INFORMATION FILTERS AND THEIR IMPLEMENTATION IN THE SYLLOG SYSTEM

Shaul Markovitch
EECS Dept.
The University of Michigan
Ann Arbor, MI 48109

Paul D. Scott
Center for Machine Intelligence
2001, Commonwealth Blvd.,
Ann Arbor, Michigan 48105

INTRODUCTION

The study of knowledge has always been one of the central issues of the AI research. The general belief has been that "the more you have it the better you are." It was well understood that incorrect knowledge is harmful and should be avoided, but correct knowledge had been mostly considered beneficial. The potential harmfulness of correct knowledge received attention in very few early works but has become a key issue in several recent works (Markovitch, & Scott, 1988a; Markovitch, & Scott, 1988b; Minton, 1988; Tambe, & Newell, 1988) .

Knowledge is harmful if the costs associated with retaining it are greater than its benefits. Irrelevant knowledge and redundant knowledge are two types of knowledge that very often is turned to be harmful. When the knowledge that a problem solver uses is supplied by a human, we can hope that it does not contain harmful elements. However, when the knowledge is acquired by a learning program, it is desirable that the harmfulness of the knowledge will be eliminated, or at least reduced by the program itself.

We introduce in this paper the notion of information filters. Information in a learning system flows from the experiences that the system is facing, through the acquistion procedure to the knowledge base, and thence to the problem solver. An information filter is any process that removes information at any stage of this flow. We call filters that are inserted between the experience space and the acquisition procedure *data filters*, and filters that are inserted between the acquisition procedure and the problem solver *knowledge filters*.

INFORMATION FILTERING

The underlying assumption of our discussion is the following general framework of a learning system. The system consists of (1) a problem solver that faces a set of tasks (2) an evaluation criteria that evaluates the performance of the problem solver and (3) acquisition module whose output is knowledge useable by the problem solver. Given such a framework, we can precisely define the notion of harmful knowledge. Basically, a knowledge element is harmful if the problem solver performance measured according to a given criteria is better without the knowledge element than with it (Markovitch, & Scott, 1989b).

Information filters are functions that are inserted between the input to the learning system and the input to the problem solver. The role of these functions is to eliminate (or reduce) harmful knowledge. There are five different locations where an information filter can be placed. It can filter the set of experiences that the learning system faces. We term such filtering as *selective experience*. It can filter the set of events or features that the system process out of a particular experience. We term such filtering as *selective attention*. Both are data filters. Knowledge can be filtered after it is generated by the acquisition procedure and before it is entered into the knowledge base. We term such filtering *selective acquisition*. If the knowledge base is connected

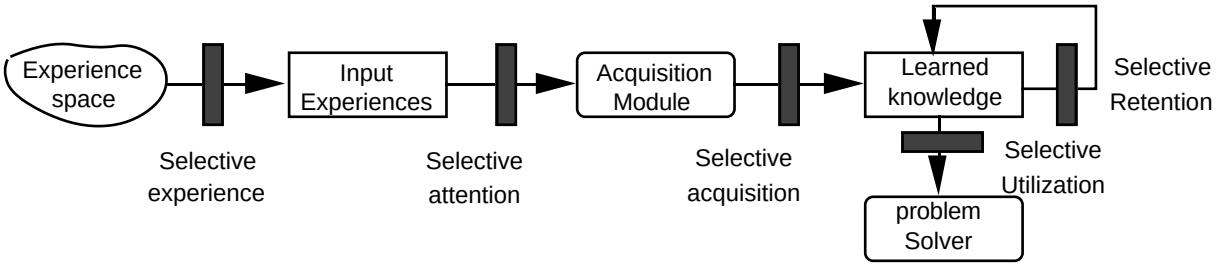

Figure 1.

to the input of a filter, and the output of the filter is connected to the same knowledge base, we have *selective retention* (or forgetting). If the knowledge is filtered between the knowledge base and the performance program, we have *selective utilization*. Figure 1 illustrates the five types of information filters.

Information filters can be a fixed part of the system architecture, or they can be learned. To differentiate between the process of learning knowledge to be used by the problem solver and the process of learning knowledge to be used by the filters, we call the first *primary learning* and the second *secondary learning*.

INFORMATION FILTERS IN THE SYLLOG SYSTEM

The SYLLOG system is a learning program that is built on top of a Prolog interpreter. The program learns domain specific knowledge by exploring the set of axioms that describes the domain. The domain specific information is then used by the interpreter to increase the efficiency of its search. SYLLOG employs the two-level learning scheme described above. Its primary level learning includes both deductive and inductive learning processes. The deductive learning program acquires lemmas – instantiated subgoals that were proved during learning. The lemmas can later be used by the interpreter as if they were regular axioms. The inductive learning program acquires information about the average costs and average number of solutions of subgoals with specific calling patterns. The knowledge about cost and number of solutions is used by the interpreter when it looks for a good ordering of the subgoals of a rule (Markovitch & Scott 1989c). The secondary level learning includes procedures for acquiring knowledge needed by the different filtering mechanisms. SYLLOG uses all five types of filters. The next two subsections discuss two of those.

SELECTIVE EXPERIENCE - THE TASK GENERATOR

SYLLOG employs a learning method called "learning by doing" or "learning by experimentation," i.e., it feeds the problem solver with problems to solve and acquires a variety of knowledge during the process of problem solving. If there are enough problems that were given in the past by external agents, then the system can use these problems as training examples. If more training problems are needed , SYLLOG generates its own problems.

Since there is potentially a very large number of possible problems, the system may invest most of its learning resources in solving problems that will lead to the acquisition of irrelevant knowledge (which is likely to be harmful). SYLLOG employs an experience filter in order to decrease the likelihood that the acquired knowledge will be harmful by building the *task model* – the system's view of what problem space it faces. Currently, SYLLOG represents its task model by a weighted set of *calling patterns.* A calling pattern is a predicate name followed by a

list of 0's and 1's that indicates which of the arguments of the predicate are bound and which are not. The task model also includes a set of domains for the various predicates.

The input of secondary level learning procedure that acquires the task model is the set of queries that the system was given in the past by external agents, and its output is the model itself. Whenever the task generator is required to produce another training example, it selects a random calling pattern according to its weight, and selects random constants from the domains to replace all the 1's in the pattern. Thus, the primary level acquisition programs face only small subset of the experience space - a subset which is similar to the set of problems that were given to the system in the past, and is less likely to produce harmful (irrelevant) knowledge.

SELECTIVE USE OF LEMMAS

For most lemmas, one can not tell whether they are useful without referring to a specific problem. If a branch of the search tree fails, then all the uses of lemmas within that branch are bound to be harmfulbecause the interpreter first uses the lemmas, and then, it is forced by the backtracking mechanism to reinvoke the rules that produced the lemmas in the first place. Doing so increases substantially the cost of the search: the cost of matching the subgoal against all the lemmas is added to the cost of the search without the lemmas. In addition, each solution is generated twice - once by the lemma itself, and once by the rules that had generated the lemma, hence all the following subgoals will be invoked twice as many times.

The phenomenon described above is inherent to any search procedure that employs backtracking and that uses deductively learned knowledge (Markovitch, & Scott, 1989a). The first four types of filters can not eliminate such harmfulness because the context of using the lemma determines whether its use is harmful or not.

To reduce the harmful uses of lemmas, SYLLOG incorporates a filter that allows the problem solver to selectively use lemmas. The filter is acquired by a second level inductive learning procedure that accumulates knowledge about the probability of subgoals with certain calling patterns failing within the given problem space. Whenever the search procedure creates an OR node, it appends all the entries for the goal's predicate to the set of lemmas for that predicate after they pass through the filter function. The filter estimates the probability of the current calling pattern of the goal failing. If it is above certain threshold (see (Markovitch, et al., 1989a) for experiments with different threshold values), the filter returns the empty set. In addition, a flag is propagated to the whole subtree below that node to turn lemma usage off. The idea is that if a subgoal fails, all lemma usage in the subtree below that subgoal is bound to be harmful. Therefore, for a subgoal which is likely to fail, lemma usage is likely to be harmful.

Figure 2 shows the learning curves of the lemma learner with and without filtering. The experiments were done with a Prolog program that describes the physical layout of a local area network and is used for proving correctness of the layout and for diagnosis purposes. Each data point represents the mean number of unifications for a test consisting of 20 problems randomly generated from the same problem space as the one used for learning. The same set was used for all tests, and learning was turned off during testing. It is interesting to note that without filtering, lemmas were actually harmful, degrading the system performance substantially, whereas the use of a filter reduces the harmfulness of the lemmas to a point where the system's performance improved by a factor of two compare to the performance without using lemmas (and by a factor of 5 compare to the performance with lemmas and without filters).

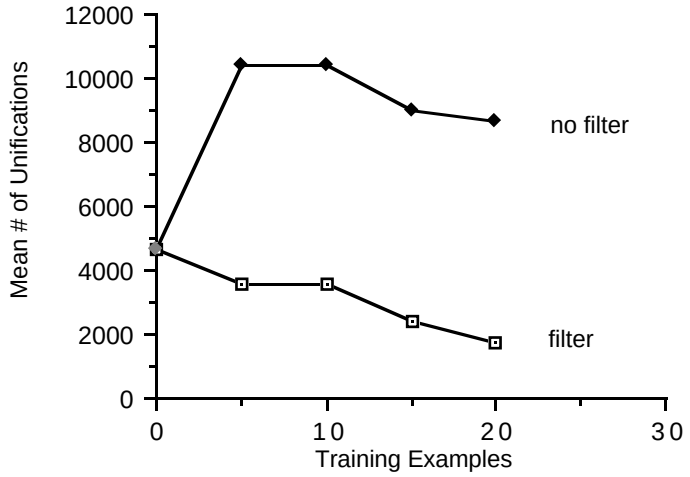

Figure 2.

CONCLUSIONS

This paper introduces a general framework for reducing harmfulness of learned knowledge. The framework defines information filters and lists five types of such filters according to where they are inserted in the system. Implementation of the filters within the SYLLOG learning system is described, in particular, two filters: selective experience and selective utilization. Results of experiments with utilization filter were shown to demonstrate the usefulness of the concept. For further details the reader should refer to (Markovitch, et al., 1989b).

References

Markovitch, S., & Scott, P. D. (1988a). *Knowledge Considered Harmful* (TR 030788). the Center for Machine Intelligence, Ann Arbor, Michigan.

Markovitch, S., & Scott, P. D. (1988b). The Role of Forgetting in Learning. *Proceedings of The Fifth International Conference on Machine Learning* . Ann Arbor, MI: Morgan Kaufmann.

Markovitch, S., & Scott, P. D. (1989a). Utilization filtering: a method for reducing the inherent harmfulness of deductively learned knowledge. *Proceedings of IIJCAI*. Detroit, Michigan.

Markovitch, S., & Scott, P. D. (1989b). *Using Information Filters to reduce Harmfulness of Learned Knowledge* (TR 009). the Center for Machine Intelligence, Ann Arbor, Michigan.

Markovitch, S., & Scott, P. D. (1989c). *Automatic Ordering of Subgoals - a Machine Learning Approach* (TR 008). the Center for Machine Intelligence, Ann Arbor, Michigan.

Minton, S. (1988). *Learning Search Control Knowledge: An Explanation-Based Approach*. Boston, mass: Klower Academic Publishers.

Tambe, M., & Newell, A. (1988). Some Chunks Are Expensive. *Proceedings of The Fifth International Conference on Machine Learning* . Ann Arbor, MI: Morgan Kaufmann.