Semantics Parsing Revisited or How a Tadpole Could Turn into a Frog

Wai-Kiang Yeap

1 Natural Language Research Group, Institute for Information Technology Research, Auckland University of Technology, Auckland, New Zealand
{wai.yeap@aut.ac.nz}

Abstract

Early AI researchers got it right; that language is about semantics parsing. However, instead of focusing on developing the necessary framework to support semantics parsing, these researchers have focused mainly on how word meanings are represented and used. In this paper we developed one such framework and argued that it provides a more natural explanation as to how children might acquire their first language. A LISP program has been implemented to test the idea developed.

1 Introduction

When AI researchers began developing parsers for natural language in the early days, there was a strong feeling among some that parsing should be done without the use of grammar rules. These AI researchers (for example, Riesbeck, 1975; Ritchie and Thomson, 1983; Schank, 1972; Wilks, 1975) called for the development of “semantics” parsers. They envisioned such a parser would, upon reading a word, retrieve its meanings immediately and use them to interpret the sentence. In their implementations, they emphasized the development of a rich description of words to provide a sufficient context for interpreting sentences. However, without the use of grammar rules, these processes become driven solely by what one knows about each word and this presents two major problems. First, how could we know every possible context beforehand and second, how could we select the right context each time, almost effortlessly and without first perceiving the whole sentence?

These two problems, that of representation and control, remain unsolved to date and the above idea of a semantics parser has never been implemented successfully. The idea has now long been discarded among AI researchers in general. One reason for its unsuccessful implementation, in my view, is precisely this lack of a proper framework for utilizing the word meanings provided. Of course, word meanings are needed. Otherwise, how else could a sentence be interpreted? But, to claim, as many of these researchers did, that one could extract the meanings of a sentence directly from knowing what each word means is to ignore the very linear nature of the input sentence.

In this paper, I want to rekindle the idea of a semantics parser. These AI researchers got the idea right and that is, language is about semantics parsing. However, the emphasis should not primarily be about deriving meanings of sentences directly from meanings of words in the sentences without the use of any grammar rule. Rather, it should be about discovering how the entire language process itself (albeit, the parser) could emerge from one’s progressive grasp of word meanings. In other words, the control knowledge (grammar rules) is derived from, and is very much a part of, how one understands what each word means. This observation stems from a very common observation made by language researchers studying how children acquire their first language. That is, children first learn how to use single words to communicate their ideas and later shift to the use of multiple words. This observation tells us that the child has first learned the rule “one word says all” and then progressively modified that rule to “a sentence (i.e. a series of words) is better”. From behavioral data, the obvious hint of the algorithm used is that adjacent words are combined together, using some left/right attachment rules. The remaining puzzle then is: How could such a simple initial process be extended so that the full complexity of language use is explained?

Pinker’s (1984, 1987, 1989a, 1989b) discussions of Baker’s paradox (and others working in this area) have highlighted that this puzzle cannot be solved as a classification problem whereby the regular rules governing language use could be detected from the input patterns (i.e. sentences). The argument is that the input is not rich enough, famously known as “the poverty of the stimulus” argument. Pinker suggested an alternative solution which he refers to as a “semantic bootstrapping approach”. The central idea of this approach is that some innate linking rules exist to “inform” a child the basic syntactic categories of words via their semantic properties. Thus, on learning that a word is referring to an agent in the current scene, the word would become the subject of the input sentence. This is because an innate linking rule exists which says that agent maps onto subject. Given these rules, Pinker argued, a child can then begin to shape her internal grammar to the one being experienced.

My approach does not require postulating the presence of any innate linking rules. Instead, it depends on developing cognitively plausible methods to increase the power of the basic mechanism as observed in children’s first transition into the use of multiple words. This has the advantage of being a more natural solution to the paradox observed. In section 2, I will describe a very basic framework for semantics parser which can process simple sentences. The parser described might be slightly more powerful than the one present in children when they first learn how to utter multi-word phrases. None-
theless I will use it as the starting point for our discussion. In section 3, I show how this basic framework can be extended in different ways to deal with more complex sentences. The aim here is neither to describe the algorithms in depth nor to present the complete program that has been implemented. Rather, the aim is to present to the readers with sufficient examples to convince them that “the tadpole could indeed be turned into a frog”. In section 4, I conclude with a brief discussion of the implications of the approach proposed.

2 A Basic Framework for Semantic Parsing

To communicate using language is to exchange abstract ideas. Thus, a semantics parser manipulates abstract ideas or semantics objects. Each word/symbol understood is assumed, for simplicity, to point to one or more semantics objects. The information contained in each semantics object is a subset of what an individual knows about its corresponding word. The individual’s knowledge, on one hand, is a (vast) web of related information and a semantic object, on the other, merely provides a context for accessing that web of information. Consider the following representation of a semantics object that will be used throughout our discussions:

(person* (:sex male*) (:height tall*))

Words followed by an asterisk are pointers to the respective part of one’s web of information. When we need to know more about the concept of a person according to this semantics object, we follow its pointer person* and retrieve information from the web which is related to a tall male person. How the web itself is represented is left as an open design question and is not our concern here. Using the above representation of semantics objects, I can now describe a basic parser that manipulates these objects.

When one begins to use multiple words to express oneself, the semantics objects need to be enriched to show how the respective semantics objects are combined when perceived. It has been observed that children pay much attention to local cues in the input sentence such as word ordering and case markings (Hirsh-Pasek and Golinkoff, 1996). This suggests that these cues should be used to provide the basic information for combining semantics objects. I use two variables, ?L (left) and ?R (right), inside a semantics object to indicate to where adjacent semantics objects from the appropriate side are attached. An example of a semantics object with these variables is as follows:

Eat: (eat* (:who ?L) (:what ?R)).

By using a stack to process these semantics objects, the basic algorithm for processing simple sentences is as shown below:

Interpret():

1. Perceive the next word and generate its semantics object.
2. If it has a ?L, do: {Pop a semantics object from the stack and use it to replace the ?L. Push the result back onto the stack. Go to step 6}.
3. If it has a ?R, push it onto the stack and go to step 6.
4. If it has no ?L or ?R, check if the semantics object on the stack has a ?R. If it has, do: {Use the incoming semantics object to replace the ?R. Go to step 6}.
5. Push the current semantics object onto the stack.
6. Repeat the process.

Below shows a simple sentence parsed by the algorithm above:

(1) I told her.

[told* (:actor (I* (:person)))
   (:what (her* (:sex (female*))))]

The dictionary entries are:

I: (I* (:person))
Her: (her* (:person) (:sex (female*)))
Told: (told* (:who ?L) (:what ?R))

The next section shows how the above framework can be extended to deal with more complex sentences. It is worthwhile emphasizing here that this work is not concerned with the use of semantics to process language. Thus nothing will be said about how word meanings are represented and used in the parsing process. Rather it is about the kind of grammar one might acquire when experiencing first language use.

3 Extending the Power of the Basic Parser

How could the basic algorithm be extended to handle the complex variations in language and in ways which do not require information not made available as input to the (child’s) process? The solution lies in one’s ability to extend the initial ?L/?R labels for more sophisticated processing of semantics objects.

A system has been implemented which shows how this could be done for a variety of grammatical constructs known in the English language. The basic approach is as follows. As the basic algorithm shows, if the input contains a ?R, it has to be pushed onto the stack and one waits for the next semantics object to appear. When that happens, the basic algorithm uses the incoming semantics object to initialize the ?R variable in the semantics object on the stack. Let us refer to this ?R as a ?R+ variable. The ‘+’ indicates information has to be added to the semantics object holding this variable from its right hand side (i.e. the next input). One straightforward means to extend the algorithm is to realize that in addition to information being added to this semantics object from the right, one could subtract information.
from this semantics object and add it to the semantics object on its right. Hence, this gives rise to the label, ?R-.
Such a label would capture, among others, the grammatical category known as adjectives. Once this is established, one could further introduce ?R--, indicating information is to be subtracted from the semantics object holding this variable and added immediately to the next semantics object on its right. This label would capture, again among others, the grammatical category known as adverbs that modify adjectives.

The next section describes, in more details, some of the possible ways to extend the ?L label. Due to a lack of space, a complete description of the available system will not be described here.

3.1 Handling Objects from the Left

Like ?R+, the ?L+ is used for handling the basic verbs. We extend the ?L+ to ?L- for handling adverbs and ?L-* for handling prepositions.

3.1.1 Handling ?L-

Like the ?R-, a ?L- label is used to subtract information from the incoming semantics object and add it to the one existing on the stack. This label represents the class of words known as adverbs (that modify a verb). Unlike conventional wisdom, it is interesting to note that the processing of such a word does not require the detection of verbs in the sentence. For example, consider the processing of the following two sentences:

(2) I answered the questions foolishly.  
(3) I foolishly answered the question.

Their respective output as produced by the program is:

[answered* (:actor (I* (:noun)))  
(:what (questions* (:noun))  
(:manner (foolishly*)))]

[answered* (:actor (i* (:noun))  
(:manner (foolishly*)))  
(:what (questions* (:noun)))  
(:manner (foolishly*)))]

The word, foolishly, is represented as:

(310x800) ?L- (:manner (foolishly*)))

It is only when the semantics object is being interpreted that we get the correct meaning. When the adverb is attached to a noun, it indicates that is how the noun will perform any action attached to it. However, when the adverb is attached to the verb, it is indicates that is how that action is being performed.

3.1.2 Handling ?L-*

What could lead to a more sophisticated ?L-type of processing? One possibility is the observation that when a ?L- variable is encountered, the stack might need to be processed in order to produce the correct semantics object. That is, it is not always the case that the first semantics object on the stack is the correct object to use. Furthermore, one might encounter that there are no semantics object on the stack. In this case, one could pass its information to the right, even though this is a ?L-type object. Some example outputs are shown below:

(4) I saw the car of John in town.

[saw* (:actor (I* (:noun)))  
(:what (car* (:noun)) (:modifier (the*)))  
(:of* (John* (:name (john*))))  
(:in* (town* (:noun))))]

Note that in the example above, “in” is not attached to “John” even though the semantics object of “John” is found on top of the stack.

(5) I saw her in the park with a telescope.

[saw* (:actor (I* (:noun)))  
(:what (her* (:sex (female*)))  
(:in* (park** (:noun) (:modifier (the*)))))  
(:with* (telescope* (:noun))))]

The above example demonstrates the multiple attachment of prepositions.

(6) With a guitar he sang a song.

[sang* (:actor (he* (:sex (male*)))  
(:with* (guitar* (:noun)) (:modifier (a*))))]

The above example demonstrates a situation where no semantics object is found on the stack when a ?L-* is encountered.

The last example is interesting in that the attachment to the right is not due to a change in the labeling scheme but as a result of reasoning with semantics objects on hand. When the word, with, is encountered and no se-
mantics object is found on the stack, the result is that the stack now contains the semantics object of with itself:

```
[?L* (:with* ?R+)]
```

Having processed the phrase, with a guitar, one of the stacks now contains:

```
[?L* (:with* (guitar* (:noun) (:modifier (a*))))]
```

When the word, he, appears, one has to reason how to deal with a semantics object with an unfilled ?L-* and an incoming semantics object without a label. One logical conclusion would be to pass the information of the semantics object with the ?L-* to the incoming semantics object. The result becomes:

```
[he* (:sex (male*)) (:with* (guitar* (:noun) (:modifier (a*)) ))]
```

This is semantics parsing.

Below shows an example of a complex sentence being parsed by the system:

```
(7) When transit NZ rolled out its draft ten-year state highway construction plan in January this year Auckland motorists were relieved while the rest of the country was disappointed.
```

\begin{verbatim}
[while*
 (:ms1 (when*)
   (:ms1 (rolled* (:actor (NZ * (:noun)) (:x-words (transit*) (NZ *))))
   (:out* (plan* (:noun)) (:x-words (draft*) (ten-year*) (state*) (highway*) (construction*) (plan*)) (:modifier (its*)) (:in* (:time*
     (:when (january*)) (:when (this*)) (year*)))))))
(:ms2 (motorists* (:x-words (auckland*) (motorists*)) (:were* (relieved*))))]
\end{verbatim}

4 Discussions

I began by pointing out the lack of a suitable framework for the semantic processing of human languages. I then developed one such framework based upon a strategy of left/right attachment of words. This strategy is chosen because it was hypothesized that children could have learned a similar strategy when they begin communicating using a couple of words. If one could extend this strategy to process adult sentences, then it is possible that the framework developed would be useful for explaining how language works. The extension has been completed and briefly reported in this paper. I now discuss some of the lessons learned.

It is important to distinguish between a framework for semantic processing and a parser which, at some stage, uses semantic knowledge. The latter is seen in many classical implementations of natural language parser by AI researchers. Typically, they implemented some form of a grammar rule and when the output of their parser is ambiguous, they turn to the use of semantic knowledge to resolve ambiguity. In a framework for semantic processing, (meanings of) words are combined only if they form a meaningful entity. What the framework provides is information about which pair of words one should be combining. Note that in a semantic parser, it is more appropriate to think in terms of meanings of words rather than words. Thus, for example, in the processing of the first two words of sentence, “I answered the questions foolishly” (see sentence (2) above), we get:

```
[answered* (:actor (I* (:noun))) (:what ?r+)]
```

The above is interpreted as the meaning of the phrase, “I answered” and not that the phrase is legally correct.

There exist some traditional parsers whose implementation appears to be closely related to the approach proposed here. The most obvious one is categorial grammar. This approach is similar to the one developed here because of the following key characteristics of categorial grammar (Wood, 1993):

1. The lexical entries of words encode all the information about how words are combined into phrases; there is no separate component of syntactic rules (Karttunen, 1989).
2. Its basic rule is about combining words from its left and right.

On the surface, categorial grammar appears to be similar. However, one should note that the lexical entries of words in categorial grammar consist of instructions for combining words legally according to a grammar of the language. Two examples are shown below (reproduced from Wood, 1993):

```
\begin{verbatim}
John likes Jane
n (n\text{s}/n) n
\end{verbatim}
```
Chris gave a fish to Tigger

\[ \text{np} \ (\text{ns}/\text{np})/\text{pp} \ \text{np} \ \text{pp} \]

where \((\text{ns})/\text{n}\) is interpreted as waiting for a noun term to appear on its right and when it is found, \((\text{ns})/\text{n}\) becomes \(\text{n/s}\). The latter is interpreted as waiting for a noun term to appear on its left and when it is found, \(\text{n/s}\) becomes \(\text{s}\) (a legal sentence).

Note that although categorial grammar employs the technique of left/right attachment of words, it is really an alternative way for implementing a formal grammar for processing natural languages. It does not address the question as to how children could acquire grammatical terms such as “pp” and “s” above and the rules such as “\((\text{ns})/\text{n}\)” and “\((\text{s/np})/\text{pp})/\text{np}\)”. By contrast, this work is not about presenting an alternative approach to implementing a parser based upon a formal grammar. The left/right attachment of words is not applied to form syntactic categories. Rather, it is applied to provide a framework for combining word meanings.

Although I have shown that the basic left/right strategy can be extended to process complex adult sentences, one question still remains: is this approach learnable by children under the “poverty of the stimulus” environment. Ultimately, empirical studies of children acquiring their first language would need to be conducted in order to confirm that children indeed are using such a framework for learning languages. However, for now, I will briefly point to some characteristics of this framework that are useful for learning:

1. Modification of each “rule” within this framework could be done locally. That is, no universal set of grammar rules is learned;
2. Extension/addition of each rule is done in a modular fashion; thus allowing performance not to be affected drastically; and
3. Learning of each rule is done via a process of discovering the meanings of each word and not its syntactic category.

The first two characteristics would allow the child to make mistakes while learning the complex rules of language. The third characteristic would be important to overcome the “poverty of the stimulus” environment in which a child acquires their first language.

5 Conclusion

In this paper, I outlined a new framework for processing human languages. The framework is developed using a strategy that children appear to have learned when they begin to communicate using a couple of words.

I have demonstrated how this simple strategy can be extended to deal with the more complex adult sentences. Two examples are shown in this paper, “L-” and “L*”. Others that I have already implemented include the processing of conjunctions, connectives, infinitive-to, wh-words, questions, and the different kinds of verbs. The program has also been incorporated as part of a system for the visualization of text (Yeap, Reedy, Min and Ho, 2005).

Acknowledgments

This project was supported initially by a grant from the NZ Law Foundation and is now supported by a grant from the NZ Government New Economy Research Fund.

References