge and in isolation from any specific natural language understanding task. In Kenmore, however, the learning mechanisms are directly incorporated into the larger NLP system and learning takes place within this system as it processes text. We believe that natural language processing systems can benefit in a number of ways from the use of such embedded machine learning components:

First, parsing knowledge is learned in a form directly usable by the sentence analyzer. Second, because the learning system has access to structural knowledge (from the parser) and class-level knowledge (e.g., syntactic and semantic class information for lexical items) in addition to individual words, less training data may be required when compared with statistical approaches to ambiguity resolution based on lexical cooccurrence data. Hidden Markov Models (HMM) trained using the Viterbi algorithm [25] on the same data as the Kenmore experiments above (Fig. 1, for example, achieved accuracies of 89.0% correct (bigram model) and 73.8% versus Kenmore's 95.0% for the part-of-speech tagging task.

Finally, evaluation of the language learning components can be guided by their performance within the larger natural language processing task. Ultimately, it is the performance of the NLP system in the larger task that matters and some types of errors from the machine learning component undoubtedly will be more grievous than others with respect to this task.[5] In information extraction, for example, part-of-speech tagging errors that mistake non-verbs for verbs and vice versa are the most grievous. In the lexical tagging experiments of Fig. 1, 55% of the errors were of this variety. Similarly, accurate prediction of domain-specific semantic features is more important than that of domain-independent semantic features. This is, in fact, the situation in Kenmore: it achieves 64.2% correct on domain-dependent semantic features in comparison to 71.2% correct for domain-independent semantic features. By monitoring the overall performance of the NLP system, we can focus on reducing "important" errors in the machine learning component and can end the training phase as soon as performance on the larger NLP task has reached a plateau.

5.1 The Need for New Learning Algorithms

Thus far, Kenmore has been used only for a relatively small number of natural language learning tasks. Still, however, we have found that the characteristics of the resulting datasets exhibit a number of complications for the embedded learning algorithms:

- *Noisy training (and test) cases.* Some of the noise is in the form of inconsistent and/or incorrect class information provided by the human supervisor or annotated corpus and some of the noise is in the form of incorrect feature values provided by the sentence analyzer. Because the concepts being learned are open-textured concepts that lack distinct boundaries, some portion of the noise is unavoidable.
- *Missing features in training and test cases.*
- *Irrelevant features in the training and test cases.* In the Kenmore framework, at least as described above, the parser always provides the "context" in terms of the same set of features. Intuitively, however, different subsets of features might be important for each ambiguity task.
- *Cases may not include all features necessary for the learning task to succeed.* There are many opportunities to make use of constructive induction algorithms that can add features to a case representation as needed.
- *Concepts to be learned have many classes.* The tasks tackled in the above experiments have a minimum of ten class values.

[5] It is also the case that some types of errors will be more or less important with respect to the type of *parser* employed.

– *Some features are better described as relations rather than attribute-value pairs.* Using relational learning systems (e.g., FOIL [22]) will be important here.

While most standard machine learning algorithms handle a subset of the listed problems, it proved difficult to find a single system that simultaneously addressed all of them for the NL datasets. In addition, a number of meta-level learning issues must be addressed before machine learning techniques can be employed as general-purpose learning components in a natural language understanding system: when should we stop learning? which learning algorithms are best for each language learning task? which sentences from the corpus will make the best training examples? should the current case representation evolve over time? how can information that is acquired in one learning task be transferred to another?

6 Conclusions

In summary, we have presented a general framework for the machine learning of natural language that relies on embedded, symbolic, inductive learning components. We have shown that the architecture can be use to acquire solutions to a number of difficult problems in sentence analysis within the context of a practical text-processing task. In related work, symbolic machine learning algorithms, and case-based algorithms in particular, have also been found to work well for a number of low-level language acquisition tasks (e.g., stress acquisition [13] and grapheme-to-phoneme conversion [3]) and discourse-level tasks (e.g., anaphora resolution [2, 19] and segmentation [18]). This work provides additional evidence for the viability of a machine learning approach to natural language learning and knowledge acquisition for natural language processing systems. Finally, we have argued that the language learning components should be directly incorporated into the larger natural language understanding system and should be evaluated in the context of practical language-processing tasks.

References

1. D. Aha, D. Kibler, and M. Albert. Instance-Based Learning Algorithms. *Machine Learning*, 6(1):37–66, 1991.
2. Chinatsu Aone and William Bennett. Evaluating Automated and Manual Acquisition of Anaphora Resolution Strategies. In *Proceedings of the 33rd Annual Meeting of the ACL*, pages 122–129. Association for Computational Linguistics, 1995.
3. A. van den Bosch and W. Daelemans. Data-oriented methods for grapheme-to-phoneme conversion. In *Proceedings of the Sixth Conference of the EACL*, pages 45–53, Utrecht, 1993. Also available as ITK Research Report 42.
4. E. Brill. Some Advances in Transformation-Based Part of Speech Tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 722–727. AAAI Press / MIT Press, 1994.

5. C. Cardie. Corpus-Based Acquisition of Relative Pronoun Disambiguation Heuristics. In *Proceedings of the 30th Annual Meeting of the ACL*, pages 216–223, University of Delaware, Newark, DE, 1992. Association for Computational Linguistics.
6. C. Cardie. Learning to Disambiguate Relative Pronouns. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 38–43, San Jose, CA, 1992. AAAI Press / MIT Press.
7. C. Cardie. Using Decision Trees to Improve Case-Based Learning. In P. Utgoff, editor, *Proceedings of the Tenth International Conference on Machine Learning*, pages 25–32, University of Massachusetts, Amherst, MA, 1993. Morgan Kaufmann.
8. C. Cardie. *Domain-Specific Knowledge Acquisition for Conceptual Sentence Analysis*. PhD thesis, University of Massachusetts, Amherst, MA, 1994. Available as University of Massachusetts, CMPSCI Technical Report 94–74.
9. E. Charniak. Equations for Part-of-Speech Tagging. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 784–789, Washington, DC, 1993. AAAI Press / MIT Press.
10. T. Chen, V. Soo, and A. Lin. Learning to Parse with Recurrent Neural Networks. In *Proceedings of European Conference on Machine Learning Workshop on Machine Learning and Text Analysis*, pages 63–68, 1993.
11. N. Chinchor, L. Hirschman, and D. Lewis. Evaluating Message Understanding Systems: An Analysis of the Third Message Undestanding Conference (MUC-3). *Computational Linguistics*, 19(3):409–449, 1993.
12. K. Church. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 136–143. Association for Computational Linguistics, 1988.
13. W. Daelemans, G. Durieux, and S. Gillis. The Acquisition of Stress: A Data-Oriented Approach. *Computational Linguistics*, 20(3):421–451, 1994.
14. D. Fisher. Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning*, 2:139–172, 1987.
15. W. Lehnert. Symbolic/Subsymbolic Sentence Analysis: Exploiting the Best of Two Worlds. In J. Barnden and J. Pollack, editors, *Advances in Connectionist and Neural Computation Theory*, pages 135–164. Ablex Publishers, Norwood, NJ, 1990.
16. W. Lehnert, J. McCarthy, S. Soderland, E. Riloff, C. Cardie, J. Peterson, F. Feng, C. Dolan, and S. Goldman. University of Massachusetts/Hughes: Description of the CIRCUS System as Used in MUC-5. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, pages 277–291, San Mateo, CA, 1993. Morgan Kaufmann.
17. W. Lehnert and B. Sundheim. A performance evaluation of text analysis technologies. *AI Magazine*, 12(3):81–94, 1991.
18. Diane J. Litman and Rebecca J. Passonneau. Combining Multiple Knowledge Sources for Discourse Segmentation. In *Proceedings of the 33rd Annual Meeting of the ACL*, pages 108–115. Association for Computational Linguistics, 1995.
19. Joseph F. McCarthy and Wendy G. Lehnert. Using Decision Trees for Coreference Resolution. In C. Mellish, editor, *Proceedings of the Fourteenth International Conference on Artificial Intelligence*, pages 1050–1055, 1995.
20. *Proceedings of the Third Message Understanding Conference (MUC-3)*. Morgan Kaufmann, San Mateo, CA, 1991.
21. *Proceedings of the Fifth Message Understanding Conference (MUC-5)*. Morgan Kaufmann, San Mateo, CA, 1994.

22. J. R. Quinlan. Learning Logical Definitions from Relations. *Machine Learning*, 5:239–266, 1990.
23. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1992.
24. P. Utgoff. An Improved Algorithm for Incremental Induction of Decision Trees. In W. Cohen and H. Hirsh, editors, *Proceedings of the Eleventh International Conference on Machine Learning*, pages 318–325, Rutgers University, New Brunswick, NJ, 1994. Morgan Kaufmann.
25. A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.
26. S. Wermter. Combining Symbolic and Connectionist Techniques for Coordination in Natural Language. In *Proceedings of the 14th German Workshop on Artificial Intelligence*, Eringerfeld, Germany, 1990.
27. S. Wermter and W. Lehnert. A hybrid symbolic/connectionist model for noun-phrase understanding. *Connection Science*, 1(3), 1989.
28. David Yarowsky. Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French. In *Proceedings of the 32th Annual Meeting of the ACL*, 1994.