Requirements for Robust Natural Language Interfaces: The LanguageCraft™ and XCALIBUR experiences

Jaime G. Carbonell
Carnegie-Mellon University
and Carnegie-Group Inc.
Pittsburgh, PA 15213, USA

Introduction

Natural Language interfaces to data bases and expert systems require the integration of several crucial capabilities in order to be judged *habitable* by their end users and *productive* by the developers of applications. User habitability is measured in terms of linguistic coverage, robustness of behavior and speed of response, whereas implementor productivity is measured by the amount of effort required to connect the interface to a new application, to develop its syntactic and semantic grammar, and to test and test and debug the resultant system assuring a certain level of performance. These latter criteria have not been addressed directly by natural language researchers in pure laboratory settings, with the exception of user-defined extensions to an existing interface (e.g., NanoKLAUS [4], VOX [6]). But, in order to amortize the cost of developing practical, robust and efficient interfaces over multiple applications, the implementor productivity requirements are as important as user habitability. We treat each set of criteria in turn, drawing from our experience in XCALIBUR [2] and in LanguageCraft™ [5], a commercially available environment and run time module for rapid development of domain-oriented natural language interfaces.[1] In our discussion we distill the general lessons accrued from several years of experience using these systems, and conducting several small-scale user studies.

User Habitability

Natural language used for communication in task-oriented dialogs differs from that of published text. Perhaps it should not be surprising that dialog phenomena, especially ellipsis and anaphora [3] dominate over complex syntactic constructions, but the implications of this observation for habitability of natural language interfaces have not yet become widely known or accepted. Conversely, the criteria for user habitability itemized below apply only to interfaces, not to the comprehension of longer written texts.

- ***Coverage*** – All significant domain concepts (objects, relations, states and actions) must be incorporated in the grammar and knowledge base. Conceptual coverage is more crucial than extended syntactic coverage, as users will adapt to syntactic limitations but will not tolerate the total inability to express concepts or operations they judge significant. Moreover, users of interactive natural language systems very seldom type long complicated sentences of the type that abound in literary works. Even such common grammatical structures as subordinate clauses and clause-level coordination occur with relatively low frequency in task-oriented interfaces,

- ***Ellipsis Resolution*** – Brevity is the key to successful communication. Natural language has been compared unfavorably with artificial command languages on the grounds that it is often more verbose, and typing is an activity one wishes to minimize. However, we have found that in many communications with expert systems and in some database query tasks, fully half of all interactions are expressed as extremely brief elliptical utterances. Exploiting contextual information, one can sometimes communicate with fewer keystrokes in natural language than in an artificial language lacking in elliptical or anaphoric reference. Support for ellipsis is therefore a central design component of LanguageCraft and XCALIBUR.

- ***Anaphora Resolution*** – Anaphora is almost as ubiquitous as ellipsis. Using pronouns like "it" or dydatic references like "that calculation" to refer to objects or actions of arbitrary complexity makes communication more natural and much briefer. A surprising result from one of our early user studies showed that whereas it is possible to restrict users from employing complex grammatical structures, it is not possible for them to avoid use of anaphors. Users will understand and follow the instruction to avoid anaphors for a couple of sentences and revert back to using them as soon as they concentrate on the task at hand.

- ***Metalanguage*** – Utterances about other utterances occur with some regularity, e.g., "I meant to type gauss.for instead", "Oops, I didn't mean that!", or "When I say 'print' I mean on the terminal". However these are more difficult to handle systematically and therefore go beyond the scope of the current practical implementations.

- **Robustness** – Users invariably commit errors of orthography, switch word order, violate agreement, omit function words, insert spurious words, or use incorrect punctuation. Moreover, they often do not notice their errors, as task knowledge and redundancies in the language allow for fairly easy human comprehension of sentences that fail to respect all grammatical niceties. Approximately a third of all sentences in our analyzed sample of several hundred interactions were extragrammatical in a strict sense, mostly due to sloppy user input. However, initial work at automated recovery when possible, and focused interactive correction when needed, shows promise for future improvements in this important aspect of user habitability [1].

- **Response time** – Next to grammatical coverage, perhaps the most widely recognized requirement for habitability is real-time response. We find that whereas this is indeed an issue, the combination of new parsing techniques [8], faster hardware, and on-line parsing[2] mean that real time performance will be less of a concern for developers of task oriented natural language interfaces.

- **Back-end response** – Last but certainly not least, the manner by which the backend system responds to the user is crucial. An ideal natural language recognizer coupled to an expert system or data base that returns its answers in a form totally incomprehensible to the user is of little use. Thus in both XCALIBUR and LanguageCraft we have developed natural language generators (as well as graphics and tabular output generators) to close the communication loop with the user.

[1]LanguageCraft, Plume and Grammar Writer's Workbench are trademarks of Carnegie Group Inc.

[2]On-line parsing means that a system parses the input as it is being typed from left to right, and thus exploits user typing time that would otherwise be idled away.

System Builder Productivity

The more elaborate a natural language interface, the harder it is to port to new application domains. In this manner there is some tension between habitability and complexity of development. But, in order to ease the difficulty without materially sacrificing the habitability requirements set forth above, several principles and development tools have emerged,-to wit:

- ***Decomposition*** – Traditionally, the syntactic recognition and semantic interpretation components of a natural language system were compartamentalized into separate subsystems because the former is domain general whereas the latter is domain specific. However, such separation entails serious performance compromises, both in speed and accuracy of the resultant analysis (e.g. the inability to resolve syntactic ambiguities without semantic criteria, and the inability to recover from ill-formed input unless both semantic and syntactic constraints are unified in the recognition process). Lately, a new approach is emerging, where separate syntactic and semantic knowledge sources are precompiled into a unified grammar [7], thus sharing the advantages of separation of knowledge sources at development time and integrated robust parsing at run time.

- ***Grammar development workbenches*** – In order to speed the development of a new interface, and to ensure consistency and well-formedness of new grammars and lexicons, specialized software tools are begin developed, much like the structured editors and programming environments that improve programmer productivity. Moreover, grammars are more highly structured than computer programs, thus such tools have an even greater impact in improving grammar-writer productivity.

- ***Run-time tracing and displays*** – Once again borrowing from software engineering, utilities to trace the application of a grammar to a set of examples (and to display the processing and output in meaningful ways exploiting graphic capabilities of the new workstations) are enhancing the debugging and quality assurance aspects of new grammar development.

- ***Systematic backend translator*** – As the productivity of the grammar developer increases, the effort to connect the parser to the backend application becomes a larger fraction of the total development cost. The major part of this problem entails the translation of semantic structures output by the parser (such as case frames) into the input language required by the application (such as data base query languages). In order to enhance developer productivity and minimize errors of *ad-hoc* programming, we have developed a systematic transformation language (KAFKA) in XCALIBUR, and a rule-based translator in LanguageCraft.

The systematic development of natural language interfaces requires a run time system capable of providing the habitability requirements listed in the previous section, and a development environment capable of providing the grammar-writer and applications engineering support listed here. LanguageCraft is the first commercial system to provide most of these capabilities. Plume, its case-frame run-time parser, contains a substantial part of common English syntax (and recently a JPLUME contains Japanese syntax), and manages dialog issues such as ellipsis resolution and interactive disambiguation. The Grammar Writers Workbench provides the LanguageCraft development environment, consisting of a structured grammar editor, a consistency checker, debugging and tracing facilities, and support for a rule-based language to connect to different applications. We expect these facilities to improve in LanguageCraft, especially as it becomes multilingual and our experience base with different applications increases. And, we also expect other systems to emerge that incorporate different methods for meeting the requirements set forth in the present document.

References

1. Carbonell, J. G. and Hayes, P. J., "Recovery Strategies for Parsing Extragrammatical Language," *American Journal of Computational Linguistics,* Vol. 9, No. 3-4, 1983, pp. 123-146.

2. Carbonell, J. G., Boggs, W. M., Mauldin, M. L. and Anick, P. G., "The XCALIBUR Project, A Natural Language Interface to Expert Systems and Data Bases," in *Applications in Artificial Intelligence,* S. Andriole, ed., Petrocelli Books Inc., 1985.

3. Carbonell, J. G., "Discourse Pragmatics in Task-Oriented Natural Language Interfaces," *Proceedings of the 21st annual meeting of the Association for Computational Linguistics,* 1983.

4. Haas, N. and Hendrix, G. G., "Learning by Being Told: Acquiring Knowledge for Information Management," in *Machine Learning, An Artificial Intelligence Approach,* R. S. Michalski, J. G. Carbonell and T. M. Mitchell, eds., Tioga Press, Palo Alto, CA, 1983.

5. Carnegie Group Inc., *The LanguageCraft Reference Manual,* Pittsburgh, PA, 1985.

6. Meyers, A., "VOX - An Extensible Natural Language Processor," *Proceedings of IJCAI-85,* Los Angeles, CA, 1985, pp. 821-825.

7. Tomita, M. and Carbonell, J. G., "Another Stride Towards Knowledge-Based Machine Translation: An Entity Oriented Approach," *Proceedings of COLING-86,* 1986.

8. Tomita, M., *Efficient Parsing for Natural Langauge,* Kluwer Acadmic Pulishers, 1986.