LOCAL SYNTACTIC CONSTRAINTS

Jacky Herz and Mori Rimon ([1]

The Computer Science Department
The Hebrew University of Jerusalem,
Giv'at Ram, Jerusalem 91904, ISRAEL

ABSTRACT

A method to reduce ambiguity at the level of word tagging, on the basis of local syntactic constraints, is described. Such "short context" constraints are easy to process and can remove most of the ambiguity at that level, which is otherwise a source of great difficulty for parsers and other applications in certain natural languages. The use of local constraints is also very effective for quick invalidation of a large set of ill-formed inputs. While in some approaches local constraints are defined manually or discovered by processing of large corpora, we extract them directly from a grammar (typically context free) of the given language. We focus on deterministic constraints, but later extend the method for a probabilistic language model.

1. Introduction: Local Constraints and their Use

Let S = W_1,..., W_N be a sentence of length N, {Wi} being the words composing the sentence. Ideally, a lexical-morphological analyzer can assign to each word W_i a unique tag t_i, expressing its grammatical characteristics (typically part of speech and features). The unique tag image t_1,..., t_N of S could then serve as input to NLP applications, including - but not limited to - parsing.

In reality, however, W_i may have more than one interpretation, hence t_i is not uniquely defined. Examples for ambiguity at this level in English are nouns (both in singular and in plural forms) which can be often interpreted at word-level as verbs; words ending with "ing" which are ambiguous between tentative readings as a progressive verb, a gerund and an adjective; etc. Hebrew, our main language of study, poses a much greater difficulty, because of the complexity of its morpho-syntax and the "terse" nature of the vowel-free writing system. In modern written Hebrew, nearly 60% of the words in running texts are ambiguous with respect to tagging, and the average number of possible readings of words in a running text is found to to be 2.4 (See [Francis 82] for data on English).[2] In addition, in many cases the morphological analysis of a Hebrew word yields a sequence of tags rather than a single tag, and different interpretations may be mapped to sequences of different lengths (similar phenomena may be found in other Semitic languages and in Romance languages where cliticization occurs). This is in fact a different order of the ambiguity issue. Consider as an example the written character string VRD (ורד), which can be interpreted in Hebrew as:

```
     [ Noun ]  ("vered" = a rose)
or:  [ Adj ]   ("varod" = rosy)
or:  [ Conj, Verb ] ("v-red" = and descend).
```

We will refer to a sequence of M tags ($M \geq N$) , which is a legal (per word) tag image corresponding to the sentence S = W_1,..., W_N , as a path. The number of potentially valid paths can

---

[1] The first author is also affiliated with the Open University.
The second author's main affiliation is the IBM Scientific Center, Haifa, Israel.
Please address e-mail correspondence to: rimon@hujics.BITNET rimon@haifasc3.IINUS1.IBM.COM

[2] The degree of ambiguity is obviously affected by the grain of the tagging system (the level of detail of the tag set).

be exponential in the length of the sentence if all words are ambiguous. A parser will reduce this number to the minimum feasible. But we are interested in quicker, even if not perfect methods to reduce the number of valid paths and word-level ambiguity.[3]

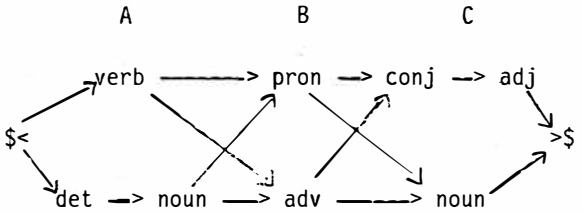
This paper describes a method to reduce tagging ambiguity, based on local syntactic constraints. A local constraint of length k on a given tag t is a rule disallowing a sequence of k tags from being in the Short Context of t. Intuitively, a Short Context with length k of a tag t in a given sentence S, denoted by SC(t,k), is its right or left tag environment. Before giving the formal definitions, let us mention that short context methods of one form or another are not new. They can be found in papers such as [Beale 88], [Choucka 85], [DeRose 88], [Katz 85], [Lozinskii 86], [Marcus 80], [Marshall 83] - to name a few. Our approach differs in various aspects, but mainly in the manner by which short context constrains are defined and identified. In the next chapter we will show how the constraints are retrieved directly from a grammar of the language, establishing a finite state mechanism which approximates the grammar.

To start with a more formal treatment of the short context notion, let us first add to the sentence S two special "words": "$<", denoting "Start" as the beginning of sentence marker, and ">$", denoting "End" at the end of sentence. These markers are also added to the tag image of the sentence.

We can now look at the resolution of ambiguity as a graph searching problem. As an example, suppose we have a sentence with three words, A B C, and assume that the initial tagging output of the lexical analyzer is the following (rather unlikely for English, but quite realistic for Hebrew):

```
for A:  [ verb ] or [ det, noun ]
for B:  [ pron ] or [ adv ]
for C:  [ conj, adj ] or [ noun ]
```

Then we can look at SG, the Sentence Graph, which is a directed graph where arcs represent all a-priori possible local paths:



Every path from "$<" to ">$" represents a possible interpretation of S as a stream of tags. Note that invalidating even a small number of arcs from SG reduces rapidly the number of possible paths.

As said above, we use local constraints to remove invalid arcs, and to finally arrive at the Reduced Sentence Graph.

Let T be the set of all possible tags - the tag set. The Right Short Context of length n of a tag t is defined by:

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

  { tz | z is in T* ,                          }
= {    |z| = n  or                             }
  {    |z| < n if ">$" is the last tag in z,   }
  {    and tz is a valid sequence of tags      }
```

The Left Short Context of length n of a tag t is denoted by SCl (t,n), and is defined in a symmetric way.

The definition of "validity" of tag sequences can vary. In our approach validity will be relative to a given formal grammar of the language, not to independent linguistic intuitions. This will be elaborated in the next chapter.

The Right (or Left) Positional Short Context i is the same as SCr (t,n) (or SCl (t,n)), but with the restriction that t may start only in position i in a sentence (or, in fact, in the tag image of a sentence). We denote the Right Positional

[3] Note that the two sub-goals of the tagging ambiguity problem - reducing the number of paths and reducing word-level possibilities - are not identical. One can easily construct sample sentences where each word is two-way ambiguous, hence the sentence has 2^N potential paths, of which only two are valid, while still keeping all word-level ambiguity.

Short Context of length n of a tag t in position i by PSCr (t,n,i); similarly, PSCl (t,n,i) denotes the Left Positional Short Context of length n of a tag t in position i.

The examples in this paper will refer to the Right Short Context (positional and non-positional) of length 1. This is done mainly for the sake of clarity, but empirically it seems that even the limited set of constraints which can be expressed in these terms is powerful enough to invalidate many arcs in the Sentence Graph, thus resolving a great deal of the tagging ambiguity. See also a comment to that effect in [Marshall 83].

In the ideal case, by removing arcs from the graph on the basis of local constraints, the reduced sentence graph will contain the one and only globally valid path from Start ("$ <") to End ("> $"). In such cases, all tag assignments are also uniquely determined. But there may be cases where several paths survive the short context tests, not only because there exist more than one legal syntactic analysis, but due to the fact that even illegal analyses at the sentence level may conform to local constraints. This means that some of the words may still have ambiguous tag assignments, and, if followed by parsing, the parser will have to rule out the (hopefully few) impossible combinations. There is another interesting case, where no path at all exists after reduction. This signifies an illegal input sentence; hence a quick and effective means to invalidate (at least part of the) illegal inputs.

The probabilistic model, which will be discussed in chapter 4, suggests a different scheme for reducing the sentence graph. Here arcs are not necessarily removed, but rather evaluated for relative plausibility. Only high probable overall path(s) through the graph will be selected.

2. Extracting Local Constraints from a Grammar

If a formal grammar G exists for the language L, then, by definition, it contains all the syntactic knowledge about L. As such, it also contains the knowledge about Short Contexts. However, most of this knowledge is not explicit; for example, boundary conditions (the adjacency of a final tag in a constituent phrase with the initial tag of the following phrase) are not explicitly stated in a phrase structure grammar; they have to be extracted to be used for preliminary screening of lexical and morphological ambiguities as described above.

In the following we will assume that an unrestricted context-free phrase structure grammar (CFG), G, exists for the given language L. Later we will discuss other grammars too. We will use the following notations:

T = The set of Terminal symbols (the tag-set)
$< = The sentence start terminal
> $ = The sentence end terminal
V = The set of Variables (non-terminals)
S = The root variable for derivations
P = Production rules of the form A --> α ,
where A is in V, α is in (V U T)*

For technical purposes, we will substitute every grammar rule of the form S --> α with an equivalent rule S --> $< α > $, thus adding the two special terminals mentioned above to T.

We will now revise the definitions of Short Context from chapter 1, relative to the given grammar G. The rules in G are the only source for determining the validity of tag sequences.

The Right Short Context of length n of a terminal t (tag) relative to the grammar G is defined by:

$$SC^{r}_{G}(t,n) \quad \text{for } t \text{ in } T \text{ and for } n=0,1,2,3\ldots$$

$$= \left\{ \begin{array}{l} tz \mid z \text{ is in } T^{*}, \\ \quad |z| = n \text{ or} \\ \quad |z| < n \text{ if ">\$" is the last tag in } z, \\ \quad \text{and there exists a derivation of the} \\ \quad \text{form: } S \Longrightarrow \alpha\ t\ z\ \beta \\ \quad \text{where } \alpha \text{ and } \beta \text{ are in } (V \cup T)^{*} \end{array} \right\}$$

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

$$SC^{l}_{G}(t,n) \quad \text{for } t \text{ in } T \text{ and for } n=0,1,2,3\ldots$$

For short context with n = 1, it is useful (and natural) to define:

$$\text{next}(t) = \{ z \mid tz \text{ belongs to } SC^{r}_{G}(t,1) \}$$

The Right Positional Short Context of length n of a tag t in position i, relative to the grammar G, is defined by:

$$PSC^{r}_{G}(t,n,i) \quad \text{for } t \text{ in } T,\ n=0,1,2,3,\ldots,\ i>0$$

$$= \left\{ \begin{array}{l} tz \mid z \text{ is in } T^{*}, \\ \quad |z| = n \text{ or} \\ \quad |z| < n \text{ if ">\$" is the last tag in } z, \\ \quad \text{and there exists a derivation of the} \\ \quad \text{form: } S \Longrightarrow \alpha\ t\ z\ \beta \\ \quad \text{where } \alpha \text{ and } \beta \text{ are in } (V \cup T)^{*} \\ \quad \text{and } t \text{ is in the } i\text{-th position in a} \\ \quad \text{tag-image of a sentential form of } S \end{array} \right\}$$

The Left Positional Short Context is defined in a similar way and denoted by:

$$PSC^{l}_{G}(t,n,i)$$

The following is a procedure to compute the function next(t), from a CFG. Without loss of generality, one may assume that this CFG has no inaccessible symbols, has no useless symbols and is e-free, i.e. has no rules of the form V --> the empty string. [Aho 72] describes efficient algorithms to achieve this normal form.

We find the next(t) set by examining P, the rules of G:[4]

1. If there is a rule in P of the form:
   A --> α t x β and x is in T,
   then x is in next(t).

2. If there is a rule in P of the form:
   A --> α t B β and B is in V,
   then the set first(B) is
   a subset of next(t).

3. If there is a rule in P of the form:
   A --> α t
   then the set follow(A) is
   a subset of next(t).

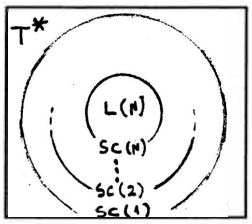
The computational complexity of the construction of the set next(t) depends on the complexity of computing the first and follow set. There are well known algorithms to find these sets from a given CFG. The complexity of follow(t) is exponential in the size of the look ahead window, which is the length of the context. This is another reason to limit the contexts to really short ones (although note that the extraction of constraints from the grammar is a one-time preprocessing phase, hence the performance issue is not critical).

To conclude this chapter, we borrow the concept of event dependency from probability theory, just to offer the following view on short context constraints. The events being concatenation of tags, the short context basically defines *independent* constraints, while in the full grammar the *dependent* constraints are expressed. This distinction is particularly apparent in SCr(t,1) or SCl(t,1), where "events" only apply to a pair of neighbors; as the context gets longer, the constraints become more dependent and closer to the full grammar. The metaphorical description above gets especially interesting when a statistical dimension is added to the model (see chapter 4). There, indeed, SC(1) considers independent probabilities of possible neighbors, where a full probabilistic grammar is supposed to look at the dependent events of tag concatenation along the full sentence.

It is therefore clear that the Short Context technique will license more sentences than a grammar would; or, from a dual point of view, it will invalidate only part of the impossible combinations of tag assignment. SCr(t,2) will have a closer fit coverage than SCr(t,1), and only in SCr(t,N) (where N is the finite length of a given sentence) the licensing power will be identical to the weak generative capacity of the full grammar

[4] The functions "first" and "follow" are used here much like in standard parsing techniques for both programming languages and natural languages; see [Aho 72] as a general reference.

(see illustration). However, SCr(t,N) has only the time complexity of a finite automaton (beware space complexity, though). The (theoretical and empirical) rate of convergence of the finite approximation is an interesting and important research topic. If indeed for a rather small number M, SCr(t, M) provides most of the licensing power of a given full grammar, then the performance promise of short context methods is consequential for a variety of applications (cf. [Church 80]). As mentioned before, it appears that even SCr(t,1) can drastically reduce the a-priori polynomial number of tag sequences, typically to a number linearly proportional to the length of the sentence.



3. An Example

Consider the following "toy grammar" for a small fragment of English (a variant of the basic sample grammar in [Tomita 86]).

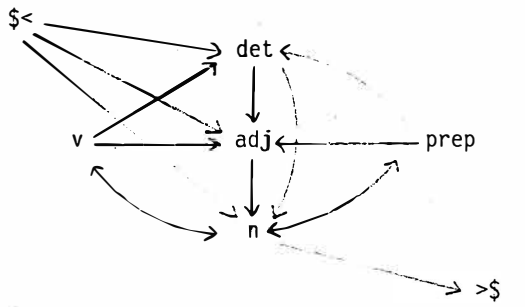
The tag set includes only: n (noun), v (verb), det (determiner), adj (adjective) and prep (preposition). The context free grammar G is:

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

G is a slightly modified version of a standard grammar, where the special symbols "$<" (start) and ">$" (end) are added.

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

```
      r
The SC (t,1) Graph
      G
```



The table of valid neighbors is derived directly from the graph:

```
   r
SC (t,1) Table
  G
```

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

This table describes the closure of next(t) for all terminals in G.

Of special interest is the complement of the SC(t,1) table, relative to T^2 . Here, information about terminal pairs which can never appear in a legal sentence is represented. Such a table may be used by grammar developers to test a grammar, presenting small "checklist tests" which are easy to make.

From the SC(t,1) graph above we can now extract information about the Positional PSC(t,1,i) possibilities. This is done by tracing the way from "$<" forward. The Positional Short Context tables are the following:

$$PSC_G^r(t,1,i)$$

Position:									
0 ---> 1		1 ---> 2		2 ---> 3		3 ---> 4			
$<	det	det	n	n	v	v	...		
$<	n	det	adj	n	prep	n	...		
$<	adj	n	prep	n	>$	prep	...		
		n	v	prep	det	det	...		
		n	>$	prep	n	adj	...		
		adj	n	prep	adj				
				v	det				
				v	n				
				v	adj				
				adj	n				

Note that from positions 3- > 4 on, the table gets identical to the general SC(t,l) table (the closure).

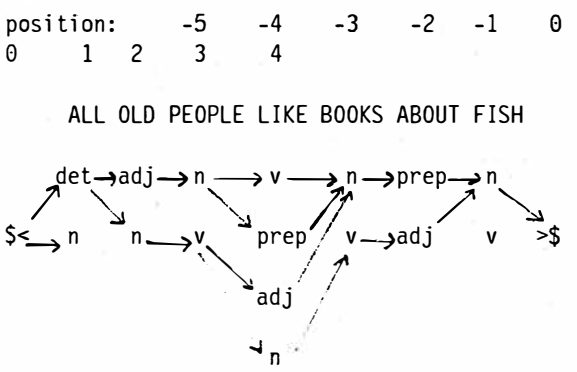
Another useful information one can obtain from the SC(t,l) graph is the inverse of the tables above - the Positional SC that may be allowed when going from the end of a sentence backwards. This is, in fact, the Positional Left Short Context. What has to be done to create the tables is to invert every arc in the SC(t,l) graph. Other than that, the procedure is the same. It is interesting to note that in our example the closure appears later when scanning the sentence backwards - from right to left.

A final technical comment before showing the operation on a sample sentence: When the short context of distinct occurrences of the same terminal is different, it is useful to distinguish between them using an index. This will add more information about the PSC when tracing the Sentence Graph.

Let us now consider the following sentence:

"All old people like books about fish."

The chart below shows the Reduced Sentence Graph - the original Sentence Graph from which invalid arcs (relative to the PSC tables) were removed.[5]



We are left with four valid paths through the sentence, out of 256 a-priori possible paths (256 = 2*2*2*4*2*2*2). Two paths represent legal syntactic interpretations (of which one is "the intended" meaning). The other two are locally valid but globally invalid (having either two verbs or no verb at all, in contrast to the grammar). SCr(t,2) would have invalidated one of the wrong two.

Note that in this particular example the method was quite effective in reducing sentence-wide interpretations (for applications like parsing), but it was not very good in individual word tagging disambiguation.

Finally, let us emphasize that, while it is not trivial to construct an interesting example in English to demonstrate all the above, in Hebrew, even relative to a grammar similar to the above, it is hard to find a written sentence without considerable ambiguity. Moreover, as mentioned earlier, Hebrew poses tagging ambiguity of a second order, where different-length tag sequences may be assigned to a single given word. But in graph terms, it only means that a certain sequence of tags can be represented as a sequence of linked vertices in SG, the sentence graph. Hence "second order ambiguity" does not present a problem to our method.

[5] The sentence is analyzed here relative to the limited tag set of the sample grammar. Depending on the tag set, the lexicon and the grammar, the level of ambiguity (and the results in this particular case) may be different.

4. Extensions

The method described above can be extended to be useful in a variety of situations other than those presented. In this chapter we briefly discuss several such extensions.

We already demonstrated how effective and efficient word tagging and path reduction can be used in a pre-parsing filter. We also mentioned applications (e.g. some types of proof-reading aids) which do not call for full parsing, but require "stand alone" tagging disambiguation and can benefit from fast recognition of many illegal inputs. On the other hand, for other applications, one may think of incorporation of short-context techniques directly into a parser. In such an environment, when the parser is about to test a hypothesis concerning the existence of a constituent, it will first check if local constraints do not rule out that hypothesis. The motivation is the same as that beyond different techniques combining top-down and bottom-up considerations. To render the method more effective, distinctions should be made between identical tags (terminals, categories) appearing in different constituents (phrase types). The process of extracting local constraints from the grammar can be changed to account for the required distinction (e.g. by indexing).

Another direction for extensions is to go beyond the model of straightforward context free grammars. The same process will hold as long as the short context can be easily computed from the grammar. The following are two such examples.

1. [Black 89] describes a process of transforming certain feature grammars into a finite state machine. The transition arcs in such a machine provide the full information required to construct our PSC tables.
2. Even when no efficient parser exists, L(SC) may still be easy to recognize. [Shamir 74] proved that testing membership in the family of the so-called context-free programmed languages is NP-complete; nevertheless, extracting local constraints from such grammars is easy. In fact, the recognition of L(SC) only depends on the existence of a formal grammar, not a parser.

We now turn to discuss a probabilistic language model, and see how short context considerations can be extended to account for probabilistic constraints.

In the probabilistic environment, adjacent tags are not only valid (1) or invalid (0), but are allowed in any given probability between 0 and 1. This model may be more realistic for NLP systems which process real-life texts, where some phenomena happen more frequently than others. The Short Context tables will therefore have to include weights.

We will first assume that a probabilistic context free grammar, such as described by [Fujisaki 89], [Wright 89] and others, exists for the given language. In a probabilistic CFG, rules are labeled by probability estimators. Typically, the sum of probabilities is 1 for all production rules sharing the same left hand side. The probability of a sentential form is computed from all estimators of the rules used in the process of derivation.

The probabilistic tables of the (Positional) Short Context can be extracted from such a grammar in various ways. The most natural (but not trivial!) method requires attachment and carrying over of probabilities through the procedure for calculating next(t), described in chapter 2. Another method to assign a probability to a tag pair [t1,t2], in a sentence image of n tags, could be based on evaluation of "dummy sentences" having t1 and t2 in positions i and i+1 respectively, and "wild card" entries elsewhere. But, since the probabilities attached to rules in the probabilistic CFG were most likely drawn from a corpus, it may make sense to calculate the short context information directly from the corpus, in parallel to the calculation of rule probabilities for the grammar. This is done by a simple counting of tag pairs appearing in successful analyses. To achieve a more natural normalization of statistical values, it may be better to define the weight of a tag pair in positions (i, i+1) in a sentence relative to all other possible tag pairs in the same positions. The method can be generalized for longer sequences of adjacent tags.

Similarly to the way a probabilistic CFG is constructed - by first defining the deterministic rules

and then attaching weights to rules - we can draw deterministic local constraints from a grammar and later assign relative frequency values to entries in the short context tables. Given a new sentence, one can first filter out all deterministically invalid arcs and only then evaluate paths in the reduced graph (where arcs are labeled with frequency estimators) for relative plausibility.

The resulting graph is similar to the notion of "span" in [Marshall 83] and [DeRose 88]. [DeRose 88] describes an efficient algorithm to find a plausible path in such graphs. The only difference is that our approach does not require unambiguous words to bound the scope of disambiguation - in our case the "$ <" and "> $" markers will define a scope of the full sentence.

Note that if no probabilistic grammar exists for the language, and even if there is no formal context free grammar available at all, but some operational parser is available, probabilistic constraints can still be drawn from a corpus. The process will involve analysis of sentences by the given parser, and counting of tag pairs (or longer tag sequences) present in successful analyses.[6] At the end of the corpus analysis, there will be a group of arcs for which the counter is still 0 (or below a given threshold). This may happen either because the arcs are indeed invalid - such arcs can be now removed completely from the tables (thus embedding, in fact, the deterministic method within the framework of the probabilistic one); or they may represent a marginal syntactic phenomenon in the text domain of the given corpus (here practical considerations will determine the decision whether to keep or to delete such arcs from the Short Context tables).

In this model it may be more convenient not to use probabilities, but rather to assign to each arc a rank, representing the complement of the counter relative to the largest one found. The larger the rank is, the less frequent (hence less plausible) is the corresponding arc.

A labeled sentence graph SG will now be created for input sentences, using these ranks. From this labeled graph, only the most probable path from start ($ <) to end (> $) is selected. For that, we suggest the algorithm by [Dijkstra 59], which efficiently finds the shortest weighted path between two vertices in a directed graph. In principle, one may want to identify more than the one most probable path, e.g. if the second best is also highly ranked. For that different (and more complex) algorithms are needed.

Note that the acquisition from a corpus described above brings the model very close to the corpus-based M-gram model, applied at the level of parts of speech; see [Katz 85], [Atwell 88], [Marshall 83], for accounts of related methods.

To conclude this chapter, we note that one may consider construction of deterministic grammars from corpora. Here the rules themselves will be defined based on data found in the text. Such grammars tend to be very large (cf. [Atwell 88]). Part of the reason is the grain of the tag set: such grammars might be inflated by the creation of "families" of very similar rules, not being able to recognize a generalization over similar tags. Another reason is in the distribution of rules (phrase structure) - only a small number of rules apply in a significant number of sample sentences, while most of the rules were derived from single examples. The performance efficiency of parsers (deterministic or probabilistic) based on such methods will greatly suffer from the large size of the grammar. But for the processing of local constraints, the size of the grammar is not terribly important. Once the preprocessing phase has been completed, the actual testing of constraints is not badly affected by the size of the constraints tables, thus making the local constraints approach effective in such an environment as well.

---

[6] It may not be absolutely required that only cases appearing in correct analyses are counted. Data resulting from wrong analyses may turn to be statistically insignificant, relative to real and frequent phenomena. cf. [Dagan 90].

5. Final Remarks

We have not attempted a rigorous discussion of the performance gains expected when applying tagging disambiguation in a pre-parsing filter and/or in the parsing process itself. The question is not easy. It strongly depends on the parsing technique, on one hand, and on the degree of ambiguity at the given language (as reflected in a given grammar), on the other hand. Naive bottom-up parsers, which assume a single combination of tags in each analysis pass against the grammar, can certainly benefit, by drastically reducing the exponential number of passes needed a-priori in cases of heavy ambiguity. Other more sophisticated parsing techniques (cf. [Kay 80], for example), can also save in computational complexity, by taking earlier decisions on inconsistent tag assignments and/or by requiring a smaller grammar. The detailed analysis here is not simple. But it seems that, although the constraints are drawn only from the grammar, and as such they are somehow expressed (explicitly or implicitly) and will take effect during parsing, the different order of computation and the restriction to finite-length considerations are sources for considerable time saving.

Another important question concerns properties of the grammar that help build an effective filter of tentative paths. The grain of the tag set is such a significant factor. A better refined tag set helps express more refined syntactic claims, but it also gives rise to a greater level of tagging ambiguity. It also requires a larger grammar (or longer lists of conditions on features, attached to phrase structure rules, which we here assume to be already reflected in the rules themselves), hence a larger set of local constraints. But these constraints will be much more specific and therefore more effective in resolving ambiguities. A rigorous analysis of this issue will help understand better what makes an effective disambiguator. An important point to make is that our method guarantees uniformity of the tag set used for the filter and for any parser acting upon the given grammar, thus making it useful in a variety of environments.

Acknowledgements

M. Bahr and E. Lozinskii gave us helpful comments and suggestions on earlier drafts of this paper. We gratefully acknowledge their contribution.

References

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

[**Atwell 88**] Eric S. Atwell and Clive Souter. Experiments with a Very Large Corpus-based Grammar. Proc. of *the 15th int'l conference of the ALLC*, Jerusalem, June 1988.

[**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.

[**Black 89**] Alan. W. Black. Finite State Machines from Feature Grammars. Proc. of *the 1st International Parsing Workshop*, Pittsburgh, June 1989.

[**Choueka 85**] Yaacov Choueka and Serge Lusignan. Disambiguation by Short Contexts. *Computers in the Humanities*, no. 3, Vol. 19, July 1985.

[**Church 80**] Kenneth W. Church. On Memory Limitations in Natural Language Processing. *MSc thesis, MIT*. 1980.

[**DeRose 88**] Steven J. DeRose. Grammatical Category Disambiguation by Statistical Optimization. *Computational Linguistics*, 14/1, Winter 1988.

[**Dagan 90**] Ido Dagan and Alon Itai. Processing Large Corpora for Reference Resolution. Proc. of *the 13th COLING conference*, Helsinki, 1990.

[**Dijkstra 59**] Edsgar W. Dijkstra. A Note on Two Problems in Connection With Graphs. *Numerische Mathematik*, 1, pp. 269-271.

[**Francis 82**] W. Nelson Francis and Henry Kucera. *Frequency Analysis of English Usage: Lexicon and Grammar*, Houghton-Mifflin, 1982.

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

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

1971.

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

[**Kay 80**] Martin Kay. Algorithm Schemata and Data Structures in Syntactic Processing. Report CSL-80-12, 1980. Reprinted in *Readings in Natural Language Processing*, Grosz, Sparck-Jones and Webber (eds.), Morgan Kaufman, 1986.

[**Lozinskii 86**] Eliezer L. Lozinskii and Sergei Nirenburg. Parsing in Parallel. *Comp. Languages*, UK, Vol 11, No. 1, pp 39-51, 1986.

[**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.

[**Shamir 74**] Eli Shamir and Catriel Beeri. Checking Stacks and Context Free Programmed Grammars Accept P-complete Languages. Proc. of *the 2nd Colloq. on Automata languages and programming*, Lecture Notes in Computer Science, Vol. 14, pp 27-33, 1974.

[**Tomita 86**] Masaru Tomita. *Efficient Parsing for Natural Language*, Kluwer Academic Pub., 1986.

[**Wright 89**] J. H. Wright and E. N. Wrigley. Probabilistic LR Parsing for Speech Recognition. Proc. of *the 1st International Parsing Workshop*, Pittsburgh, June 1989.