A PRODUCTION SYSTEM MODEL OF FIRST LANGUAGE ACQUISITION

Pat Langley
Department of Psychology
Carnegie-Mellon University
Pittsburgh, Pennsylvania USA 15213

Abstract

AMBER is a model of first language acquisition that improves its performance through a process of error recovery. The model is implemented in ACTG, an adaptive production system language. AMBER starts with the ability to say only one word at a time, but adds rules for inserting additional words in the correct order, based on comparisons between predicted and observed sentences. These insertion rules may be overly general and lead to errors of commission; in turn, these lead to more conservative rules with additional conditions. AMBER's learning mechanisms account for many of the developments observed in children's speech.

Introduction

The acquisition of language has been a popular topic among researchers in Artificial Intelligence. Impressive language learning programs have been developed by Siklossy [1], Hedrick [2], Anderson [3], Selfridge [4], and Berwick [5]. The generality and power of these systems vary greatly, but they share one characteristic: none of the programs provide a psychologically plausible model of children's language learning.

In this paper I describe the beginnings of a more realistic model of first language acquisition. This model is called AMBER, an acronym for Acquisition Model Based on Error Recovery. As its name implies, the model simulates the incremental nature of the child's language learning process. AMBER is concerned with the production component of children's speech, since most of the reliable data relate to production rather than the understanding process.

Below I summarize the major developments found during this period. After this, I present an overview of ACTG, the production system language in which the model is stated. Next I consider some assumptions about the child's linguistic knowledge at various stages during the learning process. After considering the initial and final stages of AMBER, I discuss the learning mechanisms leading to the transition process. Finally, I consider the limitations of the model and propose directions for future research.

The Major Phenomena

Children do not learn language in an all-or-none fashion. They begin their linguistic careers uttering one word at a time, and slowly evolve through a number of stages, each containing speech more like that of the adult than the one before. In this section I discuss the features of the three stages which AMBER attempts to explain. I discuss these stages in their order of occurrence, dealing only with the major phenomena in each case.

The One-Word Stage

Around the age of one year, most children begin to produce words in isolation, and continue this strategy for some months. Presumably the child spends much of this period connecting particular words to particular concepts; once this has been done, he can produce these words under the appropriate circumstances. AMBER does not attempt to explain the word-learning process. Like Anderson's LAS [3], it assumes that links between words and concepts have already been established.

Bloom [6] has examined this period in detail, with an eye to understanding the relation between the one-word stage and those which follow it. Early in this stage, successive one-word utterances seem entirely disconnected; the child randomly comments on anything that happens to be in the environment. Later, he begins to name in succession different aspects of the same event or object; words are still separated by noticeable pauses and no regular order can be detected, but conceptual continuity seems present. Moreover, this development occurs only a few months before the child begins to combine words into very simple sentences. AMBER's starting point lies somewhere within this later part of the one-word stage.

Telegraphic Speech

Around the age of 18 months, the child begins to combine words into meaningful sequences. In order-based languages such as English, the child usually follows the adult order. Initially only pairs of words are produced, but these are followed by three-word and later by four-word utterances. The simple sentences occurring in this stage consist almost entirely of content words. Brown [7] has described speech during this period as telegraphic, since grammatical morphemes such as tense endings and prepositions are absent, as they would be in a telegram.

Brown has also noted that the majority of two-word utterances express a rather small set of pairwise semantic relations. AMBER assumes a small number of case relations such as agent, action, and possession from which Brown's pairwise relations can be derived. In addition, the child uses a few function words like "there", "more", and "all-gone" to express simple forms of nomination, recurrence, and negation. AMBER attempts to learn the relative word orders for expressing these recurring relations.

The Acquisition of Grammatical Morphemes

Brown [7] has also studied the period from about 24 to 40 months, during which the child masters the grammatical morphemes which were absent during the previous stage. Brown pointed out that these morphemes modulate the major meanings of sentences which are expressed through content words. AMBER reflects this distinction by representing the information expressed by contents words and grammatical morphemes in different ways. These morphemes are learned gradually; the time between the initial production of a morpheme and its mastery (i.e., when it is correctly used in all required contexts) may be as long as 16 months.

In addition, Brown has examined the order in which 14 English morphemes are acquired, and has found this order to be remarkably consistent across children. For example, present progressive (eat*ing*) and plural (dog*s*) were always learned quite early, while third person singular (eat*s*) and copulas (is, are) took longer. He found that the syntactic and semantic complexities of the morphemes were highly correlated with their order of mastery. Since the current version of AMBER cannot deal with exceptions, I will consider only regular constructions in this paper.

The ACTG Formalism

AMBER is implemented in ACTG, an adaptive production system language. Below I present an overview of ACTG, beginning with a discussion of its propositional network. After this I consider the representation of procedures as productions. Finally, I examine ACTG's facilities for changing its own behavior through the creation of new productions.

The Propositional Network

ACTG stores its factual, declarative knowledge in a long-term propositional network. Individual facts are stored as propositions, which may be arbitrary list structures. As we will see in more detail below, AMBER incorporates two main types of propositions. One sort expresses a goal to say a particular word in a certain position. The second type of proposition expresses various kinds of relations, including facts like x possesses y, y is a *ball, and "ball" is the word for *ball (where concepts are preceded by "*" to distinguish them from their associated words).

At any given time, some subset of the propositional network is active. Many of the active propositions have been recently added to the network by productions. Others, after lying dormant for a time, have been reactivated through their association (i.e., sharing of symbols) with other recently activated facts. AMBER uses this process of spreading activation primarily to retrieve information about the words associated with particular concepts. The level of activation for a proposition naturally decays over time, unless it is offset by other factors.

The Production System

ACTG represents procedural knowledge as a set of condition-action rules called productions. The conditions and actions of these rules can be quite general, since they may contain variables that match against arbitrary structures. When all the conditions of a production match against some portion of active memory, its actions may be carried out. These may interact with the environment, or add new propositions to the active part of the network. Structures matching variables in the conditions remain bound to these variables in the actions. After a production has been applied, the state of memory is reexamined and the system cycles.

If two or more productions are found to be true, one must be selected in preference to the others. This decision is based on the relative strengths of the productions, and on the summed activations of the propositions matched by each. The product of these two numbers is computed, and the production with the highest value is selected. Since a single production can match against a set of propositions in different ways, ties may sometimes occur. In such cases, one of the matches is selected at random.

The ACTG Learning Mechanisms

ACTG incorporates a powerful set of mechanisms for modeling learning phenomena. The most basic of these is the designation process, which allows the creation of a new production as one of the actions of an existing rule. Variables bound in the conditions of the learning rule are passed to the offspring, making the new rule more specific than its creator. Most of AMBER's learning heuristics rely on the designation process.

A second mechanism leads to the strengthening of a production each time it is recreated. Since the strength of a rule plays an important role in the selection phase, productions which have been relearned many times will be preferred. On the other hand, the strength of a rule can be decreased if it leads to an error, lowering its chances for selection.

The discovery of an error also leads to a call on the discrimination process. Here the recent firings of the responsible production are examined. If one or more propositions have been present at successful firings and absent at faulty ones, they are added as extra conditions on a new, more conservative version of the rule. Together with the strengthening and weakening processes, this mechanism gives ACTG the ability to recover from overgeneralizations.

AMBER's Linguistic Knowledge

Learning is the result of an interaction between a set of relatively general techniques for acquiring knowledge and the environment in which they find themselves. In this section I consider AMBER's representation of that environment. After this I examine the procedures the model assumes at the outset, as well as the form of the rules at which it eventually arrives.

Representing Sentences

Before AMBER can learn how to generate legal sentences, it must be exposed to examples of such sentences. One might represent a sentence as a simple list of words in the order they are said. However, though children learn to produce words in the correct order very early on, they also omit many words that an adult would include. For example, the utterance "Daddy ball" omits information about the action being carried out, as well as tense information. AMBER's representation of the sentences it hears reflects this ability to note order in the absence of information about adjacency.

The model represents the occurrence of each morpheme as a separate proposition, each containing information about the speaker, the word being produced, and the relative order of occurrence. Thus, the fact that Mommy said the sentence "Daddy bounces the ball" would be stored as a set of seven propositions: (said 1 Mommy pause); (said 2 Mommy Daddy); (said 3 Mommy bounce); (said 4 Mommy s); (said 5 Mommy the); (said 6 Mommy ball); and (said 7 Mommy pause). The first and last propositions act as delimitors which mark the beginning and end of the sentence. This representation, combined with ACTG's pattern-matching capability, allows the statement of learning rules which focus on relative word order but ignore adjacency information. The resulting production rules omit words, just as the child does.

Representing Meaning

Adults conversing with a child almost invariably discuss recent or ongoing events, so that the child can associate some event with every sentence he hears. The language acquisition process does not consist solely of learning to produce or parse legal word combinations; it consists of learning the mapping between meanings and words. Accordingly, AMBER is presented not with isolated sentences as its data, but with sentence/meaning pairs.

AMBER represents the meaning of a sentence as a number of propositions, each incorporating one of a small set of relations. The most prevalent of these is the type relation, which connects tokens to the various concepts of which they are examples. There is no restriction on the number of type relations which may come off a token;

thus, the propositions (token-1 type red) and (token-1 type ball) state that the object token-1 is both red and a ball. Events are represented with relations such as agent, action, and object. The propositions (event-1 agent token-2), (event-1 action token-3), and (event-1 object token-4) represent an event with an agent, action, and object whose types have yet to be specified.

AMBER's representation makes a strong distinction between the main meaning of a sentence as expressed through its content words and the modulations of this meaning as expressed through its grammatical morphemes. The model assumes that a type relation pointing to a particular concept (e.g., *ball) is present for every content word (e.g., ball) found in the associated sentence. Moreover, a word-for relation is assumed present to establish the connection between word and concept. The presence of these two relations tells AMBER when a word contributes to the major meaning of a sentence.

Modulations on this meaning are represented by a different set of relations, such as number, time-of-action, possession, and so forth. Some of these relations connect tokens to various values, as in (token-1 number singular) and (token-2 time-of-action past). Others, as in (token-4 possesses token-5) and (token-5 in token-6), actually relate tokens.

AMBER's Initial Performance System

AMBER starts with the ability to produce single words in isolation. But even at this stage, the model draws on a set of general heuristics for generating utterances which will still be useful after its learning is complete. AMBER does not say words as soon as they come to mind; first there is an active planning stage during which sequential goals are set.

The model starts with rules for initializing and ending this planning phase, and for implementing its plans once they are complete (that is, actually saying the words in the planned order). The goals which result from the planning process look very like the data from which AMBER learns. The two-word utterance "Daddy ball" would be represented by the propositions (goal 1 AMBER pause), (goal 2 AMBER Daddy), (goal 3 AMBER ball), and (goal 4 AMBER pause), in which the model is the speaker.

At the outset, AMBER has only a single rule for inserting such goals in memory; stated in English for the sake of clarity and with its variables underlined, it is:

```
If you have no goals yet,
   and you see vtoken with type vtype,
   and vword is the word for vtype,
then set up a goal to pause,
   followed by a goal to say vword,
   followed by a goal to pause.
```

This rule separates the goal utterance from others by initial and final pauses. Thus, even though successive words may describe different aspects of the same event, they will be separated by noticeable gaps just as Bloom observed. Only after additional rules have been formed for inserting sounds between the initial word and the pauses can multi-word utterances begin to occur.

AMBER at Later Stages

On the basis of comparisons between sentences it hears and those it predicts, AMBER creates and modifies rules for saying multiple words at a time. These rules lead to the insertion of new goals between existing ones. Thus, they are dependent on the innate rules described above for initializing the goal insertion process and for carrying out goals once they have been set.

Imagine a situation in which AMBER sees Daddy bouncing a ball. Also suppose that the one-word rule we saw above happens to select "bounce" as the word that should be said. This would lead to three goals: (goal 1 AMBER pause), (goal 2 AMBER bounce), and (goal 3 AMBER pause). After some experience with English, the model will have generated a rule like:

```
If you have a goal to pause,
   followed by a goal to say vword2,
   and you have no intermediate goals,
   and vword2 is the word for vtype2,
   and vtoken2 is of type vtype2,
   and vtoken2 is the action of vevent,
   and vtoken1 is the agent of vevent,
   and vtoken1 is of type vtype1,
   and vword1 is the word for vtype1,
then insert a goal to say vword1
   between the other goals
```

This rule would add a goal to say the agent "Daddy" after the first pause and before "bounce", using the proposition (goal 1.5 AMBER Daddy). Similar rules lead to the production of two- and three-word sentences expressing the major relations described by Brown.

Later, AMBER also acquires rules for inserting grammatical morphemes. Since most grammatical morphemes are adjacent to the word whose meaning they modulate, they are generally inserted directly before or after the content word with which they occur. For example, a rule for regular pluralization might be stated:

```
If you have a goal to say vword,
   and vword is the word for vtype,
   and vtoken is of type vtype,
   and vtoken is the agent of vevent,
   and the number of vtoken is plural,
then insert a goal to say S
   directly after vword
```

This rule is specific to the agent of an event, but similar rules could be learned for objects and locations. Some morphemes express a relation between two content words, such as the prepositions "in" and "on" and the morpheme for possession. In these cases, the morpheme is inserted between the two related content words.

The Acquisition Process

For a system to learn from its mistakes, it must be able to compare its own actions to the desired ones, note the differences between them, and modify its behavior accordingly. In this section I describe AMBER's error correction mechanisms. First I examine the model's prediction mechanism and its relation to the goal structures mentioned earlier. Next I discuss AMBER's response to errors of omission, first for content words and then for grammatical morphemes. Finally, I consider errors of commission and the resulting call on the discrimination mechanism.

The Equivalence of Goals and Predictions

AMBER learns by comparing its predictions about what will be said in a given situation to what it actually hears. However, a learning system must do more than improve its ability to predict; it must also improve its ability to perform. AMBER accomplishes this by using the same productions for making predictions and for planning its speech acts. As we saw above, these rules add goal structures such as (goal 3 AMBER bounce) when AMBER is the speaker. When another person is the speaker, the resulting structures, such as (goal 3 Mommy bounce) if Mommy is the speaker, are treated as predictions instead of goals. Learning occurs only when someone else is speaking and the system is in prediction mode, while sentences are produced only in performance mode.

Correcting Content Word Omissions

AMBER's transition from the one-word to the multi-word stage is primarily due to the actions of a single learning heuristic. This rule applies when a content word is heard between two other words (or pauses) but was not predicted there; the result is a new performance rule for inserting analogous words in analogous positions in the future. AMBER knows enough about the nature of language to generalize across the particular words and concepts involved. However, it retains information about case relations and shared tokens (e.g., two of the words may have described aspects of the same object).

As an example, suppose AMBER sees Daddy bouncing a ball. The model predicts the one-word sentence "Daddy" (preceded and followed by pauses), while it actually hears "Daddy is bounce ing the ball" (again bounded by pauses). Since the grammatical morphemes "is", "ing", and "the" are not connected to concepts by word-for links, they are ignored by the current learning heuristic. However, the words "bounce" and "ball" each have associated concepts which occurred in the observed event. An insertion rule is created for each, the first inserting the action word after the agent word and before the final pause. The second is very similar, inserting the object word after the agent and before the pause.

These rules give AMBER the ability to generate agent-action and agent-object combinations, but no more. The new rules cannot cooperate to produce agent-action-object combinations, for once one of the rules has fired, the conditions of the other are no longer met. But once this has happened, the system can learn additional insertion rules, such as that for inserting the object word between the action and the final pause. Yet even after this has occurred, the performance rules are dependent on the selection of the agent word as the initial goal. Additional insertion rules must be learned to deal with cases in which the action or object is the first goal to be inserted.

Correcting Morpheme Omissions

As it is improving its ability to produce strings of content words, AMBER is also learning to insert grammatical morphemes. Some of the morphemes which modify a single token, such as tense and pluralization endings, occur after the words describing the token. Others, such as copulas (is, are, were) and articles (a, the), occur before the modified words. Separate learning heuristics are necessary for these two cases, but there forms are nearly identical.

These learning rules are evoked when a particular morpheme is heard before or after a content word, but was not predicted in that position. The result is a performance rule which inserts the morpheme either before or after words playing similar roles in the future (e.g., "ing" after the word for the action). AMBER knows that the particular content word is irrelevant; however, the case relation filled by that word and the morpheme are retained.

AMBER also knows that several content words may be used to describe the same object (e.g., "the big red ball s"), and that these words will occur together in any legal sentence. This is analogous to an assumption made by Anderson's LAS [3], which he has called the graph deformation condition. AMBER incorporates this assumption into its morpheme insertion rules, ensuring that a morpheme will be inserted either before the earliest or after the latest content word describing a token (thus, "big the red s ball" would never be produced).

The acquisition of relational morphemes, such as those expressing possession and location, is handled by a different rule. This heuristic is evoked in the same situations as the heuristic for content words, except that the unpredicted morpheme must not be associated with any concept via a word-for link. In addition, the objects described by the two correctly predicted words must be directly related (e.g., a token of milk is on a token of table). The resulting performance rule inserts the morpheme between the sets of words describing objects in the observed relation. Note that AMBER cannot acquire such relational morphemes until it can correctly predict the order of the words to be related.

Correcting Errors of Commission

Once AMBER has learned a number of rules for inserting goals, it can make a new sort of error: the model can incorrectly predict that a word will occur in a certain position. A single learning heuristic is sufficient to deal with all such errors of commission. Its condition is simple, but in addition to weakening the offending rule, its action calls on the discrimination mechanism to produce a more conservative variant.

To reiterate, this technique compares the last successful application of a rule to the more recent faulty application in the hope of finding additional conditions to constrain it in the future. Thus, if AMBER predicted the sequence "ball red" when "red ball" was heard, the discrimination process would be evoked. Comparing this case to an earlier one in which "blue block" was correctly predicted, AMBER would note that "blue" is a color while "ball" is not. Thus, the new rule would insert one word before another describing the same object only if the former were a color like "blue" or "red".

Although discrimination is useful in learning some content word orders, its major import lies with grammatical morphemes. Since the initial rules for nonrelational morphmemes are too general, their use quickly leads to wrong predictions. For instance, if the morpheme "ed" was incorrectly expected to follow an action word, AMBER would note that correct predictions of "ed" occurred only when the time of the action was past. Similarly, AMBER would quickly learn to insert "s" after the word for the agent only when its number was plural.

The conditions under which some morphemes are applied can be more complicated. Thus, the morpheme "were" is inserted before the action word only when the time of the action is past, and when the number of the agent associated with that action is plural. AMBER would be forced to learn these conditions in two stages, first creating a variant with one condition and later a version including both.

As a result, the more complex the conditions under which a morpheme occurs, the longer AMBER will take to master its use. If one equates the number of conditions with semantic complexity, then the discrimination process provides an elegant explanation of Brown's data on the order of acquisition for grammatical morphemes. Semantically more complex morphemes are mastered later because they require more conditions, and these conditions can be learned only one at a time.

Suggestions for Future Research

In summary, AMBER does a fine job of accounting for the major phenomena described at the beginning of the paper. However, the model makes a number of simplifying assumptions and stops improving after it has reached a certain level of expertise. In this section I suggest some directions in which AMBER should be extended.

Learning Word/Concept Associations

AMBER assumes that words and concepts are already connected through word-for links stored in long-term declarative memory. These connections play an important role in letting the system distinguish between content words and grammatical morphemes. At least some of these connections must be present before any ordering rules can be learned, but the model provides no explanation of their origin. Extending AMBER to let it make its own word/concept associations is clearly a direction for future work.

Selecting a Representation

AMBER relies heavily on the representational distinction between major meanings (expressed by type links) and modulations of those meanings (expressed by others). Unfortunately for the model, some languages express through content words what others express through grammatical morphemes. This suggests that the child does not start with a representation like AMBER's, though it may arrive at the same point as the result of experience with a particular language. Future research should consider how the child comes to treat some meanings as major and some as minor as a function of his native language.

Dealing With Exceptions

In its current version, AMBER cannot deal with irregular grammatical constructions. Some past forms, such as "ate", require a special word for past events, but this cannot be expressed in the current formalism for word/concept associations. Some plural forms require different endings than most, such as "oxen". These can be expressed in production form, but no conditions exist to distinguish these situations from the majority. Future versions of AMBER should have extended representations which address these issues.

Explaining Later Stages

AMBER's progression stops after it has mastered the grammatical morphemes. It never learns how to ask questions, or how to generate sentences with relative clauses. In fact, to present the model with other than simple declarative sentences would be an invitation to disaster. Future incarnations of the system should begin with the basic notions of recursion and transformation. Coupled with the existing learning mechanisms and the extensions discussed above, this should allow AMBER to progress far beyond its present level of expertise, and to become a true language user.

References

[1] Siklossy, L. Natural language learning by computer. In H. A. Simon and L. Siklossy (eds.), Representation and Meaning: Experiments with Information Processing Systems. Englewood Cliffs, N. J.: Prentice-Hall, 1972.

[2] Hedrick, C. Dissertation, Carnegie-Mellon University, 1974.

[3] Anderson, J. R. Induction of augmented transition networks. Cognitive Science, 1977, 1, 125-157.

[4] Selfridge, M. Dissertation, Yale University, 1979.

[5] Berwick, R. Dissertation, Massachusetts Institute of Technology, 1980.

[6] Bloom, L. One Word at a Time. The Hague: Mouton, 1976.

[7] Brown, R. A First Language: The Early Stages. Cambridge, Massachusetts: Harvard University Press, 1973.

Acknowledgements

This research was supported by Grants SPI-7914852 and IST-7918266 from the National Science Foundation. I would like to thank John R. Anderson for useful discussions which led to many of the ideas presented in this paper.