THE RECOGNITION CAPACITY OF LOCAL SYNTACTIC CONSTRAINTS

Mori Rimon[1]
Jacky Herz[2]

The Computer Science Department
The Hebrew University of Jerusalem,
Giv'at Ram, Jerusalem 91904, ISRAEL
E-mail: rimon@hujics.BITNET

Abstract

Given a grammar for a language, it is possible to create finite state mechanisms that approximate its recognition capacity. These simple automata consider only short context information, drawn from local syntactic constraints which the grammar imposes. While it is short of providing the strong generative capacity of the grammar, such an approximation is useful for removing most word tagging ambiguities, identifying many cases of ill-formed input, and assisting efficiently in other natural language processing tasks. Our basic approach to the acquisition and usage of local syntactic constraints was presented elsewhere; in this paper we present some formal and empirical results pertaining to properties of the approximating automata.

1. Introduction

Parsing is a process by which an input sentence is not only recognized as belonging to the language, but is also assigned a structure. As [Berwick/Weinberg 84] comment, recognition per se (i.e. a weak generative capacity analysis) is not of much value for a theory of language understanding, but it can be useful "as a diagnostic". We claim that if an efficient recognition procedure is available, it can be most valuable as a pre-parsing reducer of lexical ambiguity (especially, as [Milne 86] points out, for deterministic parsers), and even more useful in applications where full parsing is not absolutely required - e.g. identification of ill-formed inputs in a text critique program. Still weaker than recognition procedures are methods which approximate the recognition capacity. This is the kind of methods that we discuss in this paper.

More specifically, we analyze the recognition capacity of automata based on local (short context) considerations. In [Herz/Rimon 91] we presented our approach to the acquisition and usage of local syntactic constraints, focusing on its use for reduction of word-level ambiguity. After briefly reviewing this method in section 2 below, we examine in more detail various characteristics of the approximating automata, and suggest several applications.

2. Background: Local Syntactic Constraints

Let S = $W_1, ..., W_N$ be a sentence of length N, {Wi} being the words composing the sentence. And let $t_1, ..., t_M$ be a tag image corresponding to the sentence S, {ti} belonging to the tag set T - the set of word-class tags used as terminal symbols in a given grammar G. Typically, $M=N$, but in a more general environment we allow $M \geq N$. This is useful when dealing with languages where morphology allows cliticization, concatenation of conjunctions, prepositions, or determiners to a verb or a noun, etc.; in grammars for Hebrew, for example, it is convenient

[1] M. Rimon's main affiliation is the IBM Scientific Center, Haifa, Israel, E-mail: rimon@haifasc3.iinus1.ibm.com

[2] J. Herz was partly supported by the Leibniz Center for Research in Computer Science, the Hebrew University, and by the Rau foundation of the Open University.

to assume that a preliminary morphological phase separated word-forms to basic sequences of tags, and then state syntactic rules in terms of standard word classes.

In any case, it is reasonable to assume that the tag image $t_1,\dots, t_M$ cannot be uniquely assigned. Even with a coarse tag set (e.g. parts of speech with no features) many words have more than one interpretation, thus giving rise to exponentially many tag images for a sentence.[3]

Following [Karlsson 90], we use the term *cohort* to refer to the set of lexically valid readings of a given word. We use the term *path* to refer to a sequence of M tags ($M \geq N$) which is a tag-image corresponding to the words $W_1,\dots, W_N$ of a given sentence S. This is motivated by a view of lexical ambiguity as a graph problem: we try to reduce the number of tentative paths in ambiguous cases by removing arcs from the Sentence Graph (SG) - a directed graph with vertices for all tags in all cohorts of the words in the given sentence, and arcs connecting each tag to all tags in the cohort which follows it.

The removal of arcs and the testing of paths for validity as complete sentence interpretations are done using local constraints. A local constraint of length k on a given tag t is a rule allowing or disallowing a sequence of k tags from being in its right (or left) neighborhood in any tag image of a sentence. In our approach, the local constraints are extracted from the grammar (and this is the major aspect distinguishing it from some other short context methods such as [Beale 88], [DeRose 88], [Karlsson 90], [Katz 85], [Marcus 80], [Marshall 83], and [Milne 86]).

For technical convenience we add the symbol "$<" at the beginning of tag images and ">$" at the end. Given a grammar G (which for the time being we assume to be an unrestricted context-free phrase structure grammar), with a set T of terminal symbols (tag set), a set V of variables (non-terminals, among which S is the root variable for derivations), and a set P of production rules of the form $A \rightarrow \alpha$, where A is in V and α is in $(V \cup T)^*$, we define the Right Short Context of length k of a terminal t (tag):

SCr (t,k) for t in T and for k = 0,1,2,3...

$$= \left\{ tz \;\middle|\; \begin{array}{l} z \in T^*,\ |z|=k \text{ or } |z|<k \text{ if} \\ \text{'>\$' is the last tag in } z, \\ \text{and there exists a derivation} \\ S \overset{*}{\Rightarrow} \alpha t z \beta \ \ (\alpha, \beta \in (V \cup T)^*) \end{array} \right\}$$

The Left Short Context of length k of a tag t relative to the grammar G is denoted by SCl (t,k) and defined in a similar way.

It is sometimes useful to define *Positional* Short Contexts. The definition is similar to the above, with a restriction that t may start only in a given position in a tag image of a sentence.

The basis for the automaton which checks a tag stream (path) for validity as a tag-image relative to the local constraints, is the function next(t), which for any t in T defines a set, as follows:

$$\text{next}(t) = \{ z \mid tz \in \text{SCr}(t,1) \}$$

In [Herz/Rimon 91] we gave a procedure for computing next(t) from a given context free grammar, using standard practices of parsing of formal languages (see [Aho/Ullman 72]).

3. Local Constraints Automata

We denote by LCA(1) the simple finite state automaton which uses the pre-processed {next(t)} sets to check if a given tag stream (path) satisfies the SCr(t,1) constraints.

In a similar manner it is possible to define LCA(k), relative to the short context of length k.

We denote by L the language generated by the

[3] Our studies of modern written Hebrew suggest that about 60% of the word-forms in running texts are ambiguous with respect to a basic tag set, and the average number of possible readings of such word-forms is 2.4. Even when counting only "natural readings", i.e. interpretations which are likely to occur in typical corpora, this number is quite large, around 1.8 (it is somewhat larger for the small subset of the most common words).

underlying grammar, and by L(k) the language accepted by the automaton LCA(k). The following relations hold for the family of automata {LCA(i)}:

$$L(1) \supseteq L(2) \supseteq \dots \supseteq L$$

This guarantees a security feature: If for some i, LCA(i) does not recognize (accept) a string of tags, then this string is sure to be illegal (i.e. not in L). On the other hand, any LCA(k) may recognize sentences not in L (or, from a dual point of view, will reject only part of the illegal tag images). The important question is how tight are the inclusion relations above - i.e. how well LCA(k) approximates the language L. In particular we are interested in LCA(1).

There is no simple analytic answer to this question. Contradictory forces play here: the nature of the language -- e.g a rigid word order and constituent order yield stronger constraints; the grain of the tag set -- better refined tags (different languages may require different tag sets) help express refined syntactic claims, hence more specific constraints, but they also create a greater level of tagging ambiguity; the size of the grammar -- a larger grammar offers more information, but, covering a richer set of structures, it allows more tag-pairs to co-occur; etc.

It is interesting to note that for Hebrew, short context methods are most needed because of the considerable ambiguity at the lexical level, but their effectiveness suffers from the rather free word/constituent order.

Finally, a comment about the computational efficiency of the LCA(k) automaton. The time complexity of checking a tag string of length n using LCA(k) is at most $O(n \times k \times \log|T|)$, while a non-deterministic parser for a context free grammar may require $O(n^3 \times |G|^2)$. ($|T|$ is the size of the tag set, $|G|$ is the size of the grammar). The space complexity of LCA(k) is proportional to $|T|^{k+1}$; this is why only truly short contexts should be used.

Note that for a sentence of length k, the power of LCA(k) is identical to the weak generative capacity of the full underlying grammar. But since the size of sentences (tag sequences) in L is unbounded, there is no fixed k which suffices.

4. A Sample Grammar

To illustrate claims made in the sections below, we will use the following toy grammar of a small fragment of English. Statements about the correctness of sentences etc., are of course relative to this toy grammar.

The tag set T includes: n (noun), v (verb), det (determiner), adj (adjective) and prep (preposition). The context free grammar G is:

```
S  --> $< NP VP >$
NP --> (det) (adj) n
NP --> NP PP
PP --> prep NP
VP --> v NP
VP --> VP PP
```

To extract the local constraints from this grammar, we first compute the function next(t) for every tag t in T, and from the resulting sets we obtain the graph below, showing valid pairs in the short context of length 1 (again, validity is relative to the given toy grammar):

$$\begin{tikzcd}
\$< \arrow[drr] \arrow[ddrr] \arrow[dddrr] & & & & \\
 & & det \arrow[d] \arrow[dd, bend left] & & \\
v \arrow[urr] \arrow[rr] \arrow[drr, bend right] & & adj \arrow[d] & & prep \arrow[ull, bend right] \arrow[ll] \arrow[dll] \\
 & & n \arrow[urr] \arrow[ull, bend left] \arrow[drr] & & \\
 & & & & >\$
\end{tikzcd}$$

This graph, or more conveniently the table of "valid neighbors" below, define the LCA(1) automaton. The table is actually the union of the SCr(t,1) sets for all t in T, and it is derived directly from the graph:

$<	det	adj	n	prep	adj
$<	adj	v	det	prep	n
$<	n	v	adj	n	prep
det	adj	v	n	n	v
det	n	prep	det	n	>$

5. A "Lucky Bag" Experiment

Consider the following sentence, which is in the language generated by grammar G of section 4:

(1) The charming princess kissed a frog.

The unique tag image corresponding to this sentence is: [$<, det, adj, n, v, det, n, >$].

Now let us look at the 720 "random inputs" generated by permutations of the six words in (1), and the set of corresponding tag images. Applying LCA(1), only two tag images are recognized as valid: [$<, det, adj, n, v, det, n, >$], and [$<, det, n, v, det, adj, n, >$]. These are exactly the images corresponding to the eight syntactically correct sentences (relative to G),

(1a-b) The/a charming princess kissed a/the frog.
(1c-d) The/a charming frog kissed a/the princess.
(1e-f) The/a princess kissed a/the charming frog.
(1g-h) The/a frog kissed a/the charming princess.

This result is not surprising, given the simple sentence and toy grammar. (In general, a grammar with a small number of rules relative to the size of the tag set cannot produce too many valid short contexts). It is therefore interesting to examine another example, where each word is associated with a cohort of several interpretations. We borrow from [Herz/Rimon 91]:

(2) All old people like books about fish.

Assuming the word tagging shown in section 6, there are 256 ($2 \times 2 \times 2 \times 4 \times 2 \times 2 \times 2$) tentative tag images (paths) for this sentence and for each of its 5040 permutations. This generates a very large number of rather random tag images. Applying LCA(1), only a small number of images are recognized as potentially valid. Among them are syntactically correct sentences such as:

(2a) Fish like old books about all people.

and only less than 0.1% sentences which are locally valid but globally incorrect, such as:

(2b) * Old fish all about books like people.

(tagged as [$<, n, v, n, prep, n, v, n, >$]).

These two examples do not suggest any kind of proof, but they well illustrate the recognition power of even the least powerful automaton in the {LCA(i)} family. To get another point of view, one may consider the simple formal language L consisting of the strings $\{a^m b^m\}$ for $1 \leq m$, which can be generated by a context-free grammar G over T = {a, b}. LCA(1) based on G will recognize all strings of the form $\{a^j b^k\}$ for $1 \leq j,k$, but none of the very many other strings over T. It can be shown that, given arbitrary strings of length n over T, the probability that LCA(1) will not reject strings not belonging to L is proportional to $n/2^n$, a term which tends rapidly to 0. This is the over-recognition margin.

6. Use of LCA in Conjunction with a Parser

The number of potentially valid tag images (paths) for a given sentence can be exponential in the length of the sentence if all words are ambiguous. It is therefore desirable to filter out invalid tag images before (or during) parsing.

To examine the power of LCAs as a pre-parsing filter, we use example (2) again, demonstrating lexical ambiguities as shown in the chart below. The chart shows the Reduced Sentence Graph (RSG) - the original SG from which invalid arcs (relative to the SCr(t,1) table) were removed.

ALL OLD PEOPLE LIKE BOOKS ABOUT FISH

$$\begin{tikzcd}
 & \text{det} \arrow[r] \arrow[d] & \text{adj} \arrow[r] & \text{n} \arrow[r] \arrow[dr] & \text{v} \arrow[r] & \text{n} \arrow[r] & \text{prep} \arrow[r] & \text{n} \arrow[dr] & \\
\$< \arrow[ur] \arrow[r] & \text{n} & \text{n} \arrow[r] & \text{v} \arrow[dr] \arrow[ddr] & \text{prep} \arrow[ur] & \text{v} \arrow[r] & \text{adj} \arrow[ur] & \text{v} & >\$ \\
 & & & & \text{adj} \arrow[uur] & & & & \\
 & & & & \text{n} \arrow[uur] & & & &
\end{tikzcd}$$

We are left with four valid paths through the sentence, out of the 256 tentative paths in SG. Two paths represent legal syntactic interpretations (of which one is "the intended" meaning). The other two are locally valid but globally incorrect, having either two verbs or no verb at all, in contrast to the grammar. SCr(t,2) would have rejected one of the wrong two.

Note that in this particular example the method was quite effective in reducing sentence-wide interpretations (leaving an easy job even for a deterministic parser), but it was not very good in individual word tagging disambiguation. These two sub-goals of tagging disambiguation - reducing the number of paths and reducing word-level possibilities - are not identical. It is possible to construct sentences in which all words are two-way ambiguous and only two disjoint paths out of the 2^N possible paths are legal, thus preserving all word-level ambiguity.

We demonstrated the potential of efficient path reduction for a pre-parsing filter. But short-context techniques can also be integrated into the parsing process itself. In this mode, when the parser hypothesizes the existence of a constituent, it will first check if local constraints do not rule out that hypothesis. In the example above, a more sophisticated method could have used the fact that our grammar does not allow verbs in constituents other than VP, or that it requires one and only one verb in the whole sentence. The motivation for this method, and its principles of operation, are similar to those behind different techniques combining top-down and bottom-up considerations. The performance gains depend on the parsing technique; in general, allowing early decisions regarding inconsistent tag assignments, based on information which may be only implicit in the grammar, offers considerable savings.

7. Educated Guess of Unknown Words

Another interesting aid which local syntactic constraints can provide for practical parsers is "an oracle" which makes "educated guesses" about unknown words. It is typical for language analysis systems to assume a noun whenever an unknown word is encountered. There is sense in this strategy, but the use of LCA, even LCA(1), can do much better.

To illustrate this feature, we go back to the princess and the frog. Suppose that an adjective unknown to the system, say "Transylvanian" was used rather than "charming" in example (1), yielding the input sentence:

(3) The Transylvanian princess kissed a frog.

Checking out all tags in T in the second position of the tag image of this sentence, the only tag that satisfies the constraints of LCA(1) is *adj*.

8. "Context Sensitive" Spelling Verification

A related application of local syntactic constraints is spelling verification beyond the basic word level (which is, in fact, SCr(t,0)).

Suppose that while typing sentence (1), a user made a typing error and instead of the adjective "charming" wrote "charm" (or "arming", or any other legal word which is interpreted as a noun):

(4) The charm princess kissed a frog.

This is the kind of errors[4] that a full parser would recognize but a word-based spell-checker would not. But in many such cases there is no need for the full power (and complexity) of a parser; even LCA(1) can detect the error. In general, an LCA which is based on a detailed grammar, offers cheap and effective means for invalidation of a large set of ill-formed inputs.

Here too, one may want to get another point of view by considering the simple formal language L = $\{a^m b^m\}$. A single typo results in a string with one "a" changed for a "b", or vice versa. Since LCA(1) recognizes strings of the form $\{a^j b^k\}$ for $1 \leq j,k$, given arbitrary strings of length n over T = {a, b}, LCA(1) will detect all but two of the n single typos possible - those on the borderline between the a's and b's.

[4] Remember that everything is relative to the toy grammar used throughout this paper. Hence, although "the charm princess" may be a perfect noun phrase, it is illegal relative to our grammar.

9. Assistance to Tagging Systems

Tagged corpora are important resources for many applications. Since manual tagging is a slow and expensive process, it is a common approach to try automatic heuristics and resort to user interaction only when there is no decisive information. A well-built tagging system can "learn" and improve its performance as more text is processed (e.g. by using the already tagged corpus as a statistical knowledge base).

Arguments such as those given in sections 7 and 8 above suggest that the use of local constraints can resolve many tagging ambiguities, thus increasing the "speed of convergence" of an automatic tagging system. This seems to be true even for the rather simple and inexpensive LCA(1) for languages with a relatively rigid word order. For related work cf. [Greene/Rubin 71], [Church 88], [DeRose 88], and [Marshall 83].

10. Final Remarks

To make our presentation simpler, we have limited the discussion to straightforward context free grammars. But the method is more general. It can, for example, be extended to CFGs augmented with conditional equations on features (such as agreement) - either by translating such grammars to equivalent CFGs with a more detailed tag set (assuming a finite range of feature values), or by augmenting our automata with conditions on arcs. It can also be extended for a probabilistic language model, generating probabilistic constraints on tag sequences from a probabilistic CFG (such as of [Fujisaki et al. 89]).

Perhaps more interestingly, the method can be used even without an underlying grammar, if a large corpus and a lexical analyzer (which suggests pre-disambiguated cohorts) are available. This variant is based on a technique of invalidation of tag pairs (or longer sequences) which satisfy certain conditions over the whole language L, and the fact that L can be approximated by a large corpus. We cannot elaborate on this extension here.

References

[Aho/Ullman 72] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation and Compiling*. Prentice-Hall, 1972-3.

[Beale 88] Andrew David Beale. Lexicon and Grammar in Probabilistic Tagging of Written English. Proc. of *the 26th Annual Meeting of the ACL*, Buffalo NY, 1988.

[Berwick/Weinberg 84] Robert C. Berwick and Amy S. Weinberg. *The Grammatical Basis of Linguistic Performance*, The MIT Press, 1984.

[Church 88] Kenneth W. Church. A Stochastic Parts Program and Noun Phrase Parser for Running Text. Proc. of *the 2nd ACL conf. on Applied Natural Language Processing*. 1988.

[DeRose 88] Steven J. DeRose. Grammatical Category Disambiguation by Statistical Optimization. *Computational Linguistics*, vol. 14, no. 1, 1988.

[Fujisaki et al. 89] T. Fujisaki, F. Jelinek, J. Cocke, E. Black, T. Nishimo. A Probabilistic Parsing Method for Sentence Disambiguation. Proc. of *the 1st International Parsing Workshop*, Pittsburgh, June 1989.

[Greene/Rubin 71] Barbara Greene and Gerald Rubin. Automated Grammatical Tagging of English. *Technical Report*, Brown University, 1971.

[Herz/Rimon 91] Jacky Herz and Mori Rimon. Local Syntactic Constraints. Proc. of *the 2nd International Workshop on Parsing Technologies*, Cancun, February 1991.

[Karlsson 90] Fred Karlsson. Constraint Grammar as a Framework for Parsing Running Text. The 13th COLING Conference, Helsinki, 1990.

[Katz 85] Slava Katz. Recursive M-gram Language Model via Smoothing of Turing Formula. *IBM Technical Disclosure Bulletin*, 1985.

[Marcus 80] Mitchell P. Marcus. *A Theory of Syntactic Recognition for Natural Language*, The MIT Press, 1980.

[Marshall 83] Ian Marshall. Choice of Grammatical Word-Class Without Global Syntactic Analysis: Tagging Words in the LOB Corpus. *Computers in the Humanities*, vol. 17, pp. 139-150, 1983.

[Milne 86] Robert Milne. Resolving Lexical Ambiguity in a Deterministic Parser. *Computational Linguistics*, vol. 12, no. 1, pp. 1-12, 1986.