"No Better, but no Worse, than People"

David D. McDonald
University of Massachusetts at Amherst

1. Generation versus Understanding

Natural language understanding and natural language generation could employ the same knowledge of language.[1] They could even represent their knowledge in the same way, provided that it was a nonprocedural encoding.[2] However, the two processes that draw on the knowledge cannot be the same because of the radical differences in information flow: Decision-making is a radically different kind of process than hypothesis maintainance. Understanding proceeds from a sequentially scanned text to content and intentions; generation does just the opposite. Understanding processes must cope with ambiguity and underspecification, problems that do not arise in generation (i.e. an audience receives more information from situationally controlled inferences than from the literal text). Generators on the other hand must on some basis choose from an oversupply of syntactic and lexical mechanisms all the while remaining consistent with the constraints imposed by grammaticality, linear order, and stylistic convention--a classic planning problem that now invites careful solutions closely tuned to the special demands of natural language.

Neither process is particularly more heuristic in its judgements than the other. If generation appears more algorithmic, it is because of the weakness of the present models of intentionality, situation, lexical sources, and especially audience reactions. People have no assurance that their

[1] Presently they don't--generation uses more. Generation demands knowledge of the conventions and heuristics of language use, but understanding systems today do not attempt to recover any such assessments of why speakers say what they do in the particular manner that they do. They don't have to--the programs they are presently working for wouldn't appreciate the information if they did.

[2] My own candidate for the neutral, shared encoding would be a catalogue of all the minimal elementary surface structure trees of the language, plus the rules that govern how they can be combined, e.g. a Tree Adjoining Grammar. Paired with each tree fragment would be a mapping function associating it with the situations in which its use was appropriate for the individual speaker. Besides my own use of TAGs (McDonald & Pustejovsky, 1985), this framework is a reasonable description of at least the Phran and Phred systems at Berkeley (Jacobs, 1985), and Doug Appelt's (1985) use of functional unification grammar.

choice of what to say will be effective; when programs have richer models they won't be certain either. For all but the most mundain tasks, the complexity of the circumstances will preclude fixed procedures. Instead, our programs will have to do what we appear to: make their choices heuristically, anticipating how the rest of the discourse will go if their assumptions are correct, and being prepared to adjust if it turns out that they are not.

Any conclusion other than that the same knowledge structures underly both understanding and generation would be a drastic philosophic jump from our common view of language as an interpersonal medium and an interface to thought. Any difficulties we presently have in making the same structures "go both ways" reflects weaknesses in our conceptual designs rather than facts about people. In particular, present target representations for understanding are impoverished: inverting them leads to badly underspecified messages since they contain no information about deliberately adopted perspectives or connotated information (such as newness or value judgements), and very little about what the original speaker's goals were.

2. Generation is special.

To do good work in generation one is forced to come to grips with problems that other tasks are today able to ignore. Three examples: One cannot work on generation and ignore syntax. One cannot avoid accounting for the control of variation in linguistic form through appeals to synonomy or cannonical form. One cannot passively accept the semantic representations of one's colleagues' knowledge-based reasoning systems without first determining that they are notationally and epistemologically able to support the distinctions that language makes.

It is not clear to me that these are the sorts of issues that will draw the otherwise reluctant linguist to consider AI, but they without question draw the AI people who work on them to linguistics. Proficiency in the technicalities of syntax and morphology is obligatory in generation research. More importantly, generation people must have a linguist's skill at arguing the consequences of alternative theoretical decisions: Working as we do from empirically

unestablished models of intention and knowedge out to text, we have to justify our designs using indirect evidence and comparative reasoning, something that linguists are well trained for.

This difficulty is in other respects a great advantage (one that linguists in my experience well appreciate) when we are working on today's cutting edge problems in computational linguistics, such as the structure of discourse or the signaling of intended inferences and their relationship to underlying knowledge of the world and social behavior. Our established tools such as example-driven comparative analysis do not fare well on these problems because of the enormous number of factors that contribute to them. Descriptive theories of underlying abstract structure are unsatisfying because the abstractions are slippery to evaluate.

What they need is the synthetic approach provided by generation. The generation process converts abstract structures into concrete texts whose properties we can evaluate empirically. Theories of discourse now can stand or fall on whether they lead to effective conversations, theories of inferencing on whether texts based on them do evoke the intended conclusions.

3. Two non-problems

The possibility of a program somehow generating things that no human could understand is a red herring.[3] People say things all the time that other people don't understand, yet we don't think anything unusual is happening. Usually the audience fails to make an expected inference rather than misunderstand some literal part of the utterance, a problem that can happen quite easily when the speaker misjudges what the audience already knows, or the speaker thinks that they share some judgement or context when they do not. Another source of the problem comes from the speaker thinking that a certain turn of phrase should signal a certain inference but the audience is opaque to that signal.

[3] Since programs wouldn't talk to us if they didn't need to communicate, saying things to us that we don't understand would just be failing to achieve their own goals. Perhaps they might choose to talk this way to each other (though why should they, since given any commonality in their internal designs, telepathy would be much simpler and more satisfying), but if we give them any sensitivity to their audience's reactions (and how could communication be effective without it) they will quickly realize that we're missing the point of most of what they're saying to us and change their techniques.

The very same mistake could be made by a program--we cannot program them to be superhumanly aware of their audience. The only protection is incorporating into language interfaces the same kind of sensitivity to later audience reactions that we have ourselves. We know what the effect of following our inferences should be on our audiences, and we can sense when they have missed our intent. We especially know how to feed back a communications failure onto our own generation strategies so that we will make different choices the next time we need to get across a similar idea. We should make our machines able to do the same.

The problem of how best to match a system's input and output language abilities is likely to turn out to be a red herring as well, one that will go away naturally as soon as our understanding systems become as syntactically and lexically competent as our generators.[4] The problem is that presently if the generator produces a more sophisticated construction than the understander can parse or uses a word that it does not know, then the human user, mimicing what the generator has done, will be frustrated when he turns out not to be understood.

If this were the only difficulty, then it could be solved by straightforward software engineering: consistency tools would force one to drop items from the generator's repertoire that the understander did not know. Unfortunately the problem goes deeper than that. The mismatch is not the issue, since people's abilities do not match either: we all can understand markedly more than we would ever say. The real problem for a non-research interface is--direct queries for literal information aside--that machine understanding abilities are so far below the human level that any facile, inference motivating output from the generator is going to suggest to the user that the system will understand things that it cannot.

Because of this, I personally would never include language input in a non-research interface today. Interactive graphics and menu facilities do not suffer from the ambiguity and scope of inferencing problems faced by language, and give a realistic picture of what a system is actually able to comprehend. Interfaces based on a "graphics in, graphics and speech out"

[4] It is trivial to specify a linguistically complex phrase and have a generator utter it by rote. Such canned or template-based text is often the best route to take in a practical interface. If the programmer is sure that the situation warrants the phrase then it can safely be used, even though there may be no explicit model within the system from which the phrase could have been deliberately composed.

paradigm have not been given enough study by the language and communications research community, and are likely to be a much better match to the deliberative and intentional abilities of the programs we can experiment with today.

4. Controling Decision-making

There are volumes to be said on how one could or should control for syntactic and lexical choice--this is the primary question that any computational theory of generation answers. Rather than attempt to summarize my position in the little space that remains, let me point out two issues that I believe distinguish much of the work presently going on; for a larger discussion of these see McDonald, Vaughan, and Pustejovsky (1987).

The first issue is whether one attempts to make psychological claims with the form and operation of the generator. This is the more demanding road to take. It may also turn out to be the only one that provides for continuous extension and elaboration. Language, like vision, may be so tied up with the nature of the human mind and its computational properties that no design that goes against those properties will ever be more than a special purpose hack. Making claims with a computational process requires one to take exceptional discipline in designing the operations and representations it will use. Much of the explanatory load will be taken on by the restrictions on the mechanism's behavior, and these can be easily diluted by the kinds of programming conveniences that make a generator easier to engineer. Adopting a psychological point of view can thus retard efforts to make a generator more competent.

The second issue is how much generation knowledge is to be found in the non-linguistic, "underlying program" in whose service the generator is operating. The more that we take to be there, the greater the burder we place on our knowledge-based system colleagues to make sure that it is included; however our theories may have very good reasons for requiring it. This knowledge might be the direct encoding of rhetorically relevant structural relations: How deeply do we believe that the notions of "compare and contrast" are to be found in the mind or should be found in a program? It might be of lexical identities: Are the conceptual primitives of the underlying program fine-grained and closely matched to real words, or large-grained and

abstract? It might also be in the modularity of the underlying system's information: Is it propositional and easily mapped onto kernel clauses and noun phrases, or does it have some drastically different organization?

Generation research today has the lion's share of the important computational linguistics problems. As more and more people work in it, it will quickly become the cutting edge, forcing extensions on understanding and knowledge representation if they are to match it as a source of insight into the nature of language and thought in the human mind. There is no appropriate goal for generation research short of matching human performance, part of which entails coming to understand the limits on that performance. We don't really know how good people are at using language; our experiments with mechanical speakers may someday tell us.

5. References

Appelt, Doug (1985) Planning English Sentences, Cambridge University Press.

Jacobs, Paul (1985) "PHRED: A generator for natural language interfaces", Berkeley Computer Science Department TR 85/198.

McDonald, David & Pustejovsky, James (1985) "TAGs as a Grammatical Formalism for Generation", proc. ACL-85, Chicago, 1985, 94-103.

_____, _____, & Vaughan, Marie (1987) "Factors Contributing to Efficiency in Natural Language Generation", in Kempen (ed) **Papers from the Third International Workshop on Language Generation**, Martinus Nijhoff Press (Kluwer), The Netherlands.