Slightly Ambiguous Triggers for Syntactic Parameter Setting

William Gregory Sakas (sakas@hunter.cuny.edu)
Hunter College and The Graduate Center, City University of New York

Janet Dean Fodor (jfodor@gc.cuny.edu)
The Graduate Center, City University of New York

Preview

It has been claimed that children rarely mis-set syntactic parameters (e.g. Wexler, 1998)

CAN DETERMINISTIC PARAMETER SETTING BE MODELLED?

Error-free parameter setting needs unambiguous triggers. Linguists have sought these, but they have proved elusive.

We will argue that:

a) no-error learning is over-optimistic (within psychologically realistic bounds); at least some trial-and-error learning is required;

b) nevertheless there are *almost-unambiguous* triggers; these contain useful structural information which an efficient learner should exploit.

An *almost-unambiguous* trigger is an input sentence compatible with few grammars in the domain of human languages.

A *fully* unambiguous trigger for parameter value Pv is an input sentence compatible only with grammars in which parameter P is set to value v.

How do current learning models deal with parametric ambiguity?

We compare three types of learners which exemplify differing approaches to trigger ambiguity.

a) Require unambiguous triggers; e.g. ***Waiting-STL***

b) Can use almost-unambiguous triggers to make informed guesses when fully unambiguous triggers are lacking; e.g. ***Guessing-STL***

c) Grammar guesses are tested against input sentences but are not input-triggered; ***e.g. TLA***

CUNY CoLAG Language Domain

13-parameters, 3,072 structurally distinct (finite) languages
(For details, see Sakas, ACL 2003)

1. Subject Initial
2. Object Final
3. Complementizer Initial
4. Obligatory Topic
5. Null Subject
6. Null Topic
7. Wh Movement
8. Pied Piping
9. Topic Marking
10. V to I Movement
11. I to C Movement
12. Affix hopping
13. Question Inversion

Access the languages and grammars online at http://146.95.2.133

Simulation procedure

1,000 learning trials (equivalent 1,000 children) per target grammar, per learning algorithm.

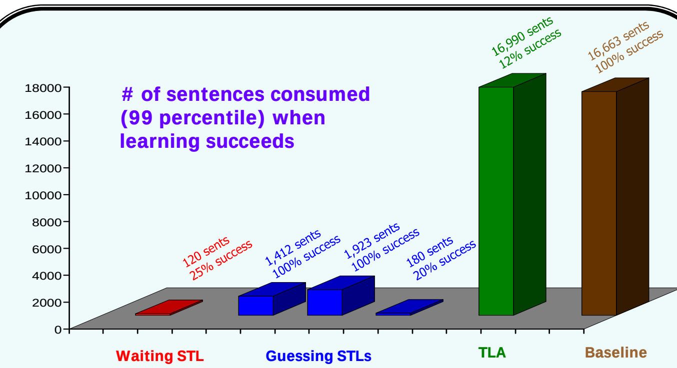
For each trial, measure how many sentences consumed by the learning model before attaining the target grammar.

More than 100,000 input sentences consumed = a 'time-out' failure

Subset Principle violations were excluded in advance.

1-parameter distant local maxima were excluded. All other local maxima were allowed to time-out.

Learning Efficiency Data	Failure Rate	# of sents needed for 99% of learners to attain target	mean # of sents needed for attaining target
Structural Triggers Learner (STL) : Waiting			
Waiting STL	75	120	28
Structural Triggers Learner (STL) : Guessing			
Minimal Chain	0	1,412	160
Local Attachment	0	1,923	197
Nearest Grammar	80	180	30
Triggering Learning Algorithm			
TLA	88	16,990	961
Baseline			
Random grammar choice	0	16,663	3,589



Conclusions

1) Waiting for unambiguous triggers is dangerously unreliable.
(The languages it fails on include 'English', 'Japanese', 'German'.)

2) Guessing is better than waiting. Use partly ambiguous triggers!

3) But it is critical *which* guessing strategy is employed.

• TLA guessing, which is not input-triggered, is no faster and much less reliable than random guessing. (As anticipated by Berwick and Niyogi, 1996)

• STL guessing does use partly ambiguous triggers. It is totally reliable, though a little slower, if guided by the parser rather than by domain-search tactics like nearest grammar (= local hill-climbing).

Learning Models

- All models are *error-driven* = if the current grammar can parse the sentence, retain it.
- The models differ with respect to what the learner does when its current grammar fails.
  - STLs select a grammar that can parse the current input sentence. (How can a psychologically realistic learner identify such a grammar? See Fodor, 1998b.)
  - The TLA selects a grammar similar to the current grammar, but adopts it only if it can parse the current input.

(a) Structural Triggers Learner: waiting model (Fodor, 1998a)

Parse input sentence with "supergrammar." The supergrammar contains all possible parameter values (in the form of structural *treelets;* see Fodor, 1998a, for details).

If the parser encounters a choice point, set no parameter values based on the rest of the sentence.

(b) Structural Triggers Learner: guessing models (Fodor, 1998b)

Parse input sentence with "supergrammar." The supergrammar contains all possible parameter values (in the form of structural *treelets;* see Fodor, 1998a, for details).

If the parser encounters a choice point it selects one grammar:

(b1) which assigns the input sentence the fewest empty categories.

(b2) which assigns the input sentence the most compact tree.

(b3) which differs from the current grammar by the least number of parameters.

(c) Triggering Learning Algorithm (Gibson & Wexler, 1994)

Change any one parameter. Try parsing the sentence. Adopt the grammar if the parse succeeds.

(d) Random choice (baseline)

Guess any grammar