A Constraint-Motivated Model of Concept Formation*

Craig S. Miller and **John E. Laird**
Artificial Intelligence Laboratory
The University of Michigan
1101 Beal Ave.
Ann Arbor, Michigan 48109-2110

Abstract

A cognitive model for learning associations between words and objects is presented. We first list basic constraints to which the model must adhere. The constraints arise from two sources. First they stem from observed psychological phenomena including typicality effects, extension errors observed from children and belief-dependent behavior. Secondly they arise from our choice to integrate the model in a unified theory of cognition. In presenting the constraints to the model's construction, we motivate our design decisions while describing our algorithm that takes a symbolic, production-based approach. The model's adherence to the constraints is further supported by some empirical results.

Introduction

In this paper, we present a model for concept acquisition similar to the learning tasks in systems such as ID3 [Quinlan, 1986] and COBWEB (when used to predict features) [Fisher, 1987]. However, our model has been subjected to further constraints in its role of a psychological model. In particular, we account for phenomena observed in children acquiring new words as well as the more general robust phenomena observed in conceptual acquisition. Furthermore, in order to ground our model in a broader, existing theory, we propose to implement the model in a unified theory of cognition.

In this paper, the task requires our model to predict object names of unlabeled objects based upon training examples that include the object name. For our task, we choose to represent object instances as lists of attribute-value pairs. For example, a particular instance of a ball may be represented as follows: [shape:spherical color:blue texture:smooth size:large]. While limiting instance descriptions to lists of attribute-value pairs admittedly restricts expressibility, the representation still permits specifications for an adequate assortment of objects and easily suffices for illustrating some of the model's interesting behaviors.

*A similar version of this paper appeared as [Miller and Laird, 1991].

Design constraints

In this section, we present the major constraints that led us to the model's current design. The psychological motivations are discussed for each constraint. In the next section we will explain our model and how these constraints influenced its design.

Constraint 1 *The word-learning model must be consistent with a unified theory of cognition.*

A unified theory of cognition is a proposed system of mechanisms capable of producing the full range of human cognition [Newell, 1990]. Ideally a model should not operate in isolation but rather play an integral and cooperative role with other cognitive activities. By conforming to a unified theory, the relationship of the various cognitive faculties becomes more apparent.

Constraint 2 *Learning must be incremental.*

A child learning words does not have the opportunity to learn her entire vocabulary before applying it. Instead, learning and performance are interleaved. The learning of words is an ongoing, *incremental* process. For our purposes, we will consider a learning model to be incremental if the computational complexity of integrating a new instance into the system's knowledge is roughly constant relative to the number of training trials.

Constraint 3 *The model must exhibit typicality effects.*

Typicality effects include: 1) typical instances are generally processed faster than less typical ones, 2) typical instances lead to fewer errors in category prediction and 3) typical instances are frequently given as an example to a category [Rosch, 1978]. In this paper, we limit ourselves to showing how typical instances are processed faster than less typical ones.

Constraint 4 *The model must exhibit underextension errors in the early stages of learning.*

Underextensions result when the child's application of a label is too narrow in contrast to the full adult category. In regard to our model, an underextension occurs when no object name prediction is delivered for a particular object instance even though objects of the same category have been previously encountered. For example, a child may

be able to identify that a softball is a 'ball' but fail to identify a football as being one. These underextension errors often occur during the early stages of lexical development [Dromi, 1987].

Constraint 5 *The model must be sensitive to knowledge as determined by the system's beliefs, goals and theories.*

It is well documented that beliefs, goals and theories play an important role in human categorization [Murphy and Medin, 1985, Schank *et al.*, 1986]. Our model addresses this constraint by operating at the deliberate level, i.e. it can potentially bring to bear all pertinent knowledge in guiding its decisions.

Constraining the design

In applying our first constraint, we choose to construct our model within the confines of Soar, a proposed unified theory of cognition [Newell, 1990]. Soar presents its theory in the form of an architecture—a system of mechanisms that applies knowledge, represented as productions, in creating intelligent behavior. We will further motivate our choice of Soar, but first let us focus on the new constraints that result from this choice. They include: 1) all learning occurs through an experience-based learning mechanism, 2) the system does not have direct access to its long-term memory and 3) processing is fully symbolic with no native support for frequency counts or probabilities. A basic understanding of the system and its assumptions follow within this section. In particular, a short review explains some of the basic ideas of how induction should proceed in Soar. We will then explain our model and how it coincides with the rest of our constraints.

Learning in Soar

Problem solving in Soar [Laird *et al.*, 1987] consists of the sequential selection and application of operators to a representational state within a problem space. Both the selection and application of operators is determined by the knowledge represented in a long-term recognition memory encoded as productions. Learning involves the construction of new productions, called chunks, which summarize the results of a subgoal. A chunk in Soar is similar to an operationalized result in explanation based learning [Mitchell *et al.*, 1986, Dejong and Mooney, 1986]. Subgoals are born out of impasses caused by conflicting knowledge or a lack of knowledge. Subgoaling relies on additional knowledge to consider the alternatives, often requiring further search. During subgoaling, the system may try out the alternatives or it may recast the problem. In any case, the subgoal resolves the problem that caused the impasse. Furthermore, the architecture traces back through the conditions that led to the impasse's resolution. Using these conditions, a new production is created that summarizes the subgoal's result so that, in analogous applications, the summarizing information is retrieved. This avoids the impasse and thus the costly search encountered when the chunk was first created. In this scenario, the knowledge in the problem space has been partially operationalized in that an increase in efficiency has occurred. Although counter-intuitive, previous work with Soar has demonstrated that this technique can also increase the total knowledge of the system [Rosenbloom *et al.*, 1988]. This approach, called data-chunking, is a precursor to the work we present here.

Describing the model

For our approach, we rely exclusively on chunking to monotonically add productions that predict a concept name given a partial description of an object. The definition of a concept is not localized to a single production, or even a small set of productions. Instead, a concept definition is distributed across the set of productions that predict the concept name, plus a process that creates abstractions of object descriptions, attempting to find an object description that can be matched by the productions. The search plays an integral role in defining the concept; for it is here that Soar brings to bear its knowledge in determining which abstractions should be considered and in what order. In the model presentation that follows, we will see how this fully symbolic learning process adheres to the constraints described in the previous section.

When attempting to predict the object's category, the object's description serves as the initial state in a problem space. For example, if we want to classify a blue, smooth, spherical object as a ball, the initial state would contain the following object description: [shape:spherical color:blue texture:smooth].

The operators of the problem space recognize descriptions and predict categories. Initially, the system will have very few operators and be able to predict only a few categories, but with experience, more operators will be learned. For our example, let's assume that the system already has acquired the following productions for creating word-prediction operators:

```
[spherical, red, smooth] -->
  create-operator(predict word:ball)
[spherical, red] -->
  create-operator(predict word:ball)
[spherical] -->
  create-operator(predict word:ball)
```

If one of these productions exactly matched the object description, it would fire, create an operator which would then be selected and predict a word and we would be done. However, for a production to apply, we insist that it matches every feature in the object description, so that a general production will only match an abstracted description.[1] This means that even the third production is not applicable for our initial object description because it has no feature match for color and texture.

If an object can not be classified by the existing knowledge, no operators are created and an impasse results. Within the ensuing subgoal, a search is performed to find

[1] Although Soar's production matching generally does not have this property, by representing object descriptions appropriately (i.e., as linked lists), the object recognition productions will have this property.

an abstraction that modifies the object description in such a way as to allow it to be categorized. For our example, the abstraction operators render an object description more general by removing one of the attribute-value pairs from the description. Let us assume the chosen operator removes the texture feature from the description, producing the description (spherical, blue). Following an abstraction, there is the potential that the new description can be categorized. However in this example, no match occurs, and another impasse and subgoal are created. Only after the color has been abstracted out, will the object description be recognized by the third production and a prediction for the word will be made. For performance trials, the correctness of the prediction will depend on the object recognition productions store in long-term memory, as well as the abstraction performed during the search for a match.

In this formulation, there is no explicit abstraction hierarchy; instead the abstractions are generated dynamically. This allows knowledge to be used to determine which abstractions should be made first. The ramifications of this approach are that novel objects will be classified into existing categories if it differs only in features that are abstracted early on.

For training trials, the purpose is not to predict a word, but instead to learn the association between an object description and a word. Consistent with earlier work in data chunking, our approach is to treat a training trial like a performance trial, so that the system will attempt to predict which word it already would associate with the given object. The prediction will be verified by comparing it against the given word before accepting it. If necessary, abstraction will continue until a very abstract production for predicting the word is found. When the correct prediction is finally made, the training trial is finished.

How has the training resulted in any learning? Recall that in Soar, learning happens when a result is produced in a subgoal. In this case, the result is the correct prediction created in the final subgoal where the object is correctly categorized. For a performance trial, when no word accompanies the description, there is no attempt at verification and no chunk is created. For training, learning occurs for the subgoal lying just above the subgoal where recognition was achieved and verified. There the object description [spherical, blue] eventually led, following abstraction, to the results created in the lower subgoal (predict word:ball). Chunking summarizes the abstraction and recognition processing in the subgoal, and builds a new production:

```
[spherical, blue] -->
  create-operator(predict word:ball)
```

In future training trials, this production will be available and allow a prediction to be made even earlier, and thus allow an even more specific production to be learned. As practice continues, more and more specific productions are acquired. So that, although the search for a category proceeds from a specific example through varying levels of abstraction, the learning proceeds from most abstract to more specific. Thus, this learns the same way as discrimination learning and previous data-chunking work in Soar [Rosenbloom *et al.*, 1988] (abstract to specific), but this scheme attempts to perform classification by moving from specific to general (through abstraction operators) while in a discrimination test it is from general to specific (through successive tests for the values of more and more features).

Before continuing with our model's description, let us address how the approach already fulfills some of our initial constraints. First of all, since training and performance runs can be interleaved, learning is incremental, thus fulfilling our second constraint. In a sense, the Soar architecture already enforces this by disallowing direct access to long-term memory, i.e. productions cannot directly access, delete or modify productions. This simply prohibits any action of massively reorganizing the long-term memory every time a new learning instance is encountered. The model itself has a surprising consequence in that learning time per instance can potentially decrease as new instances are acquired. This is a result of having each learning trial build upon chunks acquired on previous learning trials. The more instances the system has encountered, the less likely a new object description will have to be fully abstracted in order to be integrated with the current set of productions.

The typicality effects (Constraint 3) are also a result of how the model searches for the best and most specific match first and then slowly generalizes until a match is found. Here queries using frequent and typical objects will find a match that is not only a better match, but they will also find it quicker. This assumption stems from the notion that typical instances will already have several stored exemplars that nearly (if not completely) match them. Therefore little abstraction is required in order to find the match.

So far we have assumed that there was always some way of classifying an object correctly, even if it had not been seen before. Within our model, that translates into productions that do not test any features and predict the words. Thus, the system must be "bootstrapped" with productions of this form:

```
[] --> word:ball
[] --> word:book
[] --> word:apple
```

Where do these productions come from? They in turn must be learned from lower-level components such as phonemes. This recursion bottoms out in some primitive set of components that the system can inherently generate.

The consequences of starting with these very general productions could cause one to think that many overgeneralized predictions during performance would initially result, thereby contradicting our fourth constraint, that underextensions must initially result. Some overextensions will be produced for this reason, but surprisingly, the choice of these very generalized productions actually lead to many initial underextensions. During performance, these very general productions will often suggest conflicting predictions. Since the system has no way to choose which prediction is correct, *no prediction is made when-*

ever there is a conflict in predictions.[2] Because the productions are so general, initially there will be many abstractions in trying to categorize a word, and thus there will be many conflicts. Therefore, there will initially be many underextensions.

We have already discussed how our choice of Soar has led to an incremental learning model. Our choice has also contributed to fulfilling some of our other constraints. First of all, we argue that placing the task on Soar's deliberate processing level is the appropriate for modeling typicality effects. In several psychological experiments, it was shown that differences in response times due to differences of similarity varied on the order by several hundreds of milliseconds [Rosch *et al.*, 1976]. Results from other research and analysis of Soar has led us to conclude that the application of a primitive operator in Soar corresponds to a human time of approximately 50 milliseconds [Newell, 1990, Wiesmeyer and Laird, 1990]. Since difference in response times in our model vary by several operator applications, these results suggest that we are in the ball park and that our explanation of typicality effects on an *algorithmic* level is reasonable. This contrasts with other previous work that seeks a quantitative approach (i.e. typicality is represented in the form of frequency counts or probabilities) for explaining typicality effects. In taking these approaches, strong assumptions are required concerning the underlying implementation in order to explain similarity-dependent response times. For example, in COBWEB, category prediction involves traveling to a node that explicitly represents the instance's category. Typicality timing effects come only with the strong implementational assumption that "the time required to reach a node is inversely proportional to the total predictiveness towards that node." [Fisher, 1988].

Finally Soar provides us with the support of fulfilling our last constraint—goal and belief dependent behavior in categorization. In Soar, the selection of an operator is a deliberate act, i.e. all applicable knowledge can potentially influence the choice of an operator. This knowledge, expressed through productions preferring some operators over others, can also be learned through the result of subgoaling. Although this paper does not emphasize the learning of this knowledge, the issues presented here remain a central concern of our research work. Briefly we can say that we are concentrating on three kinds of knowledge: 1) default preferences that prefer abstracting out some features before others, 2) abstraction preferences learned from prediction failures and 3) abstraction preferences favoring features that lead to the system's goals.

Empirical results

In this section, we present two experiments that confirm our predictions on extensional errors and typicality effects.

[2]Conflicting predictions do not arise during training because operators that suggest incorrect predictions are rejected. This is possible because the correct answer is provided.

	Average time in decision cycles		
Training Trials	Similarity groups $S <= 2.0$	$2.0 < S <= 3.0$	$3.0 < S$
1	23.0	18.2	17.0
2	23.0	12.2	9.8

Table 1: Response time according to similarity group

For both of these, we chose an assortment of objects that are simply described by attribute-value pairs, similar to the examples given so far. For all our data, we chose objects that roughly correspond with physical objects children are likely to encounter. The abstraction operators generalize the instance descriptions by removing a feature. Furthermore, we have included additional heuristics that guide the selection of these abstraction operators.

Our first experiment demonstrates the effect that similarity has on response time. A training set was presented twice to the system. It is a collection of object descriptions, all of them representing different kinds of balls. With each description, the appropriate French word for ball was given (either *ballon* or *balle*). In French, *ballon* is used to refer to balls that are inflated (basketballs, volleyballs, footballs, etc.), while *balle* refers to solid balls (baseballs, golf balls, tennis balls). Thus, the system must learn to distinguish between objects in the two categories even though the critical feature (inflatable or not) is not included in the input. Between the training trials, a set of object descriptions were run on the system. For each of the performance object descriptions, the similarity was mathematically calculated based upon the number of shared features the object description has with the training object descriptions of the same category. We have divided the performance descriptions into three groups of low, medium and high similarity and average their response times in terms of decision cycles. Table 1 shows that the instances with a lower similarity measure require a longer response time. This table also shows how learning times can improve as new knowledge is acquired. The performance is faster after the second training trial because more specific chunks have been acquired. This means less abstraction need occur in order to make a prediction.

In the second experiment, we trace extensional errors by testing performance on one category of objects. Here training instances drawn from several categories are given one at a time. Between each training instance, a set of testing instances, all from the same category, are run on the system. Figure 1 shows the results of the trace. Here we see an initial onslaught of underextensions and overextensions before correct predictions dominate. We consider the category to be *underextended* whenever the system fails to make a prediction. The category is *overshadowed* by another category whenever a wrong response is given for the category we are testing. The latter is a case of *overextension*, and we have included in our figure a trace of these errors for the sake of completeness. The par-

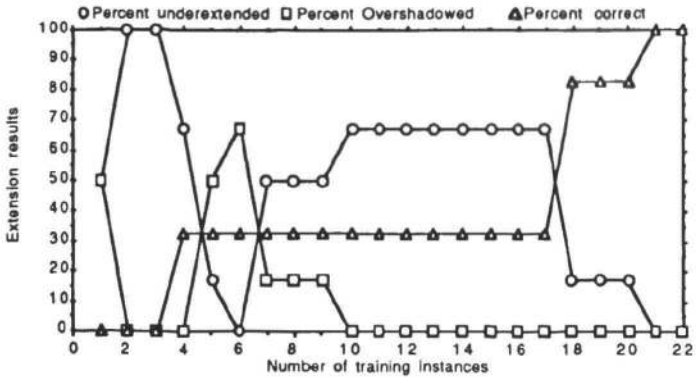

Figure 1: Trace of extensional errors

ticular conclusion that can be drawn from this figure is that several initial underextensions result from our model even though the system starts by learning more general productions first. As we have said before, this is the result of several general conflicting rules matching at the same time.

Future work

The model we have presented here is a first pass at a system which conforms to our selected set of constraints. For future work, we intend to improve the model by expanding its capabilities while also having it adhere to more constraints. The expansion of capabilities include the learning of abstraction heuristics, extending learning to predict missing features, and increasing instance description expressibility. For adding more constraints, we are looking at covering additional typicality effects and basic-level effects.

Acknowledgements This work was supported by a grant from the W.K. Kellogg Foundation through The Presidential Initiative Fund of The University of Michigan. We would like to thank Melanie Mitchell, Steve Lytinen and Mark Wiesmeyer for their helpful comments on earlier drafts of this paper.

References

[Dejong and Mooney, 1986] G. Dejong and R. Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1:145–176, 1986.

[Dromi, 1987] E. Dromi. *Early Lexical Development.* Cambridge University Press, New York, 1987.

[Fisher, 1987] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[Fisher, 1988] D. H. Fisher. A computational account of basic level and typicality effects. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 233–238, 1988.

[Laird *et al.*, 1987] J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[Miller and Laird, 1991] C. S. Miller and J. E. Laird. A constraint-motivated model of lexical acquisition. In L. Birnbaum and G. Collins, editors, *Machine Learning: Proceedings of the Eighth International Workshop*, San Mateo, CA, 1991. Morgan Kaufman.

[Mitchell *et al.*, 1986] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: A unifing view. *Machine Learning*, 1:47–80, 1986.

[Murphy and Medin, 1985] G. L. Murphy and D. L. Medin. The role of theories in conceptual coherence. *Psychological Review*, 92:289–316, 1985.

[Newell, 1990] A. Newell. *Unified Theories of Cognition.* Harvard University Press, Cambridge, MA, 1990.

[Quinlan, 1986] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Rosch *et al.*, 1976] E. Rosch, C. Simpson, and R. S. Miller. Structural bases of typicality effects. *Journal of Experimental Psychology: Human Perception and Performance*, 2:491–502, 1976.

[Rosch, 1978] E. Rosch. Principles of categorization. In E. Rosch and B. B. Lloyd, editors, *Cognition and Categorization*, pages 27–48. Erlbaum, Hillsdale, NJ, 1978.

[Rosenbloom *et al.*, 1988] P. S. Rosenbloom, J. E. Laird, and A. Newell. The chunking of skill and knowledge. In B. A. G. Elsendoorn and H. Bouma, editors, *Working Models of Human Perception.* Academic Press, London, 1988.

[Schank *et al.*, 1986] R. C. Schank, G. C. Collins, and L. E. Hunter. Transcending inductive category formation in learning. *Behavioral and Brain Sciences*, 9:639–686, 1986.

[Wiesmeyer and Laird, 1990] M. Wiesmeyer and J. Laird. A computer model of 2d visual attention. In *The Twelfth Annual Conference of the Cognitive Science Society*, pages 582–589, 1990.