A Modular Natural Language Processing Architecture to Aid Novel Interpretation

Justin Peterson
College of Computing
justin@pravda.gatech.edu

Dorrit Billman
School of Psychology
billman@pravda.gatech.edu
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

Successful and robust natural language processing must efficiently integrate multiple types of information to produce an interpretation of input. Previous approaches often rely heavily on either syntax or semantics, verb-specific or highly general representations. A careful task analysis identifies principled subsets of information from across these spectra are needed. This presents challenges to efficient and accurate processing. We present a modular architecture whose components reflect the distinct types of information used in processing. Its control mechanism specifies the principled manner in which components share information. We believe this architecture provides benefits for processing sentences with novel verbs, ambiguous sentences, and sentences with constituents placed outside their canonical position.

Introduction

The task of natural language processing is to produce an interpretation of the linguistic input. This task is extremely difficult because the mapping from input to interpretation is complex: a single form can have multiple meanings and a single meaning can be expressed in several different ways. Further, language comprehension occurs over a very wide range of circumstances. We comprehend novel utterances, with novel verbs, *Fax me a better copy* as well as close variations on familiar phrases *Hope you're feeling better*. As a consequence, it is hard to identify the information and processes sufficient for the task across varied circumstances. Is the critical information syntactic or conceptual? Is it highly specific, such as a verb's meaning and argument structure, or quite general, such as specifying that direct objects must immediately follow the verb? Are sentence interpretations retrieved from memory or constructed anew?

Our research attempts systematically to identify what distinctions in the input and in memory provide the information needed. By taking a task-analysis approach we attempt to specify just what distinctions of each type are important. We propose that very selective yet heterogeneous clues are used in interpretation. A discussion of this proposal as well the assumptions we have adopted from the work of others can be found in (Peterson & Billman 1990). The focus of this paper, however, is to outline an architecture for sentence interpretation which efficiently operates on these varied sources of information to produce a sentence interpretation.

Prior Work

Natural language processing systems vary in their assumptions about the importance of syntactic versus conceptual structure. They differ in emphasizing general principles versus specific content knowledge. And they contrast in treating language processing as retrieval of an interpretation from memory versus treating it as generation of a novel structure from much smaller, known, components. Two quite opposite sets of choices have dominated prior work. Some natural language processing systems emphasize detailed, highly specific representations that contain primarily conceptual (or semantic) information and treat interpretation as retrieving the familiar events or situation being described (Riesbeck & Martin 1986). Semantic expectations are central to this approach. We shall call this extreme the conceptual recognition view. Other systems emphasize highly schematic principles of broad scope that are primarily syntactic and that treat sentence processing as generation of the novel structure specified in a unique utterance (Woods 1978). We shall call this the syntactic construction view.

The conceptual recognition view fares well when the sentence does indeed refer to a familiar event using a familiar form. For example, a familiar verb can be accessed from the lexicon, specify slots for its arguments, and thus allow arguments to be bound to a completely prespecified structure in a simple and stereotyped way (Birnbaum & Selfridge 1981). Where expectations and background knowledge are both complete and correct, this provides a fast and effective processing strategy. Difficulties arise if the detailed expectations mismatch the utterance. This might be because 1) the event is unfamiliar, metaphorical, or in some way discrepant from expectation, 2) the sentence form is unfamiliar or novel or 3) a novel verb is used for which the system has no lexical entry. For example, utterances such as *I stared the man out of the room*, a new sentence form, or *Lucia zorched the book into the fire*, a new verb, would be uninterpretable.

The syntactic generation view has a complementary set of faults and virtues. Here the system can process

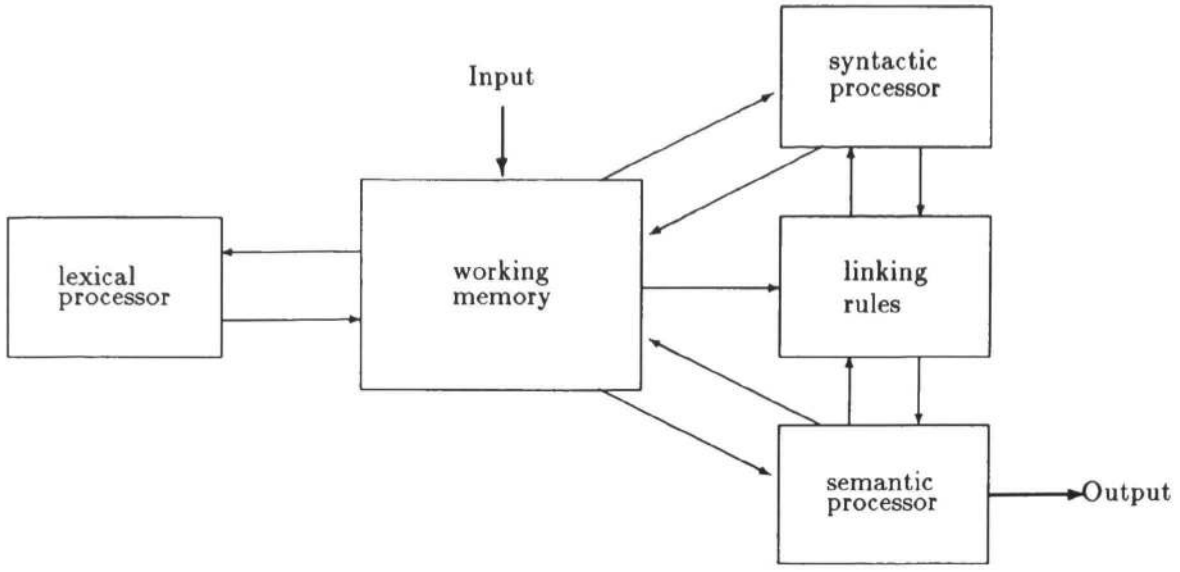

Figure 1: System Architecture

novel utterances as well as stereotyped ones. However, the generality is purchased at the cost of efficiency: a large number of structures might be consistent with the initial segments of input and a forest of possibilities generated. Since many forms can encode similar meanings a multitude of parsing rules are needed and a multitude of partial structures generated. Consider what would be needed to interpret *Which man did Pat punch in the mouth?*: 1) all possible functional roles for *which man* could be generated, 2) the identification of what element is being questioned must be delayed to the end of the sentence, and 3) many sets of rules are needed since the rules needed to process this sentence are not those used for related sentences such as *Pat punched the man in the mouth.*

People do not seem afflicted with either set of problems. New verbs and new uses of old verbs are constantly being introduced into the language; introspection and anecdote show people are quite capable of interpretation even under these circumstances. Psycholinguistic evidence shows that people do begin interpretation of a clause as information accrues and are not hamstrung prior to the final word.

Of course, the field is not as simply or as strongly polarized between the two possibilities as sketched here. Head-driven Phrase Structure Grammar (Pollard & Sag 1987) relies on syntactic information to drive interpretation, yet it specifies this in a very specific, word by word manner, not as general rules. (Lytinen 1986) proposed an approach in which semantic constraints are used to perform syntactic analysis. We believe this type of integration must be taken much further. Successful NLP systems will require a careful combinations of the available techniques and a careful analysis of what information needs to be supplied. Both syntax and semantics, both general and specific patterns, and both construction and memory will play comparably important roles in language interpretation. However, the following questions remain open: What syntactic distinctions map reliably onto subtle differences in meaning? When and in what form are highly specific forms of information used? What form need to be constructed and what can be retrieved from memory? Our research program divides in two parts: 1) a task analysis for identifying critical distinctions that constrain meaning and 2) a processing mechanism that can combine the multiple cue and constraints to provide an interpretation. Clearly both tasks are substantial and in both we are able to capitalize on many parts of analyses done by others. We focus here on describing an architecture that can derive lexical, semantic, and syntactic information and use these distinct but principled sets of cues to effectively drive sentence interpretation.

Architecture

An implementation of our approach needs to satisfy a number of requirements. First, it must provide a means of mutual communication between syntactic and semantic processing. Second, it must access and use word-specific information when it is available. Third, it must have a general information to resort to when word-specific information is absent. Fourth, both syntactic and semantic information must be used to guide structure composition.

We propose the architecture pictured in figure 1. The architecture has been modularized to reflect the structure of the task. Each module contains a distinct sort of information and distinct function. Within this model, data flows along two paths. Word-specific information flows from the lexical processor to the working memory where it is made available to the semantic, syntactic, and linking rules processor. Structural information flows through the linking rules processor. The syntactic processor and

the semantic processor communicate their structural decisions to one another by sending their products to the linking rules processor. The final interpretation is provided by the semantic processor. Below, we describe each of these modules as well as the system control.

Lexical Processor

The lexical processor provides the word-specific information to the system. It accepts a character string as input, and produces all the lexical entries associated with the word. Each lexical entry is denoted by a unique identifier. This allows individual components to ascertain which entry is being referred to in the working memory. Lexical entries contain syntactic, semantic, and correspondence information.

The syntactic information consists of a lexical category and an argument structure. For verb entries, the syntactic information includes a predicate argument structure which represents the argument structure of semantic functions (Rappaport & Levin 1988). Predicate argument structures perform two services. First, they denote the number of arguments a syntactic structure takes. For instance, the verb *put* has the following arguments

$$put: \begin{bmatrix} external & x \\ direct & y \\ indirect & z \end{bmatrix}$$

This representation indicates that the verb has a subject, a direct object, and a prepositional complement as arguments. Second, they denote the syntactic relations the verb licenses.

$$put: VP \rightarrow NP\ PP[loc]$$

This indicates the verb-phrase can be the mother node for both a noun phrase and a locative prepositional phrase.

The semantic information is either a semantic function or term. Verbs denote semantic functions. Consider the semantic entry for *put*

$$put: \begin{bmatrix} event & ACT(x\ y) \\ & effect\,[_{event}GO\,(y\ [_{path}TO\,[_{place}z]])] \end{bmatrix}$$

It can be roughly characterized as 'an actor acts upon an object causing it to move along some path'. Variables such as **y** are used to maintain identity relations among the semantic arguments. So for example in the lexical entry for **put**, the thing being acted upon **y** is also the thing being moved. These variables are also used to specify semantic constraints. Above the semantic term assigned to the variable **z** must be a **place**.

The correspondence information maintains the linkings between the syntactic and semantic variables specified in the lexical entry. *Put* specifies the following correspondences

$$put: \begin{array}{lcl} external & = & x\ in\ ACT(x\ y) \\ direct & = & y\ in\ ACT(x\ y) \\ indirect & = & z\ in\ GO\,(y\ [_{path}TO\,[_{place}z]]) \end{array}$$

Syntactic Processor

The syntactic processor contains general rules of syntax and constructs syntactic structures. It accepts as input the lexical entries from the lexical processor and any argument bindings already assigned by the linking rules processor (see below). It produces as output a parse tree and predicate argument bindings.

The syntactic processor is an adaptation of Abney's Licensing-Structure Parser (Abney 1989). It has two basic modes of operation: attachment and argument binding. The attachment operation constructs parse trees. The argument binding operation fills the predicate argument structure. Each attachment operation evokes a binding operation updating the predicate argument structure of the mother node.

As an example, consider how parser processes the follow phrase segment

put Mike ...

The *put* is the head of the verb phrase, and *Mike* is a noun phrase. An attachment is licensed by **put**

$$put: VP \rightarrow V\ NP$$

So the noun phrase *Mike* is attached to the verb phrase and assigned to **y**, and the processing of this fragment is complete. In cases where the predicate argument structure for *put* is not available, the syntactic processor invokes a general set of rules that license argument attachment (i.e., lexical redundancy rules).

The syntactic processor must check for internal inconsistency. Parsing inconsistencies are simply misparses. Misparses are identified with the aid of a parsing state. This state is used to indicate how much of the phrase has been parsed and what the possible continuations are. When the input is not a possible continuation, a misparse has occurred.

Semantic Processor

The semantic processor contains the general semantic information and constructs semantic representations. It accepts as input lexical entries from the lexical processor and semantic function bindings from the linking rules processor. It produces both complete semantic interpretation for system output and partial products used by the linking rules processor.

Like the syntactic processor, the semantic processor combines partial information. The semantic processor has two modes of operation: semantic function argument binding and function composition. Function argument binding assigns values to semantic functions. As an example consider the processing of the following fragment

put Mike

$$\left\{ \begin{array}{l} put: \begin{bmatrix} event & ACT(x\ y) \\ & effect\,[_{event}GO\,(y\ [_{path}TO\,[_{place}z]])] \end{bmatrix} \\ Mike: [HUMAN\ ANIMATE\ THING] \end{array} \right\}$$

Since **Mike** has no arguments, the possible assignments are to an argument of **put**. There are two possible assignments **y = Mike and z = Mike**. This assignment may be initiated by either the information provided by the linking rules processor or the semantic processor itself. Semantic function composition constructs complex function structures from two functions and will be discussed in an example following this section.

The semantic processor must check for internal inconsistency. There are two types of inconsistencies, violations of composition constraints and of argument binding constraints. The assignment **z = Mike** violates the *place* constraint on **z**; without a locative preposition, **Mike** does not designate a place. When a violation occurs, the semantic processor attempts to propose an alternative.

Linking Rules Processor

The linking rules processor executes the bi-directional mapping between syntax and semantics. Given a predicate argument binding (produced by the syntactic processor)as input, it produces a semantic function binding as output. Operating in the opposite mode, given function binding (produced by the semantic processor) as input, it produces an predicate argument binding as output. Linking rules allow the system to capitalize on regular correspondences between syntactic arguments and semantic roles, as in lexical redundancy rules. It has a general set of linking rules that can be extended by the correspondences specified in individual lexical entries. When these are provided, it prefers the execution of these lexically specific mappings over the general rules.

As an example consider the following linking rule

$$direct = y \; in \; ACT(x \; y)$$

If the direct object is bound by the syntactic processor, the linking processor can derive a semantic binding from the direct object assignment. The semantic processor executes this binding when it is provided. If it results in an semantic inconsistency, the semantic processor can propose an alternative (which is then communicated to the linking rules processor.) In the case of a conflict, the semantic processor takes precedence.

Working memory

Working memory executes two roles. It provides a means of communication between the lexical processor and the other processors, and it maintains the current lexical entries, a set of unique lexical items. The lexical processor provides all the lexical entries associated with each particular word. Processing is initiated with most-preferred lexical entries. If either processor finds a lexical entry unacceptable it may select another (i.e., the syntactic processor communicates its lexical category decisions, and the semantic processor specifies its selection of lexical entries.) Working memory must notify the processors of changes in the current lexical entries.

Communication

Information flows along two paths in the system. Word-specific information flows from the lexical processor to the working memory where it is made available to the semantic, syntactic, and linking rules processor, and structural information flows through the linking rules processor to the semantic and syntactic processors. Consider flow of information during the processing of the text fragment we have been pursuing in this section.

1. *lexical processor* $\rightarrow$
   $[Put.1][Mike.1]$
2. *syntactic processor* $\rightarrow$
   $direct = [Mike.1]$
3. *linking rules processor* $\rightarrow$
   $y/ACT/EVENT = [Mike.1]$
4. *semantic processor* $\rightarrow$
   $\begin{bmatrix} & ACT(...Mike.1) \\ _{event} & effect\,[_{event}GO\,(Mike.1...)] \end{bmatrix}$

In step 1, the lexical processor produces the lexical entries for *put* and *Mike*. In step 2, the syntactic processor assigns [Mike.1] to the direct object position. This is communicated to the linking rules processor. This processor produces the semantic argument specification in step 3. In step 4, the semantic processor makes the assignment and executes the inference that Mike is both the thing moving and thing being acted upon.

Global consistency is maintained by keeping the semantic and syntactic processors internally consistent and requiring that the semantic and syntactic processors be consistent with the linking rule products. In cases where the products of the linking rule processor conflicts with the state of either the semantic or syntactic processors, the linking rule products take precedence; linking rule products can only be overridden when they cause syntactic or semantic inconsistencies or violate semantic preferences.

Novelty Example

As an example of how such information could be used to process novel verbs, consider

Lucia zorched the book into the fire.

Since the predicate argument structure for *zorched* is unknown, the syntactic processor must construct the parse tree with the lexical redundancy rules, producing

$$\begin{bmatrix} external & [Lucia.1] \\ direct & [book.1] \\ indirect & [into\,[fire.1]] \end{bmatrix}$$

The linking rules processor executes the mappings which are ambiguous. The subject-direct object pair could correspond to a relation of possession or causality.

$$x/ACT/EVENT = [Lucia.1]$$
$$x/HAVE/STATE = [Lucia.1]$$

$$y/ACT/EVENT = [book.1]$$
$$y/HAVE/STATE = [book.1]$$

$$z/GO/EVENT = TO\,(IN\,([fire.1]))$$

Given the mappings, the semantic processor attempts to compose the predicates. The mapping ambiguity is resolved by the constraints on function composition, when the partial results are combined. The information associated with *into* is an effect of **GO**ing and this must co-occur with actions; hence, the **HAVE** mapping (a nonaction) can be ruled out. Only effects such as **GO** co-occur with actions, so the **HAVE** mapping can be ruled out.

$$\begin{bmatrix} & ACT([Lucia.1]\ [book.1]) \\ _{event} & effect\,[_{event}GO\,([book.1]\ TO\,(IN\,([fire.1])))] \end{bmatrix}$$

The resulting interpretation refers to an event in which Lucia's acting on the book causes it to go into the fire.

Summary

Our approach to processing provides a means of handling novelty. It also helps with some other difficult problems in natural language comprehension. Consider the case of prepositional attachment ambiguity. Giving the semantic processor the ability to propose structure allows it not only to identify semantic anomalies in the attachments proposed by the syntactic processors but also propose alternative bindings. It also provides a means of making early commitments in wh-questions. Rather than waiting for a gap in the parse tree, the syntactic processor can tentatively assign the wh-phrase to a syntactic argument. This assignment is sent to the semantic processor, allowing it to evaluate the assignment and to propose a new one if the assignment does not satisfy the semantic constraints. Finally, it aids in resolving lexical ambiguity. Changes to the lexical semantic representation of verbs can be partial. We need not discard the assignments that have already been made, if they still hold for the altered verb representation.

References

Abney, S. 1989. A Computational Model of Human Parsing. *Journal of Psycholinguistic Research* 18(1):129-144.

Birnbaum, L. & Selfridge M. 1981. Conceptual Analysis of Natural Language In R. Schank & C. Riesbeck (Eds.), *Inside Computer Understanding.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Rappaport, M. & Levin, B. 1988. What to do with Theta-roles. In W. Wilkins (Ed.), *Syntax and Semantics, Volume 21: Thematic Relations.* New York: Academic Press.

Lytinen, S. 1986. Dynamically combining syntax and semantics in natural language processing. *Proceedings of the Fifth National Conference on Artificial Intelligence,* Los Altos, Ca.: Morgan-Kaufman. 574-578.

Peterson, J. & Billman D. 1990. I'm With her and the butler did it With the Knife. Unpublished Manuscript.

Pollard, C. & Sag, I. 1987. *An Information-based Syntax and Semantics, Volume 1: Fundamentals.* Stanford, CA: Center for the Study of Language and Information.

Riesbeck, C. & Martin, C. 1986. *Towards Completely Integrated Parsing and Inferencing.* Proceedings of the Eighth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum Associates.

Woods, W. 1978. *Semantics and Quantification in Natural Language Question Answering.* In M. Yovits (Ed.), *Advances in Computers, Volume 17,* 2-64. New York: Academic Press.