Site Report

Computational Linguistics at BBN Labs

Andrew Haas
BBN Labs, Cambridge, Massachusetts

Our research has three chief goals: to handle connected discourse instead of sentences in isolation, to discover the intentions behind discourse as well as its literal meaning, and to handle failures (by the program or its user) gracefully. To achieve these we must make progress in other areas – planning and knowledge representation. I will survey our work in computational linguistics and then consider the supporting efforts in related areas of AI.

Personnel: Robert Bobrow, Madeleine Bates, Candace Sidner, N. S. Sridharan, Remko Scha, Ralph Weischedel, Marc Vilain, Andrew Haas, David Stallard, Marie Macaissa, Margaret Moser, Robert Ingria, Jos De Bruin, Bradley Goodman, James Schmolze.

1. Natural Language

1.1 Parsing and semantic interpretation

How can semantics be incorporated into parsing without sacrificing the generality and transportability of the grammar and the parser? Semantic grammars are efficient because they mix syntax and semantics. Syntactic results are checked for semantic correctness continuously, and no time is wasted building syntactic structure that later turns out to be semantically absurd. Semantic grammars also pay a high price for this efficiency. Their syntactic knowledge is hard to modify because it is not stated in a general way – the program knows about "Noun Phrases Denoting People", and "Noun Phrases Denoting Books" and so on, but not about noun phrases. And since the syntactic and semantic knowledge are intertwined, there is no hope of transporting the syntactic knowledge to a new domain.

We have been developing a parser that cleanly separates syntax and semantics, but is as efficient as a semantic grammar. We achieve this partly through a control structure called a cascade that links the parser and semantic interpreter, partly through proper representation of semantic knowledge. Like others, our parser calls semantics to check that newly built constituents are OK; but it also calls semantics each time it attaches a newly built constituent to a higher constituent. Thus semantic checking of a proposed attachment for a prepositional phrase does not wait until the clause is finished.

We represent a mixture of semantic and syntactic knowledge in a KL-2 taxonomy. Like a semantic grammar, our system has concepts such as "Noun Phrase Denoting a Person" and "Noun Phrase Denoting a Book". Unlike a semantic grammar, our system does not take them as primitive – they are defined in terms of concepts like "Noun Phrase", "Person", and "Denotation". An efficient algorithm uses these definitions to classify incoming constituents, and once they are classified we can retrieve the rules that build their semantic interpretations.

This work is being done by Robert J. Bobrow and Madeleine Bates, supported by David Stallard, Margaret Moser, and Robert Ingria.

1.2 Plans and discourse

We believe that the goals of understanding connected discourse and responding to unspoken intentions are closely related, because the structure of discourse reflects the intentions behind it. A discourse consists of units, and each unit expresses one intention of the user. To find the intentions one must divide the discourse into units.

We aim to build a plan parser – a program that takes a set of possible plans and a series of parsed sentences, and discovers the speaker's plans. The parser will use surface cues to detect the boundaries of discourse units – for example, sentences starting with "OK" or "Anyway". It will also rely on the speaker to make his meaning plain. If it reaches a point where several plans are consistent with the evidence, it doesn't undertake an elaborate search in an effort to rule some of them out. It assumes that the user will make his purpose clear later on, and waits for further input.

As an example, suppose the system is displaying a semantic net on the user's screen, and the concept "bird" is in the middle of the screen. We might get the following dialogue:

User: I can't fit a new concept below it. Can you move it up?

(System moves concept higher on the screen.)

User: OK, now put the concept "eagle" below it.

The word "OK" indicates that one of the user's intentions has been fulfilled, and marks the end of a unit of discourse concerned with that intention. "now" introduces a new unit, concerned with the user's intention to place the concept "eagle" below "bird". This work is being done by Candy Sidner.

I said above that discourse gets its structure from the intentions of the speaker. Discourse also follows certain rules of its own, and we are investigating this structure also. If I say "It's snowing", the word "it" does not refer to anything. It is there because English syntax demands that sentences have subjects. In the same way, discourse has a syntax independent of the user's plans and goals. In narrative, for example, successive clauses describe successive events. If we read "He went to the window and pulled aside the curtains", we understand that he pulled the curtains aside after going to the window. One might suppose that this is a matter of pragmatics rather than discourse rules – obviously one must go to the window before pulling aside the curtains. We can see this is wrong by reversing the clauses. "He pulled aside the curtains and went to the window" sounds strange because discourse rules tell us that he must have pulled aside the curtain before going to the window, but pragmatics rejects this. Thus we can see that discourse rules have a life of their own. We intend to devise a formal syntax and semantics of discourse along these lines. Ultimately we hope to combine both approaches to discourse, showing how speaker's intentions operate within the framework of discourse rules. This is the work of Remko Scha collaborating with Livia Polanyi of the University of Amsterdam.

1.3 Planning utterances

The first step in planning a response is to classify the current discourse situation. Perhaps the user has made an error, and the system must explain the problem; perhaps the system failed to understand the user, and needs clarification; perhaps we have a simpler situation, such as a question that demands a direct answer.

Attached to each class of situation is a content selector. This program will choose the system's discourse goals – goals like answering the user's question, informing the user that there's a problem, or asking the user what he meant. Its choice will depend on the user's plans and the information the user needs to carry out those plans. Suppose, the user asks the system to display something, but there is no room. The system shouldn't just inform the user of the problem – it should offer some solutions. If it knows the user's current plan it may realize that some of the information on the screen is no longer useful, and offer to erase it.

When the system has decided what speech acts to perform, it must organize them into a coherent piece of discourse. They must be placed in order, and appropriate discourse markers added. To produce smooth text we must refer to objects by pronouns, not names or descriptions, as much as possible; but we can't refer to an object by a pronoun until it's brought into focus. We will order the sentences so that we can use pronouns as much as possible. We can also convey some information by the ordering of sentences. Consider the following dialogue:

User: I want to display the concept "bird".

System: There's no room on the screen. I can move the current display up, or erase it and save it.

The user understands that the system's first sentence describes a problem, and the second offers two solutions. The order of the sentences is crucial – think how odd it would sound if the system had used the same two sentences in the other order.

When the information has been ordered, and discourse markers inserted, we have a message plan. This plan will be turned into English text by NIGEL – a generation program written by William Mann of USC/Information Sciences Institute. This work is being done by Margaret Moser and Robert Ingria.

1.4 Ill-formed input

Many workers agree that handling ill-formed input is a crucial problem for natural language research. They disagree about what "ill–formed" means - is it ill-formed by native speakers' standards, or by the program's standards? We take the latter position. This means that some of what we treat as ill-formed may be perfectly good by human standards. This seems a little odd, but the same thing happens to people when they talk to a speaker of another dialect. His language is well-formed by the standards of his community, but until you learn their dialect you must treat it as ill-formed by the only standard you know.

We intend to handle ill-formed input by using discourse context, which consists of the plans and goals conveyed by the speaker's discourse. The speaker has high-level goals (like registering for a class, or finding out where the best housing is) and discourse goals (like answering a question, or clarifying something he doesn't understand). For example, consider the following dialogue:

User: I'm interested in housing in the Rolling Hills area. What grade school do they attend?

System: P.S. 32.

User: Any swimming pools nearby?

Suppose the user's last utterance is beyond the system's ability to parse. It knows that the user is trying to decide whether he should buy a house in Rolling Hills, and this gives a clue to the meaning of the utterance. This work is being done by Ralph Weischedel at BBN, by Sandra Carberry (working on ellipsis) and by Lance Ramshaw (studying impossible requests), both at the University of Delaware.

We are working on another kind of ill-formed input: definite descriptions that contain errors. We have studied a large corpus of natural dialogues, in which one person tells another how to assemble a toy water pump. It's hard to describe the parts of the pump accurately, but the assembler has the parts in front of him and often manages to find the right part despite the other person's mistakes. These examples suggest a strategy of relaxation: weakening the given description until it fits one of the known objects. The trick is deciding which parts of the description to weaken, and how. This requires knowledge of common errors – for example, color names are easily confused. We have a taxonomy that includes a large class of errors, for noun phrases and other constructs. We plan to complete the implementation of an algorithm that corrects errors in definite descriptions, and to explore techniques for dealing with the other classes of errors. This work is Bradley Goodman's.

2. Knowledge Representation and Planning

Progress in these areas requires progress in the closely related areas of knowledge representation and planning. To assign semantic interpretations to sentences we need a knowledge representation language rich enough to express the content of those sentences. To detect semantically impossible parses we need an inference engine complete enough to detect conflicts between proposed interpretations and the system's knowledge of the domain. In order to analyse and synthesize discourse, we need programs that plan utterances and that recognize the plans behind another's utterances. Finally, we must consider parallel computation – both for efficiency and for the insights that come from thinking in this way.

2.1 Hybrid systems

In knowledge representation, our goal is a domain-independent inference engine that handles an expressive language without sacrificing efficiency. We can do this with a hybrid system: one with several components, each specialized to a particular kind of inference, and each supplying the others with the information they need. The individual components can be efficient because they are specialized; the whole system is expressive because its language includes several kinds of representation devices. The trick is to make sure that each component supplies the others with the information they need, without drowning them in a flood of facts.

We have built a hybrid system called KL-TWO, with two components. One (called PENNI) is a version of David McAllester's RUP. It handles propositional logic, equality, and truth maintenance. The other component (NIKL) handles inheritance in a hierarchy of properties. In NIKL we can assert that a husband is (by definition) a married man. If we then assert in PENNI that John is a man and John is married, the system will combine these two assertions into a single property, which sums up its knowledge of John. Using lambda notation, we can write the property as

(lambda x. (man x) & (married x))

NIKL now attempts to fit this property into its hierarchy of properties. It discovers that the property is identical to the known property of being a husband, and adds to PENNI's data base the statement

(man John) & (maried John) -] (husband John)

from which PENNI can infer that John is a husband. Notice that the full quantification mechanism of first-order logic does not exist in this system, which simplifies its search problem enormously. Yet we can use NIKL to define a husband as a married man, which is equivalent to saying (all x (man x) & (married x) [-] (husband x)). Thus we get a limited form of quantification – limited enough to be tractable, and not too limited to be useful.

In the future we will add more components to this system, such as a program for reasoning about time. Marc Vilain and Marie Macaissa are building our hybrid system at BBN. The development of NIKL has been joint work with USC/Information Sciences Institute, principally involving Tom Lipkis and William Mark.

2.2 Planning

Natural language programs need to plan their own speech acts and to perceive the plans behind the user's speech acts. The standard situation calculus planners cannot do these jobs, for two reasons. First, they assume that no two actions can overlap in time; one must finish before the other begins. This is not realistic if there are two agents (the system and the user) – the user might begin an action while the system is in the middle of another action. Second, these planners can make only hypothetical statements about the future – statements like "if the robot were to put block A on block B, then block A would be above block C". They cannot make factual statements like "the robot will put block A on block B". But the user has factual beliefs about the system's future actions. If he orders the system to print a report on the line printer tomorrow morning, he believes that the system is going to do this – and the system must understand that he believes it.

We can allow actions to overlap if we say that actions happen during intervals of time. Likewise conditions hold during intervals. In situation calculus a possible future is a series of situations and actions. Having abolished situations, we take possible futures as primitive objects. One of the possible futures is the actual future. We can assert that the system will actually print a document at 9 tomorrow morning by asserting that in the actual future, the printing action will happen during an interval that starts at 9. We can also assert that if the system does not print this document, the user will get fired. This means that in every possible future where the system fails to print this document, the user gets fired. This research is by Andrew Haas, who is also working on a planning program that uses these ideas.

The natural language programs we aim at must help the user without demanding that he state his goals completely and correctly at the beginning of the dialogue. He may leave out something important in his first remarks, or change his mind. This is quite different from the usual situation in planning, where a goal is given completely and correctly in each problem statement. The planner must be able to begin planning with a partial goal statement, and ask the user for more detail when needed. If the user changes his mind, the planner must salvage parts of the old plan that are still helpful in achieving the revised goal. Other workers have looked at planning with incomplete or inaccurate knowledge of the environment. We aim to extend this to handle incomplete and inaccurate knowledge of goals. This work is Marc Vilain's.

2.3 Parallel computation

The obvious reason for parallelism is speed – we would like our programs to carry on dialogue in real time. If we allow ill-formed input, we have more possibilities to look at, and this demands even more computation.

The obvious way to use parallelism is to have the major components of the system – parser, semantic interpreter, etc. – run in parallel. We have devised a control structure called a cascade that allows programs to produce output continuously, rather than in a single batch when they finish a problem. So the semantic interpreter can begin work as soon as the parser produces a significant hypothesis about the input. Thus we can start interpreting the user's utterance as soon as he starts to type, instead of waiting until he's finished.

This kind of parallelism is useful, but it can only speed up the system a little because there are only a few major components. In order to use parallelism more fully, we are designing a parallel programming language. It is a general-purpose programming language, leaning towards the special needs of AI. Unlike some parallel languages, the user does not turn on the parallelism by calling a special parallel construct. In this language parallel computation is normal – you have to go out of your way to turn it off. Even without a parallel implementation, this language will be useful for learning to think in parallel. If you assume that your program will run in parallel, you are forced to consider which parts of the solution depend on which other parts. This often leads to a better algorithm even if you end up using a serial machine. This work is by N. S. Sridharan. BBN is building the Butterfly – a parallel machine with 128 processors – so we have a chance to test our ideas with an implementation.

References

Bobrow, R.J. and Webber, B.L. 1981 Architectures for Semantic and Syntactic Interaction. In: Woods, W.A. et al., Eds., Research In Knowledge Representation for Natural Language Understanding. Annual Report (September 1980 to August 1981). Report No. 4785. BBN Laboratories: 65-113.

Goodman, Bradley 1984 Repairing Reference Identification Failures by Relaxation. In: Sidner et al.: 135-184.

Haas, Andrew 1984 Planning in a Changing World. In: Sidner et al.: 45-75.

Polyani, Livia and Scha, Remko 1984 A Syntactic Approach to Discourse Semantics. In *Proceedings of the 1984 International Conference on Computational Linguistics*. Stanford, California.

Sidner, Candace 1984 Speakers' Plans and Discourse. In Sidner et al.: 101-133.

Sidner, C.L. et al., Eds. 1984 Research In Knowledge Representation for Natural Language Understanding. Annual Report (September 1983 to August 1984). Report No. 5694. BBN Laboratories.

Sridharan, N.S. forthcoming A Semi-Applicative Language for Artificial Intelligence.

Vilain, Marc 1984 KL-TWO, A Hybrid Knowledge Representation System. In: Sidner et al.: 1-29.

Weischedel, Ralph M. and Sondheimer, Norman K. 1983 Meta-Rules as a Basis for Processing Ill-Formed Input. *American Journal of Computational Linguistics* 9(3-4): 161-177.