Syntactic and Semantic Filtering in a Chart Parser

Sayan Bhattacharyya[1] and Steven L. Lytinen[2]

[1] University of Michigan, Dept. of Electrical Engineering & Computer Science, Ann Arbor, MI 48109, USA
[2] DePaul University, Dept. of Computer Science, 243 S. Wabash, Chicago, IL 60604, USA

Abstract. This paper describes a method to enhance the performance of a unification-style bottom-up chart parser by means of top-down filtering techniques. The filter developed consists of a syntactic module which prevents the construction of redundant edges in the chart by ensuring that a proposed edge in the chart can really be syntactically combined with neighboring edges later, and a semantic module which ensures that the semantic information in a proposed edge in the chart is compatible with semantic information in other edges.

1 Introduction

Chart parsers used in natural language processing parse an input sentence by building up a data structure called the chart. A chart is a network of vertices representing points in the sentence which are linked by edges which represent constituents of the sentence. In a unification-style chart parser [5] the chart is augmented step by step by adding pieces of description according to the grammar. Each partial description added remains in the chart and serves to constrain the possibilities for further augmentation. In LINK, a bottom-up unification-style chart parser described by Lytinen [3], a directed acyclic graph (DAG) is built to represent the analysis of a sentence. Edges built by LINK are labeled by DAGs incorporating syntactic as well as semantic information. New edges are added to the chart by applying unification rules. This guarantees that the new constituent added to the parse has the necessary syntactic and semantic features. We present a way of integrating bottom-up parsing with top-down parsing, both in terms of syntax and semantics. Our approach can be described as bottom-up parsing with top-down filtering. We modified LINK using this approach. Similar approaches have been variously called in the literature a "filter" [1] and an "oracle" [4].

2 Necessity For Top-down Filtering Methods

In a top-down parser, parsing is rule-driven while in a bottom-up parser, parsing is data-driven. Thus in a top-down chart parser an active edge in the chart is sought to be expanded with all rules in the grammar which have the current symbol as the left-hand side, causing the parser to be over-productive in edge

construction. In a bottom-up parser on the other hand only complete edges are combined, by the application of rules whose right hand sides correspond to the completed edges. A bottom-up parser is also over-productive because it will build some useless edges that cannot be combined with edges lying to the left or to the right. So top-down filtering is effective in a bottom-up parser by reducing the production of useless edges by checking whether the edge that is sought to be built has any chance of combining with an adjacent edge to form part of a larger edge. Making use of top-down information about the semantic context as well can help to reduce redundancy by anticipating and blocking out unlikely choices.

3 Implementing the Filter

Ordinarily, LINK traverses the input sentence from left to right, building up possible edges as it does so. At the outset, during the initialisation of the chart, link-building is a strictly left-to-right process. After chart initialisation (after all word-level, i.e. primitive, edges have been built), phrase-level edges will be attempted to be built, guided by the heuristic that edges will be first sought to be constructed out of previously unused edges. (Thus phrase-level edge construction may not always occur strictly from left to right.)

3.1 Syntactic Component of the Filter

Chart Initialisation. The objective of initialising the chart is to build edges around each word in the input sentence. For each word of the input sentence, LINK was made to look up its corresponding lexical entry in the LINK lexicon to check for possible syntactic ambiguity, ambiguity being defined here as the existence of multiple definitions belonging to different syntactic categories. If no syntactic ambiguity is detected for the candidate word, LINK is allowed to proceed to build edge(s) around the word in the ordinary way. On the other hand, if syntactic ambiguity is detected, then for each candidate syntactic category a check is made on the chart constructed so far to determine whether a valid left-adjacent predecessor exists. The edge is constructed only if such a left-predecessor exists.

Constructing the Complete Chart. The objective is to build progressively larger edges, combining words into phrases and phrases into increasingly larger phrases until an edge is ultimately constructed which spans across the entire sentence and corresponds to a complete parse. Link construction is an iterative process that continues until the parse is complete or until there are no further edges in the chart that can be constructed. At each step of the iterative process, the chart is examined from left to right to check if there is a constituent sequence of edges that unify with the right-hand-side of a grammar rule. For the first sequence so obtained, it is checked whether there exists any left-adjacent phrase contiguous to the candidate phrase for which the edge is being sought to be

built that corresponds to a syntactic category which is a valid predecessor of the syntactic category of the proposed candidate edge. The attempt to build the edge is abandoned if such a left-adjacent phrase does not exist. If such a left-adjacent phrase does exist, a similar check is carried out for right-adjacency. However, as in general it is not guaranteed that a complete non-primitive right-adjacent edge will exist (because of the left-to-right nature of the parsing process) the right-adjacency check is limited to checking the existence of only valid right-adjacent *words* (i.e. primitive edges) and not valid right-adjacent *phrases* (i.e. non-primitive edges). A edge will be finally built if and only if both the left-adjacency and right-adjacency checks are successful.

3.2 Semantic Component of the Filter

The task of the semantic filter is to perform word-sense disambiguation early on in the parse in order to prevent construction of edges in the chart which correspond to word-senses which are apparently irrelevant in the given context. This is expected to lead to considerable savings as word-sense ambiguity is a big source of parsing inefficiency in most large applications; for example, Waltz [6] estimates that in English, on the average, each word has as many as 3.7 different senses.

LINK maintains a semantic hierarchy consisting of `IS-A` relationships between semantic categories. In order for the fillers of the slot of a certain semantic category to find something to unify with, either an entity belonging to the same semantic category of the slot filler, or an entity belonging to a descendant semantic category of the slot filler in the semantic hierarchy (not necessarily immediate descendant) must be present. Our strategy is to generate a table listing the valid semantic associates of every semantic category. The semantic associates of a semantic category would include all the slot-filler semantic categories for that semantic category and all the descendants of each slot-filler semantic category in the semantic hierarchy. Then, while parsing, as soon as an edge is sought to be built which would assign to an edge a particular semantic category, the table of valid semantic associates is looked up to determine if any valid semantic associate for that semantic category exists within the input sentence. If yes, the proposed edge is passed for construction (provided it is also passed by the syntactic component of the filter) and if not, the proposed edge is rejected. Thus a proposed edge must pass through both the syntactic and semantic filters before it is cleared for construction.

For example, for the semantic hierarchy shown in Figure 1 and the following semantic category definition:

```
PTRANS:

    (actor) = ANIMATE
```

the table of semantic associates generated will be:

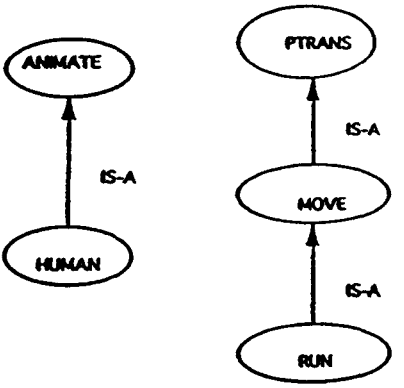

Fig. 1. An example semantic hierarchy

semantic category	semantic associates
PTRANS	ANIMATE MAN
MOVE	ANIMATE MAN
RUN	ANIMATE MAN
ANIMATE	PTRANS MOVE RUN
MAN	PTRANS MOVE RUN

As another example, consider the two sentences **John shot some bucks** and **John spent some bucks**. The word **bucks** is a source of ambiguity here. Syntactically, the two sentences are exactly alike, so that the syntactic component of the filter cannot help resolve the ambiguity. In the grammar that we used for this sentence, the word **shot** had the semantic category **PROPEL** and the word **spent** had the semantic category **ATRANS**. The word **bucks** had the choice of two semantic categories, **ANIMAL** and **MONEY**, among which it was necessary to disambiguate.

The semantic definitions of **ATRANS**, **PROPEL**, **ANIMAL** and **MONEY** were as follows:

ATRANS:

```
is-a (ACTION)
(actor)  = HUMAN
(object) = PHYS-OBJ
(from)   = HUMAN
```

PROPEL:

```
    is-a (ACTION)
    (object) = ANIMAL

MONEY:

    is-a (PHYS-OBJ)

ANIMAL:

    is-a (ANIMATE)
```

For our example grammar, the table of semantic associates had the following as entries for the semantic categories **ANIMAL** and **MONEY** after stepwise construction according to the aforementioned algorithm:

semantic category	semantic associates
ANIMAL	PROPEL PHYS-STATE
MONEY	POSSESSION ATRANS DIVIDE

Note that **ATRANS** was not a valid semantic associate of **ANIMAL** and **PROPEL** was not a valid semantic associate of **MONEY**. Thus, the edge corresponding to **MONEY** will not be built in the first sentence as no valid semantic associate of **MONEY** is present in the first sentence. Similarly the edge corresponding to **ANIMAL** will not be built in the second sentence.

Our method of semantic filtering is similar in some ways to some semantic disambiguation techniques described in the literature, such as the preference semantics used by Wilks [7] and the polaroid words technique described by Hirst [2].

4 Performance Issues

We measured the performance of our system (LINK with the filter described) *vis-a-vis* that of LINK without filter by parsing a set of forty-eight sentences, with a grammar that had been expressly written for the domain. Figure 2 shows the comparative performance improvement obtained by using the syntactic component of the filter. In this figure the ratio of rules applied during the parse for the two methods and the ratio of time needed to parse a sentence by the two methods have been plotted against the number of words per sentence. It can be seen that performance improvement appears to increase the longer is the input sentence. Figure 3 and Figure 4 show how the semantic component of the filter leads to a further improvement in performance for those sentences which can be parsed after the addition of the semantic filter. These two figures show the results of parsing a subset of 24 sentences from our domain.

Figure 3 compares the performance of the combined syntactic and semantic filter to that of the syntactic filter taken singly while Figure 4 separately compares the performance of the combined syntactic and semantic filter and the syntactic filter to the performance in the absence of any filter.

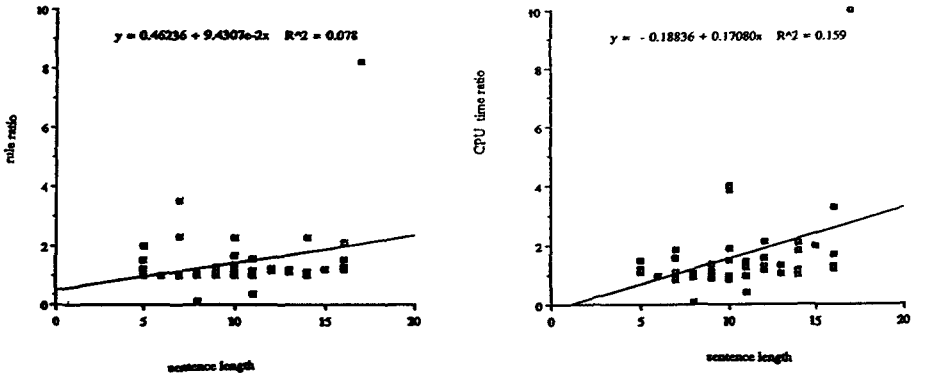

Fig. 2. (a) Plot of ratio of number of rules used by parser without filter to number of rules used with syntactic filter. (b) Plot of ratio of CPU time needed by parser without filter to CPU time needed with syntactic filter

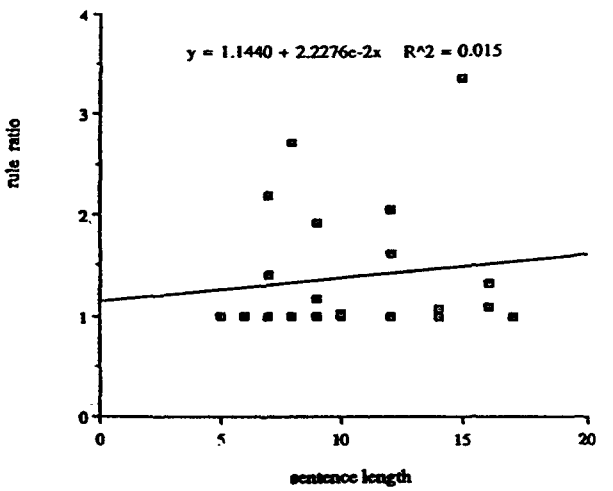

Fig. 3. Plot of ratio of number of rules used by parser with only syntactic filter to number of rules used with both syntactic and semantic filter.

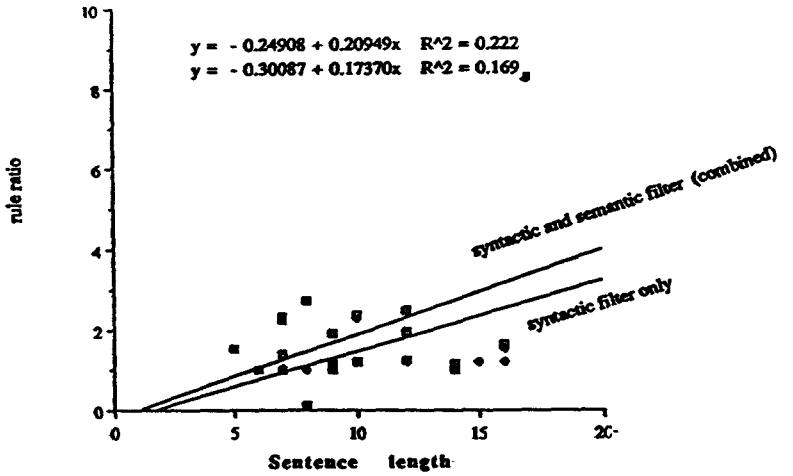

Fig. 4. Plot of ratio of number of rules used by parser with no filter to number of rules used by parser with both syntactic and semantic filters.(The plot of rule ratio improvement using syntactic filter only has also been shown for the sake of comparison)

References

1. Grishman, R.: Computational Linguistics. Cambridge University Press, Cambridge (1986)
2. Hirst, G.: Semantic Interpretation and the Resolution of Ambiguity. Cambridge University Press, Cambridge (1987)
3. Lytinen, S.L.: A unification-based integrated natural language processing system. Computers Math. Applic. Vol. **23**, no 6-9 (1992)
4. Pratt, V.: Lingol, a progress report. Advance Papers 4th Intl. Joint Conf. Artificial Intelligence 422-8 (1973).
5. Shieber, S.: An Introduction to Unification-Based Approaches to Grammar. Lawrence Erlbaum Associates, Hillsdale, NJ (1986).
6. Waltz, D.: Semantic Structures. Lawrence Erlbaum Associates, Hilldale, N.J. (1989)
7. Wilks, Y.: Preference Semantics. in E. Keenan, ed, Formal Semantics of Natural Language. Cambridge University Press, Cambridge (1975)