The evaluation metric in generative grammar

John Goldsmith

December 9, 2011

1. What is a good theory?
2. Part One: Goodness of fit
   - Harwood
   - Generative grammar
3. Part Two: Grammar complexity and algorithmic complexity
4. Minimum Description Length analysis: Rissanen
5. Conclusion

Early motivation

Conclusions we wish to avoid

- Any analysis is as good as any other.
- It will often be the case that there are multiple best analyses, depending on what it is you are interested in studying or accomplishing.
- The analysis of one language should have only a very indirect effect on the analysis of another: and the only acceptable indirect effect is that what happens in one language could stimulate the linguist's imagination to come up with novel ways of treating data in another.

Part the First

Given a set of data d and a grammar g, g can be judged in two ways:

- How well does g *fit* the data? What is the goodness-of-fit of g to d? We would like to provide a measurement that is both explicit and rigorous.
- How good is G as an analysis at all, quite independent of the data?

Part the Second

How do we relate the two answers given in Part One? How can goodness of fit and grammar complexity be combined?

Find the right function—call it UG, if you'd like—such that

Role of evaluation metric

$$UG(g_1, g_2, d) = -UG(g_2, g_1, d)$$

and $UG(g_1, g_2, d) > 0$ iff g_1 is a better grammar for d than g_2 is.

The central question of linguistic theory will have been solved when UG has been established.

A further hypothesis

$$UG(g_1, g_2, d) = \Theta\left(m(g_1, g_2), n(g_1, d), n(g_2, d)\right) \quad (1)$$

m compares grammars;
n measures the goodness of fit of data and grammar;
Θ combines the factors appropriately.

MDL Minimum Description Length

MDL as UG as MDL...

$$UG_{MDL}(g_1, g_2, d) = |g_2| - |g_1| + log_2 \frac{pr_{g_1}(d)}{pr_{g_2}(d)}$$

In other words:

$$m(g_1, g_2) = |g_2| - |g_1| \quad (2)$$

$$n(g, d) = log_2\ pr_g(d) \quad (3)$$

$$\Theta(x, y, z) = x + (y - z) \quad (4)$$

Chomsky 1965: *Language and Mind*

A third task is that of determining just what it means for a hypothesis about the generative grammar of a language to be "consistent" with the data of sense. Notice that it is a great oversimplification to suppose that a child must discover a generative grammar that accounts for all the linguistic data that has been presented to him and that "projects" such data to an infinite range of potential sound-meaning relations....The third subtask, then, is to study what we might think of as the problem of "confirmation"— in this context, the problem of what relation must hold between a potential grammar and a set of data for this grammar to be confirmed as the actual theory of the language in question.

F. W. Harwood, *Axiomatic syntax: The construction and evaluation of a syntactic calculus. Language* 31:3 (1955)

"This paper discusses methods for presenting syntactic information in the form of a calculus, and for measuring its goodness of fit to a language. . . the aim of a syntactic system is to tell us how to put together the sequences of morphemes which are used as sentences in the language. Such directions we shall call the *formation rules.* Additionally, work on syntax usually give a certain amount of information about the equivalences between some sequences and others, e.g., that *John discovered the path = The path was discovered by John.* We shall call such statements *transformation rules.*"
–A confluence of two traditions, one from Rudolf Carnap and the other from Zellig Harris.

C_p	Set of category sequences of length p permitted by grammar
K_p	Set of word sequences from finite lexicon permitted by grammar
L_p	Set of all grammatical sentences of length p in L
U_p	Set of all sequences of length p of words from lexicon
$U_p - L_p$	Set of all ungrammatical sentences of length p
$U_p - K_k$	Set of all strings of length p predicted to be ungrammatical by grammar

[We can now] define a measure of the goodness of fit of a syntactic system S to a language L. We define the positive fit (F) . . . of a syntactic system s to a language L as:

$$F = \frac{|K_p \cap L_p|}{|L_p|} \tag{5}$$

[...Today this is called *recall*, in the context of computational linguistics...]

$$f = \frac{|(U_p - K_p) \cap (U_p - L_p)|}{|U_p - L_p|} \tag{6}$$

The ideal case is where $F = f = 1$, i.e. where s generates all and only the seqeunces in L_p. The S with $F = 1$, f=0 states that all of the N possible sequences may occur.

Harwood (2)

Harwood notes, "it is the general aim to have S small in relation to" the number of sentences in the language, both observed and predicted:

> *Compactness is an important and measurable feature, but we need to consider the effect on the goodness of fit of S to L when S is modified to increase its compactness. In particular there is little point in securing compactness at the expense of negative fit. How close an approximation to perfect fit we require will depend on our purposes, but it is always necessary to have an estimate of what the fit is, i.e. of the values of F and f . . . (411)*

Early generative account of UG

Universal Grammar

$$UG(g_1, g_2, d) = UG'\left(m(g_1, g_2), n(g_1, d), n(g_2, d)\right) \quad (7)$$

$$m(g_1, g_2) = |g_2| - |g_1| \quad (8)$$

$$n(g, d) = \begin{cases} 1 & \text{if } g \text{ satisfactorily generates } d; \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

$$UG(g_1, g_2, d) = \begin{cases} 1 & \text{if } n(g_1, d) = 1 \text{ and } n(g_2, d) = 0 \\ -1 & \text{if } n(g_1, d) = 0 \text{ and } n(g_2, d) = 1 \\ m(g_1, g_2) & \text{otherwise.} \end{cases} \quad (10)$$

Grammar complexity

A tradition beginning in the 1950s starting from work of

- Andrey Kolmogorov
- Ray Solomonoff
- Gregory Chaitin
- Jorma Rissanen

leading to a conclusion much like that of generative grammar:

Key:

The goal is to understand the principles of natural induction in terms of algorithmic length: here, grammar length.

Central points, in brief:

- There are an infinite number of grammars;
- *Given* that the grammars are expressed in a finite alphabet, they can be meaningfully ranked in terms of length (which is a stand-in for complexity)
- The specifics of how we do all this depend (but only to a certain degree) on the choice of the underlying 'machine';
- An analysis of data that will be evaluated by the length of the grammar, or program for a Turing machine, need not be written in machine code; it can be written in a higher order language for grammars, as long as we include a 'compiler' that turns grammars into machine code, and as long as we recognize that we have to take the length of the compiler into account as part of the complexity of the analysis.

Essence of MDL analysis

Given a set of data d, we seek the most likely hypothesis $\hat{g}$:

$$\begin{array}{lll} \hat{g} & = argmax_g & p(g|d) \\ & = argmax_g & p(d|g)\, p(g) \\ & = argmin_g & \underbrace{-log\, pr(d|g)} \qquad\qquad \underbrace{-log\, pr(g)} \end{array}$$

Goodness of fit Length of grammar in binary format

Essence of MDL analysis (2)

$$\text{DL (data, grammar)} = \text{Length(grammar)} + \log_2 \left(\frac{1}{pr(data|grammar)} \right) \quad (11)$$

The goal is to find the grammar that minimizes this expression: one that is both short and which fits the data as well as possible, with an appropriate trade-off between these two tendencies, always in opposition to each other.

Notice that the conclusion that Θ is simple addition follows from the definition of *information.*

Conclusion

Thanks for your attention today.