PERSPECTIVES ON PARSING ISSUES

Christopher K. Riesbeck
Yale University

COMPUTATIONAL PERSPECTIVE

IS IT USEFUL TO DISTINGUISH PARSING FROM INTERPRETATION?

Since most of this position paper will be attacking the separation of parsing from interpretation, let me first make it clear that I do believe in syntactic knowledge. In this I am more conservative than other researchers in interpretation at Berkeley, Carnegie-Mellon, Columbia, the universities of Connecticut and Maryland, and Yale.

But believing in syntactic knowledge is not the same as believing in parsers! The search for a way to assign a syntactic structure to a sentence largely independent of the meaning of that sentence has led to a terrible misdirection of labor. And this effect has been felt on both sides of the fence. We find ourselves looking for ways to reduce interaction between syntax and semantics as much as possible. How far can we drive a purely syntactic (semantic) analyzer, without sneaking over into the enemy camp? How well can we disguise syntax (semantics) as semantics (syntax)? How narrow a pipe between the two can we get away with? What a waste of time, when we should be starting with bodies of texts, considering the total language analysis picture, and looking for what kinds of knowledge need to interact to understand those texts.

If our intent in overextending our theories was to test their muscle, then I would have no qualms. Pushing a mechanism down a blind alley is an important way to study its weaknesses. But I really can't accept this Popperian view of modern computational linguistics. Mechanisms are not driven beyond their limits to find those limits, but rather to grab territory from the other side. The underlying premise is "If our mechanism X can sometimes do task A, then there is no need for someone else's mechanism Y." Occam's razor is used with murderous intent.

Furthermore, the debate over whether parsers make sense has drastically reduced interaction between researchers. Each side sees the other as avoiding fundamental issues, and so the results from the other side always seem to be beside the point. For example, when Mitch Marcus' explains some grammatical constraint as syntactic processing constraints, he doesn't answer any of the problems I'm faced with. And I'm sure Mitch has no need for frame-based, domain-driven partial language analysis techniques.

This situation has not arisen because we have been forced to specialize. We simply don't know enough to qualify for an information explosion yet. Computational linguistics doesn't have hundreds of journals in dozens of languages. It's a young field with only a handful of people working in it.

Nor is it the case that we don't have things to say to each other. But — and here's the rub — some of the most useful things that each of us knows are the things that we don't dare tell. By that I mean that each of us knows where our theories fall apart, where we have to kludge the programs, fudge the inputs, or wince at the outputs. That kind of information could be invaluable for suggesting to the others where to focus their attentions. Unfortunately, even if we became brave enough to talk about, even emphasize, where we're having problems, the odds are low that we would consider acceptable what someone else proposes as a solution.

IS SIMULATION OF HUMAN PROCESSING IMPORTANT?

Yes, very much so, even if all you are interested in is a good computer program. The reason why was neatly captured in Principles of Artificial Intelligence: "language has evolved as a communication medium between intelligent beings" (Nilsson, p. 2). That is, natural language usage depends on the fact that certain things can be left ambiguous, left vague, or just left out, because the hearer knows almost as much as the speaker. Natural language has been finely tuned to the communicative needs of human beings. We may have to adapt to the limitations of our ears and our vocal chords, but we have otherwise been the masters of our language. This is true even if there is an innate universal grammar (which I don't believe in). A universal grammar applies few constraints to our use of ellipsis, ambiguity, anaphora, and all the other aspects of language that make language an efficient means for information transfer, and a pain for the programmer.

Because language has been fitted to what we do best, I believe it's improbable that there exist processes very unlike what people use to deal with it. Therefore, while I have no intention of trying to model reaction time data points, I do find human behavior important for two kinds of information. First, what do people do well, how do they do it, and how does language use depend on it? Second, what do people do poorly, and how does language use get around it?

The question "How can we know what human processing is really like?" is a non-issue. We don't have to know what human processing is really like. But if people can understand texts that leave out crucial background facts, then our programs have to be able to infer those facts. If people have trouble understanding information phrased in certain ways, then our programs have to phrase it in ways they can understand. At some level of description, our programs will have to be "doing what people do," i.e., filling in certain kinds of blanks, leaving out certain kinds of redundancies, and so on. But there is no reason for computational linguists to worry about how deeply their programs correspond to human processes.

WILL PARALLEL PROCESSING CHANGE THINGS?

People have been predicting (and waiting for) great benefits from parallelism for some time. Personally, I believe that most of the benefits will come in the area of interpretation, where large-scale memory search, such as Scott Fahlman has been worrying about, are involved. And, if anything, improvements in the use of semantics will decrease the attractiveness of syntactic parsing.

But I also think that there are not that many gains to be had from parallel processing. Hash codings, discrimination trees, and so on, already yield reasonably constant speeds for looking up data. It is an inconvenience to have to deal with such things, but not an insurmountable obstacle. Our real problems at the moment are how to get our systems to make decisions, such as "Is the question 'How many times has John asked you for money?' rhetorical or not?" We are limited not by the number of processors, but by not knowing how to do the job.

THE LINGUISTIC PERSPECTIVE

HAVE OUR TOOLS AFFECTED US?

Yes, and adversely. To partially contradict my statements in the last paragraph, we've been overly concerned with how to do things with existing hardware and software. And we've been too impressed by the success computer science has had with syntax-driven compilation of programming languages. It is certainly true that work on grammars, parsers, code generators, and so on, have changed compiler generation from massive multi-man-year endeavors to student course projects. If compiler technology has benefited so much from syntactic parsers, why can't computational linguistics?

The problem here is that the technology has not done what people think it has. It has allowed us to develop modern, well-structured, task-oriented languages, but it has not given us natural ones. Anyone who has had to teach an introductory programming course knows that. High-level languages, though easier to learn than machine language, are very different from human languages, such as English or Chinese.

Programming languages, to readjust Nilsson's quote, are developed for communication between morons. All the useful features of language, such as ellipsis and ambiguity, have to be eliminated in order to use the technology of syntax-driven parsing. Compilers do not point the way for computational linguistics. They show instead what we get if we restrict ourselves to simplistic methods.

DO WE PARSE CONTEXT-FREELY?

My working assumption is that the syntactic knowledge used in comprehension is at most context-free and probably a lot less, because of memory limitations. This is mostly a result of semantic heuristics taking over when constructions become too complex for our cognitive chunking capacities. But this is not a critical assumption for me.

INTERACTIONS

Since I don't believe in the pure grammatical approach, I have to replace this last set of questions with questions about the relationship between our knowledge (linguistic and otherwise) and the procedures for applying it. Fortunately, the questions still make sense after this substitution.

DO OUR ALGORITHMS AFFECT OUR KNOWLEDGE STRUCTURES?

Of course. In fact, it is often hard to decide whether some feature of a system is a knowledge structure or a procedural factor. For example, is linear search a result of data structures or procedure designs?

CAN WE TEST ALGORITHMS/KNOWLEDGE STRUCTURES SEPARATELY?

We do indeed try experiments based on the shape of knowledge structures, independently of how they are used (but I think that most such experiments have been inconclusive). I'm not sure what it would mean, however, for a procedure to be validated independently of the knowledge structures it works with, since until the knowledge structures were right, you couldn't tell if the procedure was doing the right thing or not.

WHY DO WE SEPARATE RECOGNITION AND PRODUCTION?

If I were trying to deal with this question on grammatical grounds, I wouldn't know what it meant. Grammars are not processes and hence have no direction. They are abstract characterizations of the set of well-formed strings. From certain classes of grammars one can mechanically build recognizers and random generators. But such machines are not the grammars, and a recognizer is manifestly not the same machine as a generator, even though the same grammar may underlie both.

Suppose we rephrase the question as "Why do we have separate knowledge structures for interpretation and production?" This presupposes that there are separate knowledge structures, and in our current systems this is only partially true.

Interpreting and production programs abound in ad hoc procedures that share very little in common near the language end. The interpreters are full of methods for guessing at meanings, filling in the blanks, predicting likely follow-ups, and so on. The generators are full of methods for eliminating contextual items, picking appropriate descriptors, choosing pronouns, and so on. Each has a very different set of problems to deal with.

On the other hand, our interpreters and generators do share what we think is the important stuff, the world knowledge, without which all the other processing wouldn't be worth a partridge in a parse tree. The world knowledge says what makes sense in understanding and what is important to talk about.

Part of the separation of interpretation and generation occurs when the programs for each are developed by different people. This results in unrealistic systems that write what they can't read and read what they can't write. Someday we'll have a good model of how knowledge the interpreter gains about understanding a new word is converted to knowledge the generator can use to validly pick that word in production. This will have account for how we can interpret words without being ready to use them.

For example, from a sentence like "The car swerved off the road and struck a bridge abutment," we can infer that an abutment is a noun describing some kind of outdoor physical object, attachable to a bridge. This would be enough for interpretation, but obviously the generator will need to know more about what an abutment is before it could confidently say "Oh, look at the cute abutment!"

A final point on sharing. There are two standard arguments for sharing at least grammatical information. One is to save space, and the other is to maintain consistency. Without claiming that sharing doesn't occur, I would like to point out that both arguments are very weak. First, there is really not a lot of grammatical knowledge, compared against all the other knowledge we have about the world, so not that much space would be saved if sharing occurred. Second, if the generator derives it's linguistic knowledge from the parser's data base, then we'll have as much consistency as we could measure in people anyway.

REFERENCE

Nilsson, N. (1980). Principles of Artificial Intelligence. Tioga Publishing Co, Palo Alto, California.