Complexity of Description of Primitives: Relevance to Local Statistical Computations

Aravind K. Joshi and **B. Srinivas**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA

{joshi, srini}@linc.cis.upenn.edu

Introduction

In this paper we pursue the idea that by making the descriptions of primitive items (lexical items in the linguistic context) more complex, we can make the computation of linguistic structure more local[1]. The idea is that by making the descriptions of primitives more complex, we can not only make more complex constraints operate more locally but also verify these constraints more locally. Statistical techniques work better when such localities are taken into account[2] Of course, there is a price for making the descriptions of primitives more complex. The number of different descriptions for each primitive item is now much larger than when the descriptions are less complex. For example, in a lexicalized tree-adjoining grammar (LTAG), the number of trees associated with each lexical item is much larger than the number of standard parts-of-speech (POS) associated with that item. Even when the POS ambiguity is removed the number of LTAG trees associated with each item can be large, on the order of 10 trees in the current English grammar in the XTAG system[3]. This is because in LTAG, roughly speaking, each lexical item is associated with as many trees as the number of different syntactic contexts in which the lexical item can appear. This, of course, increases the local ambiguity for the parser. The parser has to decide which complex description (LTAG tree) out of the set of descriptions associated with each lexical item is to be used for a given reading of a sentence, even before combining the descriptions together. The obvious solution is to put the burden of this job entirely on the parser. The parser will eventually disambiguate all the descriptions and pick one per object, for a given reading of the sentence. This is what the parser is expected to do for disambiguating the standard POS, unless a separate POS disambiguation module is used (Church, 1988). Many parsers, including XTAG, use such a module called a POS tagger.

LTAGs present a novel opportunity to reduce the amount of disambiguation done by the parser. We can treat the LTAG trees associated with each lexical item as more complex parts-of-speech which we call **supertags**. In this paper, we report on some experiments on direct supertag disambiguation, without parsing in the strict sense, using lexical preference and local lexical dependencies (acquired from a corpus parsed by the XTAG system). The information extracted from the XTAG-parsed corpus contains, for each item and its supertag, a probability distribution of the distances of other items and their supertags that are expected by it. We have devised a method somewhat akin to the standard POS tagger that disambiguates supertags without

[1] Let $\sum$ be the alphabet consisting of the names of elmentary trees in an LTAG. Then $\sum^*$ is the set of all strings over this alphabet including the null string. The tree γ_1 and γ_2 in a string of tree names are said to be $\sum^*$-local if they are separated by any string in $\sum^*$. For brevity, we will continue to use the term local instead of the term $\sum^*$-local.

[2] The work described here is completely different from the work reported in (Resnik, 1992) and (Schabes, 1992) concerning stochastic TAGs.

[3] See Section on Data Collection

doing any parsing.

The idea of using complex descriptions for primitives to capture constraints locally has some precursors in AI. For example, the Waltz algorithm (Waltz, 1975) for labeling vertices of polygonal solid objects can be thought of in these terms, although it is not usually described in this way. There is no statistical computations in the Waltz algorithm, however. The supertag disambiguation experiments, as far as we know, are the first to use these ideas in the linguistic context. Of course, we also show how the supertag disambiguation naturally lends itself to the application of statistical techniques. In the following sections we will briefly describe our approach and some preliminary results of supertag disambiguation as an illustration of our main theme: the relationship of the complexity of descriptions of primitives to local statistical computations. A more complete analysis of this technique and experimental results will eventually be reported elsewhere.

Lexicalized Tree Adjoining Grammars

Lexicalized Tree Adjoining Grammar (LTAG) is a lexicalized tree rewriting grammar formalism (Schabes, 1990). The primary structures of LTAG are called ELEMENTARY TREES. Each elementary tree has a lexical item (anchor) on its frontier and serves as a complex description of the anchor. An elementary tree provides a domain of locality larger than that provided by CFG rules over which syntactic and semantic (predicate-argument) constraints can be specified. Elementary trees are of two kinds: INITIAL TREES and AUXILIARY TREES. Examples of initial trees (αs) and auxiliary trees (βs) are shown in Figure 1. Nodes on the frontier of initial trees are marked as substitution sites by a '$\downarrow$', while exactly one node on the frontier of an auxiliary tree, whose label matches the label of the root of the tree, is marked as a foot node by a '$*$'. The other nodes on the frontier of an auxiliary tree are marked as substitution sites. LTAG factors out recursion from the statement of the syntactic dependencies. Elementary trees (initial and auxiliary) are the domain for specifying dependencies. Recursion is specified via the auxiliary trees.

Elementary trees are combined by **Substitution** and **Adjunction** operations. Substitution inserts elementary trees at the substitution nodes of other elementary trees. Adjunction inserts auxiliary trees into elementary trees at the node whose label is the same as the root label of the auxiliary tree. As an example, the component trees (α_8, α_2, α_3, α_4, β_8, α_5, α_6), shown in Figure 1 can be combined to form the parse tree for the sentence *John saw a man with the telescope*[4] as follows:

1. α_8 substitutes at the NP_0 node in α_2.
2. α_3 substitutes at the DetP node in α_4, the result of which is substituted at the NP_1 node in α_2.
3. α_5 substitutes at the DetP node in α_6, the result of which is substituted at the NP node in β_8.
4. The result of step (3) above adjoins to the VP node of the result of step (2). The resulting parse tree is shown in Figure 2.

The process of combining the elementary trees that yield a parse of the sentence is represented by the **derivation tree**, shown in Figure 2. The nodes of the derivation tree are the tree names that are anchored by the appropriate lexical item. The composition operation is indicated by the nature of the arcs–broken line for substitution and bold line for adjunction–while the address of the operation is indicated as part of the node label. The derivation tree can also be interpreted as a dependency graph with unlabeled arcs between words of the sentence as shown in Figure 2.

We will call the elementary trees associated with each lexical item super part-of-speech tags or **supertags**.

[4]The parse with the PP attached to the NP has not been shown.

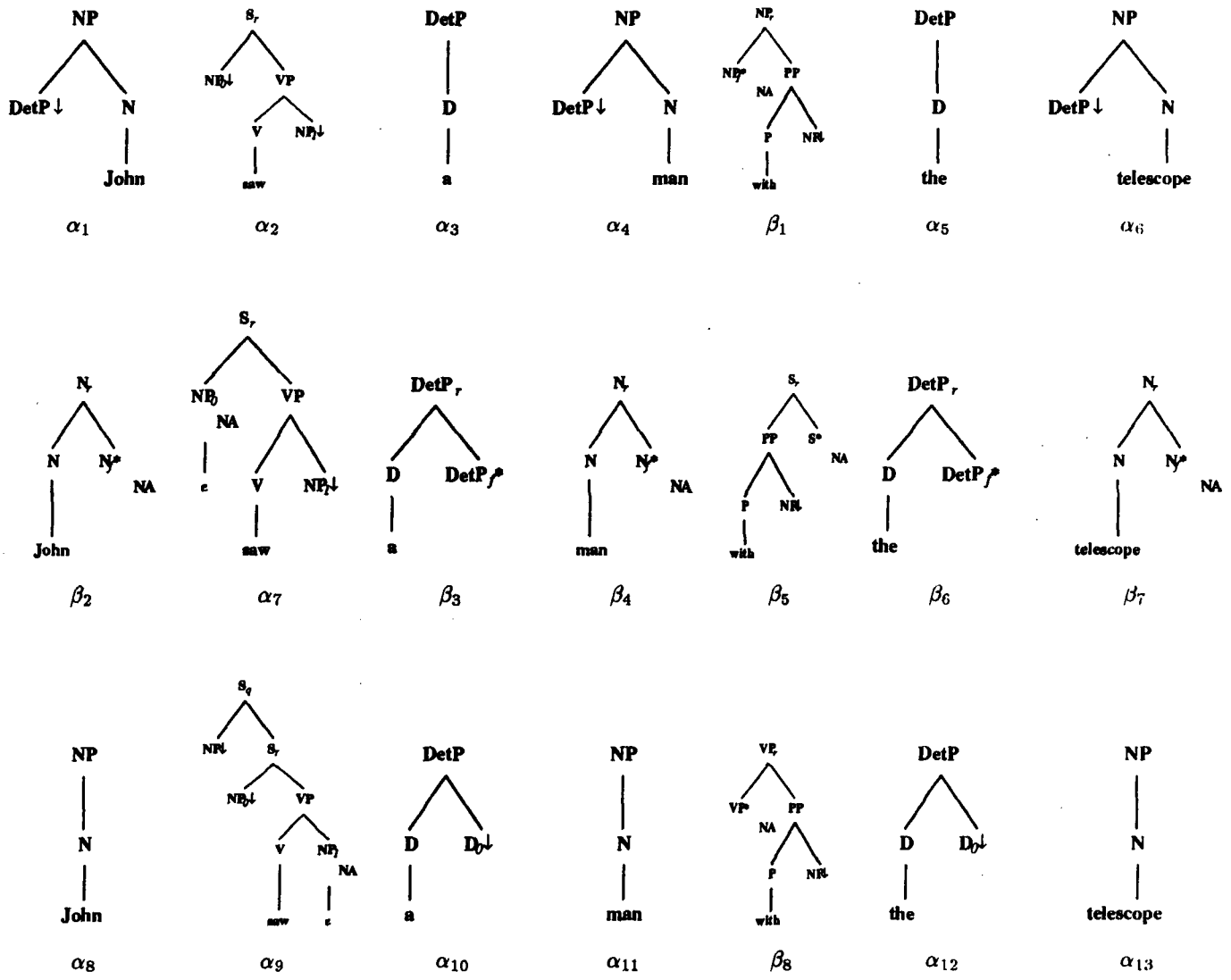

Figure 1: **Elementary trees of LTAG**

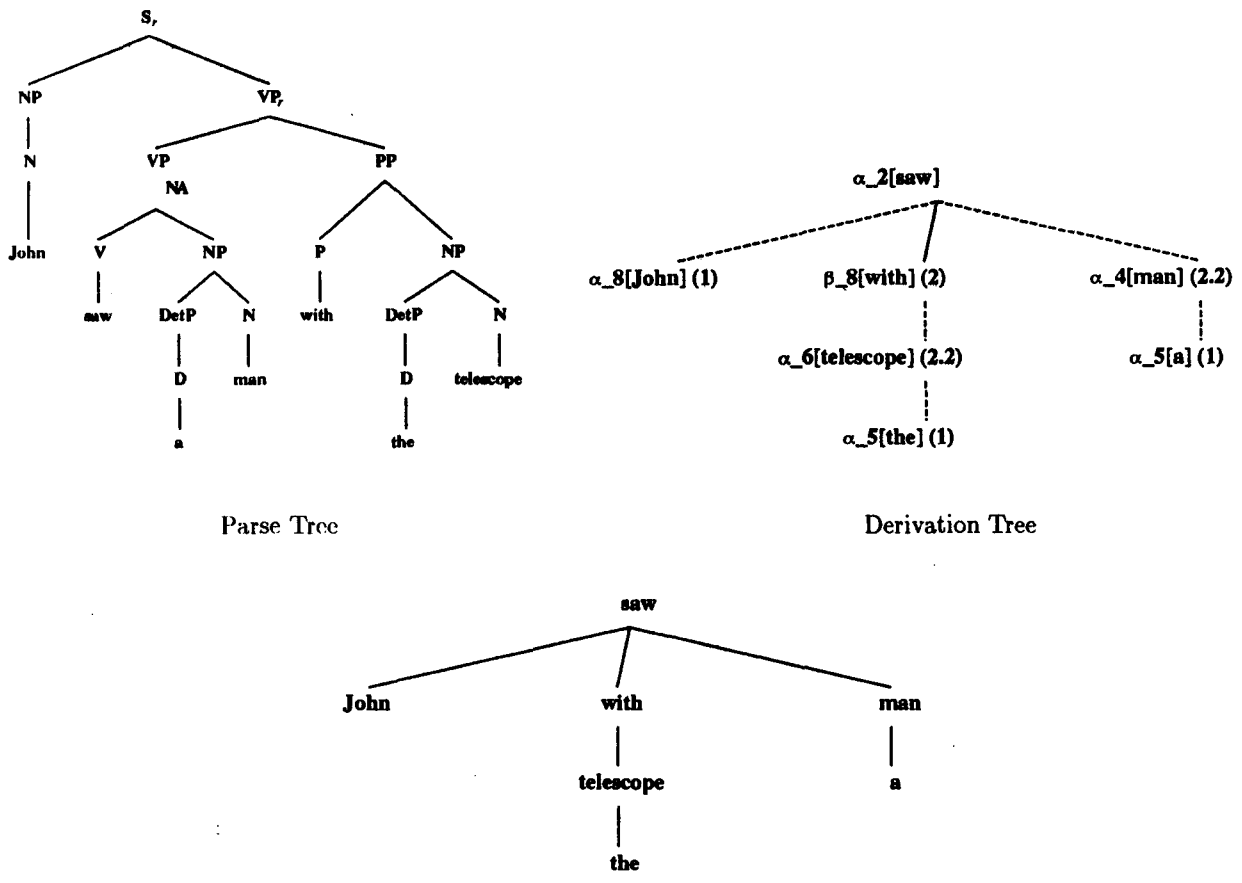

Figure 2: **Structures of LTAG**

Example of Supertagging

As a result of localization in LTAG, a lexical item may be associated with more than one supertag. The example in Figure 3 illustrates the initial set of supertags assigned to each word of the sentence *John saw a man with the telescope*. The order of the supertags for each lexical item in the example is completely irrelevant. Figure 3 also shows the final supertag sequence assigned by the supertagger, which picks the best supertag sequence using statistical information (described in the next section) about individual supertags and their dependencies on other supertags. The chosen supertags are combined to derive a parse, as explained in the previous section.

The parser without the supertagger would have to process combinations of the entire set of 28 trees; the parser with it need only process combinations of 7 trees.

Dependency model of Supertagging

One might think that a n-gram model of standard POS tagging would be applicable to supertagging as well. However, in the n-gram model for standard POS tagging, dependencies between parts-of-speech of words that appear beyond the n-word window cannot be incorporated into the model. This limitation does not have

Sentence:	John	saw	a	man	with	the	telescope.
Initial Supertag set:	α_1	α_2	α_3	α_4	β_1	α_5	α_6
	β_2	α_7	β_3	β_4	β_5	β_6	β_7
	α_8	α_9	α_{10}	α_{11}	β_8	α_{12}	α_{13}
Final Assignment:	α_8	α_2	α_3	α_4	β_8	α_5	α_6

Figure 3: **Supertag Assignment for** *John saw a man with the telescope*

a significant effect on the performance of a standard trigram POS tagger, since it is rare for dependencies to occur between POS tags beyond a three-word window. However, since dependencies between supertags do not occur in a fixed sized window, the n-gram model is unsuitable for supertagging. This limitation can be overcome if no a priori bound is set on the size of the window, but instead a probability distribution of the distances of the dependent supertags for each supertag is maintained. A supertag is **dependent** on another supertag if the former substitutes or adjoins into the later.

Experiments and Results

Table (1) shows the data required for the dependency model of supertag disambiguation. Ideally each entry would be indexed by a (word, supertag) pair but, due to sparseness of data, we have backed-off to a (POS, supertag) pair. Each entry contains the following information.

- POS and Supertag pair.
- List of + and −, representing the direction of the dependent supertags with respect to the indexed supertag. (Size of this list indicates the total number of dependent supertags required.)
- Dependent supertag.
- Signed number representing the direction and the ordinal position of the particular dependent supertag mentioned in the entry from the position of the indexed supertag.
- A probability of occurrence of such a dependency. The sum probability over all the dependent supertags at all ordinal positions in the same direction is one.

For example, the fourth entry in the Table 1 reads that the tree α_2, anchored by a verb (V), has a left and a right dependent (−, +) and the first word to the left (−1) with the tree α_8 serves as a dependent of the current word. The strength of this association is represented by the probability 0.300.

The dependency model of disambiguation works as follows. Suppose α_2 is a member of the set of supertags associated with a word at position n in the sentence. The algorithm proceeds to satisfy the dependency requirement of α_2 by picking up the dependency entries for each of the directions. It picks a dependency data entry (fourth entry, say) from the database that is indexed by α_2 and proceeds to set up a path with the first word to the left that has the dependent supertag (α_8) as a member of its set of supertags. If the first word that has α_8 as a member of its set of supertags is at position m, then an arc is set up between α_2 and α_8. Also, the arc is verified so that it does not kite-string-tangle[5] with any other arcs in the path up to α_2. The path probability up to α_2 is incremented by $\log 0.300$ to reflect the success of the match. The path probability up to α_8 incorporates the unigram probability of α_8. On the other hand, if no word is found that has α_8 as a member of its set of supertags then the entry is ignored. A successful supertag sequence is one which assigns a supertag to each position such that

[5]Two arcs (a,c) and (b,d) kite-string-tangle if $a < b < c < d$ or $b < a < d < c$.

(P.O.S,Supertag)	Direction of Dependent Supertag	Dependent Supertag	Ordinal position	Prob
(D,α_5)	()	-	-	-
(N,α_8)	()	-	-	-
(N,α_1)	(−)	α_3	−1	0.999
(V,α_2)	(−, +)	α_8	−1	0.300
(V,α_2)	(−, +)	α_8	1	0.374

Table 1: Dependency Data

each supertag has all of its dependents and maximizes the accumulated path probability. The direction of the dependent supertag and the probability information are used to prune the search. A more detailed and formal description of this algorithm will appear elsewhere.

The implementation and testing of this model of supertag disambiguation is underway. Preliminary experiments on short fragments show a success rate of 88% i.e. a sequence of correct supertags is assigned.

Data Collection

The data needed for disambiguating supertags (Section) have been collected by parsing the Wall Street Journal[6], IBM-manual and ATIS corpora using the wide-coverage English grammar being developed as part of the XTAG system (XTAG Tech. Report, 1994). The parses generated for these sentences are not subjected to any kind of filtering or selection. All the derivation structures are used in the collection of the statistics.

XTAG is a large ongoing project to develop a wide-coverage grammar for English, based on the LTAG formalism. It also serves as an LTAG grammar development system and includes a predictive left-to-right parser, a morphological analyzer and a POS tagger. The wide-coverage English grammar of the XTAG system contains 317,000 inflected items in the morphology (213,000 for nouns and 46,500 for verbs among others) and 37,000 entries in the syntactic lexicon. The syntactic lexicon associates words with the trees that they anchor. There are 385 trees in all, in the grammar which is composed of 40 different subcategorization frames. Each word in the syntactic lexicon, on the average, depending on the standard POS of the word, is an anchor for about 8 to 40 elementary trees.

[6]Sentences of length $\leq$ 15 words.

Conclusion

In this paper we have shown that increasing the complexity of descriptions of primitive objects, lexical items in the linguistic context, enables more complex constraints to be applied locally. However, increasing the complexity of descriptions greatly increases the number of such descriptions for the primitive object. In a lexicalized grammar such as LTAG each lexical item is associated with complex descriptions (supertags) on the average of 10 descriptions. A parser for LTAG, given a sentence, disambiguates a large set of supertags to select one supertag for each lexical item before combining them to derive a parse of the sentence. We have presented a new technique that performs the disambiguation of supertags using local information such as lexical preference and local lexical dependencies as an illustration of our main theme of the relationship of complexity of descriptions of primitives to local statistical computations. This technique, like POS disambiguation, reduces the disambiguation task that needs to be done by the parser. After the disambiguation, we have effectively completed the parse of the sentence and the parser needs 'only' to complete the adjunctions and substitutions.

References

Kenneth Ward Church. 1988. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *2nd Applied Natural Language Processing Conference 1988.*

Philip Resnik. 1992. Probabilistic Tree-Adjoining Grammar as a Framework for Statistical Natural Language Processing *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING '92)*, Nantes, France, July 1992

Yves Schabes, Anne Abeillé, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proceedings of the* 12^{th} *International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary, August.

Yves Schabes. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, August 1990. Available as technical report (MS-CIS-90-48, LINC LAB179) from the Department of Computer and Information Science.

Yves Schabes. 1992. Stochastic Lexicalized Tree-Adjoining Grammars *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING '92)*, Nantes, France, July 1992.

David Waltz. 1975. *Understanding Line Drawings of Scenes with Shadows* in *Psychology of Computer Vision* by Patrick Winston, 1975.

XTAG Technical Report. 1994. Department of Computer and Information Sciences, University of Pennsylvania, Philadelphia, PA. In progress.