been achieved using the right productions. It is not clear at this stage whether this is important. Further, induction of natural language grammars may include additional constraints on evaluation, such as psychological plausibility and generative capacity. Metrics for these properties might prove subjective.

References

[1] T.C. Bell, J.G. Cleary, and I.H. Witten. *Text Compression*. Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[2] T.C. Bell, I.H. Witten, and J.G. Cleary. Modeling for text compression. *Computing Surveys*, 21(4):557–591, December 1989.

[3] R. C. Berwick and S. Pilato. Learning syntax by automata induction. *Machine Learning*, 2(1):9–38, 1987.

[4] D. Conklin and I.H. Witten. Complexity-based induction. *Machine Learning*, 16(3), in press.

[5] J. A. Feldman. Some decidability results on grammatical inference and complexity. AI Memo 93.1, Computer Science Dept., Stanford University, Stanford, California, 1970.

[6] E. M. Gold. Language identification in the limit. *Information Control*, 10:447–474, 1967.

[7] George W. Hart. *Minimum Information Estimation of Structure*. PhD thesis, MIT, Cambridge, MA, April 1987. LIDS-TH-1664.

[8] J. J. Horning. *A study of grammatical inference*. PhD thesis, Computer Science Dept., Stanford University, Stanford, California, 1969.

[9] D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proc. Institute of Electrical and Radio Engineers*, 40(9):1098–1101, September 1986.

[10] A. Moffat. Implementing the PPM data compression scheme. *IEEE Trans Communications*, COM-38(11):1917–1921, November 1990.

[11] C. G. Nevill-Manning, I. H. Witten, and D. L. Maulsby. Compression by induction of hierarchical grammars. *Proc Data Compression Conference*, edited by J.A. Storer and M. Cohn, pp. 244–253. IEEE Press, Los Alamitos, CA

[12] T. W. Pao and J. W. Carr. A solution of the syntactical induction-inference problem for regular languages. *Computer Languages*, 3:53–64, 1978.

[13] R. Solomonoff. A new method for discovering the grammars of phrase structure languages. *Information Processing*, pages 258–290, June 1959.

[14] I.H. Witten and J.G. Cleary. Inductive modeling for data compression. *Proc Fourth International Symposium on Modelling and Simulation Methodology*, January 1987.

mar outlined in the previous section, we might supply the generator with the following sequence of keys.

$$0\ 1$$

This directs the generator to select the first production (using a zero base) for the instantiation of S and subsequently the second production for S.

To generate the training sequence using the second grammar requires provision of a sequence of indices which allow the generator to select the correct words from the vocabulary and concatenate them into the appropriate strings. For example, the following vocabulary is garnered from the training set (note that the full-stop must be included in this grammar).

$$V = \{\textit{the . cat hugged dog kissed}\}$$

Using this we would supply the following key sequence to the generator.

$$0\ 2\ 3\ 0\ 4\ 1\ 4\ 5\ 0\ 2\ 1$$

To generate the training set using the third grammar requires the following key sequence.

$$0\ 1\ 1\ 1\ 0\ 0$$

It is important to note that the range of values for the keys depends not on the number of terminal symbols (i.e. words) contained in the language, but on the range of choices available to the grammar at each possible branch. Thus, for the first grammar, only one of two possible instantiations for S must be indicated: use the first production or the second. We can measure (in bits) the amount of information required for reconstruction as

$$\sum_{i=1}^{n} -\log_2 p_i$$

where n is the number of choices that must be made during production, and p_i is the probability of the desired instantiation being selected. For our explicit grammar example, two choices must be made: which rule for S to use first, and which second. If we assume that each production is equiprobable, then the amount of disambiguation information can be expressed as

$$-\log_2 1/2 - \log_2 1/2$$

.

More plainly, at each point of production we must decide whether to use the first rewrite rule or the second, and we must do this twice—two bits total. That is, we could use a "0" bit to indicate *select the first production* and a "1" bit to indicate *select the second*.

For the V* grammar, each instantiation has six possible alternatives. Once again (assuming equiprobability for each alternative) requires $11 \times -\log_2 1/6$, or about 29 bits. And finally, as all choices within the third grammar entail one of two possible alternatives, we would need $6 \times -\log_2 1/2$, or 6 bits to encode the disambiguation information.

The total complexity with respect to the training set is the combined value for the complexity of the grammar and the disambiguation information. The results obtained for the example grammars are outlined in Table 1.

grammar	measure of complexity	disambiguation information	total complexity
1	91.0	2.0	93.0
2	55.5	29.6	85.1
3	94.9	6.0	100.9

Table 1: Complexity values for the sample grammars.

As previously noted, possible instantiations are rarely equiprobable, and this fact can be used to improve the predictive power of each grammar and thus reduce the amount of disambiguation information required. Techniques such as Huffman coding [9] or arithmetic coding [2, 10, 14] give optimum encodings for such decisions.

Conclusions

The minimum goal of grammatical inference is to produce a compact grammar general enough to accept all well-formed expressions of a language without admitting those that are malformed. There appears to be no shortage of methods by which grammars may be inferred. The challenge is to establish a formal method for testing the effectiveness of the grammars obtained. We argue that the complexity of the grammar measured with respect to the set of expressions used to derive it is a rigorous and objective method of evaluation.

Two final comments. It is not entirely clear how disambiguation information applies to ambiguous grammars. That is, if more than one parse exists for a given expression, the information necessary to reconstruct that expression would not indicate if this had

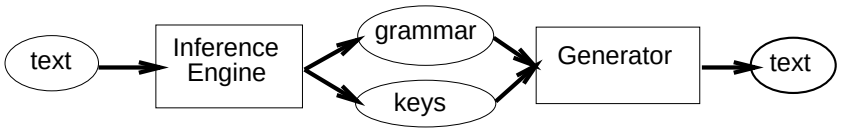

Figure 1: Inference/Generation process.

These three types of grammars are of different sizes, and, more importantly, their size grows in different ways as the corpus of text from which they are generated increases. The first includes a production for each (distinct) sentence in the corpus, and hence is approximately the same size as the corpus. The second is the smallest possible grammar, and works for any corpus of example strings independent of its size or structure. The third grammar is of intermediate size. In this very small example it is no smaller than the first one, but in a more realistic situation, with more example strings, the first grammar will be much larger. The third grammar will continue to grow as the corpus does because of the continued appearance of new syntactic constructions, though its rate of growth will steadily decline.

The size of a grammar can be quantified in different ways. One is to simply count the number of characters it contains. However, this is strongly affected by the number of characters used to express non-terminals—for instance, whether we write "NP" or "Noun-Phrase". To avoid this dependence on irrelevant details of representation, abstract identifiers can be used to denote non-terminals. Moreover, terminals can be placed in a lexicon—which will be the same for all grammars for a given corpus—and replaced in the grammar by pointers into the lexicon. This provides a "representation-independent" measure for the size of a grammar, which is essentially a count of the number of *symbols* (rather than characters) that it contains.

Unfortunately, this measure is distorted by the fact that common and uncommon symbols consume the same amount of space. For example, in the first grammar a word like "the" will appear more often than, say, "kissed"; similarly, in the third grammar, some non-terminals will be much more frequent than others. A more realistic measure of size can be obtained by taking into account the relative frequencies of the symbols, and noting that common symbols can be represented more efficiently than rare ones.

A straightforward way of doing this is to augment the lexicon, and the list of non-terminal symbols, with frequency counts that reflect how often they occur, and adding the number of bits required to code the symbol—that is, $-\log_2 p$ where p is its probability—each time it appears in the grammar. However, the need to store these frequency counts must be taken into account when calculating the amount of space occupied by the grammar. A more elegant way of achieving the same effect is to count symbols *adaptively* as they are encoded, so that the first time a symbol is seen it is encoded with a count of 1, the second time with a count of 2, and so on. This stratagem eliminates the need to record the counts explicitly. More details can be found in [1].

In practice, we find the size by translating the context-free grammar into a Prolog program and then use a general purpose program we have written for computing the complexity of Prolog programs. The complexity values obtained for the three example grammars are included in Table 1.

Disambiguation information

Grammars are used by parsers, generators, compilers, interpreters and a host of other language processing systems, both natural and artificial. A grammar is a generalisation of the structure of a language: this implies that additional information must be provided in any generative application. For example, if a particular proposition is to be presented in a specific surface form, the generator must be given disambiguation clues at each decision point of the grammar so that it can select appropriate instantiations. The amount of disambiguation information required for such processes is a measure of the utility of the grammar itself.

Consider a situation where we want to generate precisely the training sequence. We can view the Inference/Generation process as something like that shown in Figure 1. The inference engine derives a grammar (according to some specified algorithm) from the sample strings, producing with it a set of keys (or indices) that indicate the instantiations necessary for reproducing the training set.

To generate the training set using the *explicit* gram-

lations of the following principal components:

- The hypothesis space: the general set of rewrite rules that are to be considered—i.e. the candidate classes of grammars.
- The presentation criteria: in what order will examples be presented?—n.b. each string must eventually be seen.
- A criterion for success: when has the limit been reached?
- A measure of adequacy: the hypothesis must account for all sample strings plus all other strings generated by the grammar.
- Minimum complexity requirement: how tightly should the inferred grammar fit the target language?

The first three components pertain to the inference algorithm, to which much energy has been directed—such as Solomonoff's basic cycle detection method [13], implemented as Nevill-Manning's Hierarchical Grammar [11], which focuses on identifying the recursive features of a language; Horning's [8] enumerative technique for constructing context-free grammars; Pao's [12] finite-state machine reduction; or Berwick and Pilato's [3] k-reversible language technique.

Feldman's other two principles pertain to testing the quality of the inferred grammar. While the need to address these aspects of the formulation are generally acknowledged, little has been done to develop formal methods by which they can be measured.

Quality metrics for inferred grammars

In this paper we describe two methods for obtaining objective measures of the quality of inferred grammars. This work parallels more general work on the evaluation of theories using complexity-based induction [4]. The first measure, developed along the lines of Hart's [7] *Minimum Information* (MI) estimator, seeks to calculate the complexity of the grammar in terms of the information content of its formalism. Context-free grammars are translated into Prolog programs and analysed by a single general estimator.

Useful grammars must be able to account for strings other than those within the training set, but *without* admitting strings not in the language of the source grammar. Therefore a second objective metric is described to measure the amount of disambiguation information necessary for the grammar to generate exactly the training set.

Complexity

To illustrate how a complexity metric may be applied to an inferred grammar, consider the following sample strings.

The cat kissed the dog.

The dog hugged the cat.

The most specific grammar that we can infer for this training set would simply provide production rules for each individual string, as with

$$\begin{array}{rcl} S & \Rightarrow & \textit{The cat kissed the dog}. \\ S & \Rightarrow & \textit{The dog hugged the cat}. \end{array}$$

This grammar makes strong predictions about strings it has already seen, but its capacity for generalisation is very weak—non-existent, in fact, for it cannot predict any unseen expression.

In contrast, the most general grammar we can infer from the training set would be

$$\begin{array}{rcl} S & \Rightarrow & v\ S \\ v & \in & V \end{array}$$

where V is the vocabulary demonstrated in the sample expressions. This V* (*vee-star*) grammar is certainly capable of predicting all sentences of the source language. In fact, it accepts any arbitrary sequence of words from the vocabulary of the language, and consequently obviates the notion of grammaticality through its failure to reject malformed expressions.

In between these extremes are grammars that cover the training set *and* predict unseen well-formed expressions *and* reject sentences not in the source language. For example,

$$\begin{array}{rcl} \textit{S} & \Rightarrow & \textit{NP VP} \\ \textit{NP} & \Rightarrow & \textit{The N} \\ \textit{N} & \Rightarrow & \textit{cat} \mid\mid \textit{dog} \\ \textit{VP} & \Rightarrow & \textit{V NP} \\ \textit{V} & \Rightarrow & \textit{hugged} \mid\mid \textit{kissed} \end{array}$$

accounts for the training set, but is also able to predict the sentence *The dog kissed the cat*, and a variety of other unseen sentences (hypothetically) contained in the language. Equally important, this grammar will also reject presumably malformed sentences like *The dog cat the dog*.

Objective Evaluation of Inferred Context-Free Grammars

Tony C. Smith Ian H. Witten John Cleary Shane Legg

Department of Computer Science, University of Waikato, Hamilton, New Zealand
Email tcs@waikato.ac.NZ; phone: +64 (7) 838–4453; fax: +64 (7) 838–4155

July, 1994

An infinite number of context-free grammars may be inferred from a given training set. The defensibility of any single grammar hinges on the ability to compare that grammar against others in a meaningful way. In keeping with the Minimum Description Length Principle, smaller grammars are preferred over larger ones, but only insofar as the smaller grammar does not overgeneralise the language being studied. Furthermore, measures of size must incorporate the grammar's ability to cover sentences of the source language not included in the training set.

This paper describes a method for evaluating the quality of context-free grammars according to i) the complexity of each grammar and ii) the amount of disambiguation information necessary for each grammar to reproduce the training set. The sum of the two evaluations is used as an objective measure of a grammars information content. Three grammars are used as examples of this process.

Introduction

Grammatical inference, stated in its simplest form, is the discovery of an acceptable grammar for a language based on a finite set of sample strings constructed from a finite alphabet. The induction process requires that the inference device be presented with a growing corpus of example strings from the grammar being inferred. At each presentation, the device must simultaneously make guesses of the underlying rule being exemplified. By recognizing regularities within the sample set, the inference mechanism must be able to form a generalisation of the data that will permit the prediction of future data. In a keystone paper on grammatical inference [6], Gold identified three possible results that can be expected from this approach, which can be formulated into a rough criterion for success:

1. The hypothesis will converge to a single description that correctly identifies the grammar—in which case the inference is correct.
2. The hypothesis will oscillate indefinitely—which implies that the inference has failed.
3. The hypothesis will converge to an incorrect description of the grammar—in which case the inference is incorrect.

The important aspect of this formulation is the idea of convergence. Assuming that the induction is tending towards a type 1 hypothesis, the inferencing mechanism will move ever closer to a correct description of the grammar—the point in time when the grammar may be considered as *identified in the limit*. The notion of convergence implies that at no point in the inductive process can the device assert that the grammar it has inferred is correct, since further evidence may prove it incorrect. In practice, the mechanism can only detect a point in time when its hypothesis has not been changed for a significant number of consecutive sample strings—a point when it may be deemed to have reached a sufficiently high probability of correctness.

Constructing a correct grammar, however, is really only half the battle. Once a grammar has been hypothesized, the challenge becomes one of assessing its virtue. That is, given that there exists an infinite number of grammars that will cover (or generate) the training sequence, how do we defend any claim that one is superior to another?

Grammatical formulations

Feldman [5] suggested that attempts to formalize grammatical inference must include precise formu-