Workshop on the Evaluation of Natural Language Processing Systems

Martha Palmer
Unisys Center for Advanced Information Technology
Paoli, PA 19301

Tim Finin
Unisys Center for Advanced Information Technology
Paoli, PA 19301

1 Introduction

In the past few years, the computational linguistics research community has begun to wrestle with the problem of how to evaluate its progress in developing natural language processing systems. With the exception of natural language interfaces, there are few working systems in existence, and they tend to focus on very different tasks using equally different techniques. There has been little agreement in the field about training sets and test sets, or about clearly defined subsets of problems that constitute standards for different levels of performance. Even those groups that have attempted a measure of self-evaluation have often been reduced to discussing a system's performance in isolation—comparing its current performance to its previous performance rather than to another system. As this technology begins to move slowly into the marketplace, the lack of useful evaluation techniques is becoming more and more painfully obvious.

In order to make progress in the difficult area of natural language evaluation, a Workshop on the Evaluation of Natural Language Processing Systems was held on December 7–9, 1988 at the Wayne Hotel in Wayne, Pennsylvania. The workshop was organized by Martha Palmer (Unisys), assisted by a program committee consisting of Beth Sundheim (NOSC), Ed Hovy (ISI), Tim Finin (Unisys), Lynn Bates (BBN), and Mitch Marcus (Pennsylvania). Approximately 50 people participated, drawn from universities, industry, and government. The workshop received the generous support of the Rome Air Defense Center, the Association of Computational Linguistics, the American Association of Artificial Intelligence, and Unisys Defense Systems.

The workshop was organized along two basic premises. First, it should be possible to discuss system evaluation in general without having to state whether the purpose of the system is "question-answering" or "text processing." Evaluating a system requires the definition of an application task in terms of input/output pairs that are equally applicable to question-answering, text processing, or generation. Second, there are two basic types of evaluation, *black-box evaluation,* which measures system performance on a given task in terms of well-defined input/output pairs, and *glass-box evaluation,* which examines the internal workings of the system. For example, glass-box performance evaluation for a system that is supposed to perform semantic and pragmatic analysis should include the examination of predicate-argument relations, referents, and temporal and causal relations. Since there are many different stages of development that a natural language system passes through before it is in a state where black-box evaluation is even possible (see Figure 1), glass-box evaluation plays an especially important role in guiding the development at early stages.

With these premises in mind, the workshop was structured around the following three sessions: (i) defining the notions of "black-box evaluation" and "glass-box evaluation" and exploring their utility; (ii) defining criteria for "black-box evaluation"; and (iii) defining criteria for "glass-box evaluation." It was hoped that the workshop would shed light on the following questions.

- What are valid measures of "black-box" performance?
- What linguistic theories are relevant to developing test suites?
- How can we characterize efficiency?
- What is a reasonable expectation for robustness?
- What would constitute valid training sets and test sets?
- How does all of this relate to measuring progress in the field?

2 Background

Before looking at the distinctions between "black-box" and "glass-box" evaluation, it is first necessary to examine the development of a natural language system a little more closely. There are several different phases, and different types of evaluation are required at each phase. The various phases are summarized in Figure 1.

NLP System Development Steps

1. Picking the application.
2. Characterising the necessary phenomena.
3. Selecting relevant theories, if available.
4. Developing and testing algorithms that implement these theories.
5. Implementing the first pass at the system, testing it, and identifying gaps in coverage.
6. Characterising the new phenomena that have been discovered, especially those having to do with interactions between components.
7. Fine-tuning algorithms to improve efficiency, and also replacing algorithms as the characterization of the phenomena changes.
8. Second pass at implementation, to extend coverage to these new phenomena and thus fill in the gaps.
9. Third pass at an implementation in which a focus is placed on issues of extensibility.
10. Fourth and final pass at the implementation in which the system moves into a production environment. This stage pays special attention to issues of robustness.

Figure 1 There are a number of different stages in the development of a natural language processing system. Different kinds of evaluations are required and/or possible at the different stages.

Speaking very roughly, the development of a natural language processing system is usually sparked by the needs of the particular application driving it, whether it be question answering, text processing, or machine translation. What has happened in the past is that, in examining the requirements of such an application, it has quickly become apparent that certain phenomena, such as pronoun reference, are essential to the successful handling of that application. It has also quickly become apparent that for many of these phenomena, especially semantic and pragmatic ones, past linguistic analysis has very little to offer in the way of categorization. Even when it might offer a fairly rigorous account of the phenomenon, as in the case of syntax, it has very little to say about useful algorithms for efficiently producing syntactic analyses and even less to say about interaction between different types of phenomena. So, almost before beginning implementation, a great deal of effort in the computational linguistics community must of necessity be devoted to tasks that can rightly be seen as belonging to theoretical linguistics. The *Discourse Canon* that Bonnie Webber prepared for the Mohonk Darpa Workshop (1988) is an excellent example of the type of groundwork that must be done prior to serious attempts at implementation, and must be continued throughout and subsequent to said implementation. The field needs many more such "canons" for other semantic and pragmatic phenomena.

Algorithm development is equally important, and can also be carried out independently of or in parallel with an implementation. We have several different algorithms for syntactic parsing, and ways of comparing them (and ways of proving that they are all equivalent), but very few algorithms for semantics and pragmatics.

Implementing an algorithm for use in an application is a separate stage of development. Progress cannot, however, be measured in terms of accurate output until a system that uses particular algorithms to handle particular phenomena has been implemented. In addition to methods for measuring performance of entire systems, we also need ways of measuring progress in characterizing phenomena and developing algorithms that will contribute to system development.

Once a system is up and running, the accuracy of its output can then be measured. The different types of the output can be associated with the phenomena that have to be handled in order to produce each type. For example, consider a phrase from the *trouble failure report* domain (Ball 1989):

Replaced interlock switch with new one.

In order accurately to fill in the slot in the database field associated with the new-part-installed(-) relation, the "one" anaphora has to be correctly resolved, requiring a complex interaction between semantic and pragmatic analysis.

It is possible to have two systems that produce the same output, but do it very differently. This is where such issues as efficiency, extensibility, maintainability, and robustness come in. A more efficient implementation, for example, may be able to support a larger, more complex domain. With a more general implementation, it should be easier to extend the scope of the system's domain or to port the system to an entirely new domain. A system with a more convenient or more robust interface will be easier to use and, one would suppose, used more often. In a recent study comparing Lotus HAL (Lotus with a restricted natural language interface) to Lotus, not only was Lotus HAL judged to be more convenient to use, but the Lotus HAL users also had higher scores on the problem-solving tasks (Napier 1989).

2.1 BLACK-BOX EVALUATION

Black-box evaluation is primarily focused on "what a system does." Ideally, it should be possible to measure performance based on well-defined I/O pairs. If accurate output is produced with respect to particular input, then the system is performing correctly. In practice, this is more difficult than it appears. There is no consensus on how to evaluate the correctness of semantic representations, so output has to be in terms of some specific application task such as database answering or template fill (Sundheim 1989). This allows for an astonishing amount of variation between systems, and makes it difficult to separate out issues of coverage of linguistic phenomena from robustness and error recovery (see Figure 2).

In addition to the accuracy of the output, systems could also be evaluated in terms of their user-friendliness, modularity, portability, and maintainability. How easy are they to use, how well do they plug into other components, can they be ported and maintained by someone who is not a system expert? In general, it should be possible to perform

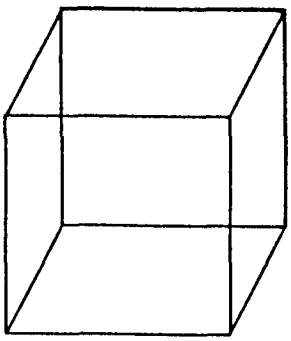

Figure 2 A "black-box evaluation" is primarily focused on "what a system does." It attempts to measure the system performance on a given task in terms of well-defined input/output pairs.

black box evaluation without knowing anything about the inner workings of the system—the system can be seen as a black box, and can be evaluated by system users.

2.2 GLASS-BOX EVALUATION

In contrast, glass-box evaluation attempts to look inside the system, and find ways of measuring how well the system does something, rather than simply whether or not it does it. Glass-box evaluation should measure the system's coverage of a particular linguistic phenomenon or set of phenomena and the data structures used to represent them (see Figure 3). And it also should be concerned with the efficiency of the algorithms being used. Many of these tests could be performed only by a system builder. They are especially useful in attempting to measure progress when a system is under development. Glass-box evaluation should also include an examination of relevant linguistic theories and how faithfully they are implemented. If a linguistic theory does not deal with all of the data and has to be modified by the developer, those modifications need be clearly documented, and the information relayed to the theory's developer. For example, as pointed out in Bonnie Webber's (Penn) presentation, there is a distinction between *Tree Adjunction Grammar* (TAG) as a linguistic theory and the several algorithms that have been used to implement TAG parsers: Extended CKY parser, Extended Earley parser, Two-pass extended Earley parser based on lexicalized TAGs, and a DCG parser using lexicalized TAGs. There is also a distinction between *Centering* as a theory for resolving anaphoric pronouns (Joshi and Weinstein 1981; Gross et al. 1983), and the attempts to use a centering approach to resolving pronouns in an implementation (Brennan et al. 1987).

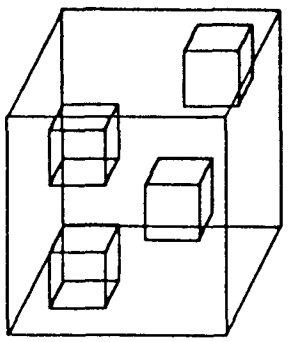

Figure 3 A "glass-box evaluation" addresses "how the system works". It attempts to look inside the system and find ways of measuring how well the system does something, rather than simply whether or not it does it.

In addition, one way of looking inside a system is to look at the performance of one or more modules or components. Which components are obtained depends on the nature of the decomposition of the system. NL systems are commonly decomposed into *functional modules* (e.g., parser, semantic interpretation, lexical lookup, etc.), each of which performs a specified task, and into analysis phases during which different functions can be performed. A black-box evaluation of a particular component's performance could be seen as a form of glass-box evaluation. For example, taking a component such as a parser and defining a test that depends on associating specified outputs for specified inputs would be a black-box evaluation of the parser. Since it is an evaluation of a component that cannot by itself perform an application, and since it will give information about the component's coverage that is independent of the coverage of any system in which it might be embedded, this can be seen as providing glass-box information for such an overall system.

3 WORKSHOP FORMAT

The workshop began with a set of presentations that discussed evaluation methods from related fields: speech processing (Dave Pallet—NIST), machine translation (Jonathan Slocum—Symantec), and information retrieval (Dave Lewis—UMASS). This was followed by a panel on reports on evaluations of natural language systems chaired by Lynn Bates (BBN), and including John Nerbonne (HP), Debbie Dahl (Unisys), Anatole Gershman (Cognitive Systems, Inc), and Dick Kitteridge (Odyssey Research, Inc). After lunch Beth Sundheim (NOSC) presented the workshop with the task for the afternoon working groups, which was to discuss black-box evaluation methodologies. The groups consisted of *Message Understanding* chaired by Ralph Grishman (NYU), *Text Understanding* chaired by Lynette Hirschman (Unisys), *Database Question-Answering* chaired by Harry Tennant (TI), *Dialogue Understanding* chaired by Mitch Marcus (Penn), and *Generation* chaired by Ed Hovy. After reporting on the results of the working groups, the workshop met for a banquet, which included a demonstration of the Dragon speech recognition system by Jim Baker. The second day began with another presentation of a method of black-box evaluation applied to syntactic parsers, (i.e., glass-box evaluation

with respect to an entire system), by Fred Jelinek (IBM), and then moved on to an introduction of the topic of glass box evaluation by Martha Palmer (Unisys). A panel chaired by Jerry Hobbs (SRI), which included Mitch Marcus (Penn) and Ralph Weischedel (BBN), discussed necessary characteristics for corpora to serve as training sets and test sets for black-box evaluation of systems and components of systems. The workshop then broke up into a new set of working groups to discuss a glass-box evaluation task introduced by Bonnie Webber (Penn): syntax, chaired by Dick Kitteredge (Odyssey Research), semantics, chaired by Christine Montgomery (Language Systems, Inc.), pragmatics and discourse, chaired by Candy Sidner (DEC), knowledge representation frameworks, chaired by Tim Finin, and systems, chaired by Lynn Bates. The final session was devoted to reports of the working groups and summarization of results.

4 Black-Box Evaluation

Beth Sundheim (NOSC) proposed a black box evaluation of message understanding systems consisting of a training set of 100 messages from a specific domain, and two separate test sets, one consisting of 20 messages and another of 10. The performance was to be evaluated with respect to a frame-filling task. There was general agreement among the workshop participants that useful black-box evaluations can be done for the message understanding and database question-answering task domains. It was also agreed that more general systems aimed at text understanding and dialogue understanding were not good candidates for black-box evaluation due to the nascent stage of their development, although individual components from such systems might benefit from evaluation. The workshop attendees were pleasantly surprised by the results of the generation group, which came up with a fairly concrete plan for comparing performance of generation systems, based on the message understanding proposal. A perennial problem with all of these proposals, with the exception of the message understanding proposal, is the lack of funding. Conferences and workshops need to be organized, systems need to be ported to the same domain so that they can be compared, etc., and there is very little financial support to make these things possible.

4.1 MESSAGE UNDERSTANDING CONFERENCE II

Beth Sundheim's black-box evaluation was in fact carried out last summer, June 1989, at MUCK II (Message Understanding Conference II) with quite interesting results (Sundheim 1989).

It quickly became clear how important it was for systems to be able to handle partial input, a characteristic normally associated with usability. A system that could only handle 60 percent of the linguistic phenomena, but could do that in a robust fashion could receive a higher accuracy rating than a system that was capable of handling 80 percent of the linguistic phenomena, but only under ideal circumstances. The overall system performance, including many features that are not directly related to natural language processing, was a more important factor in the scoring than the system's linguistic coverage. Since these tests are intended to compare mature systems that are ready for end users, this is entirely appropriate, and is exactly what the end users are interested in. They are not concerned with how the system arrives at an answer, but simply with the answer itself. However, tests that could provide more information about how a system achieved its results could be of more real utility to the developers of natural language systems.

5 Glass-Box Evaluation

One of the primary goals of glass-box evaluations should be providing guidance to system developers—pinpointing gaps in coverage and imperfections in algorithms. The glass-box evaluation task for the workshop, as outlined by Bonnie Webber (Penn), consisted of several stages. The first stage was to define for each area a range of items that should be evaluated. The next stage was to determine which items in the range were suitable for evaluation and which were not. For those that could be evaluated, appropriate methodologies (features and behaviors) and metrics (measures made on those features and behaviors) were to be specified. For items or areas that were not yet ready, there should be an attempt to specify the necessary steps for improving their suitability for evaluation.

As explained in more detail below, the glass-box methodology most commonly suggested by the working groups was black-box evaluation of a single component. The area that seemed the ripest for evaluation was syntax, with semantics being the farthest away from the level of consensus required for general evaluation standards. Pragmatics and discourse heroically managed to specify a range of items and suggest a possible black-box evaluation methodology for a subset of those items. Knowledge representation specified subtopics with associated evaluation techniques.

5.1 SYNTAX

The most clearly defined methodology belonged to the syntax group, and has since taken shape in the form of the Treebank project, which follows many of the guidelines originally suggested by Fred Jelinek. This project will be able to evaluate syntactic parsers by comparing their output with respect to previously determined *correct* parse information—a black-box evaluation of a single component, i.e., a parser. The project has recently been established at the University of Pennsylvania under Mitch Marcus and is funded by DARPA, General Electric, and the Air Force. The goal of the project is to collect a large amount of data, both written language and spoken language, which will be divided into training sets and test sets. It involves annotating the data with a polytheoretic syntactic structure. It has been agreed that the annotation includes lexical class labels, bracketing, predicate argument

relationships, and possibly reconstruction of control relationships, wh-gaps, and conjunction scope. Eventually it would be desirable to include co-reference anaphora, prepositional phrase attachment, and comparatives, although it is not clear how to ensure consistent annotation. People interested in testing the parsing capability of their systems against untried test data could deliver the parsers to the test site with the ability to map their output into the form of the corpus annotation for automatic testing. The test results can be returned to parser developers with overall scores as well as scores broken out by case, i.e., percentage of prepositional phrase bracketings that are correct.

5.2 SEMANTICS

One of the special difficulties in attempting to develop glass-box evaluation techniques is the lack of agreement over the content of semantic representations. Most people will agree that predicate argument relations, temporal relations, and modifiers (including prepositional phrase attachment) count as semantic phenomena, but will not agree on instructions for annotation or methodologies for evaluation. This is partly because semantics draws on so many diverse areas. People who are primarily interested in underlying cognitive structures have been accused of ignoring relations to surface syntactic phenomena. Logicians who are concentrating on building formal tools have been accused of ignoring lexical and cognitive issues, and people concerned with lexical semantics have been accused of ignoring everything but the dictionary. Some day this will all be brought together in peace and harmony, but meanwhile there are as many different styles of semantic representation as there are researchers in the field. The only possible form of comparative evaluation must be task-related. Good performance on such a task might be due to all sorts of factors besides the quality of the semantic representations, so it is not really an adequate discriminator.

In the Darpa Spoken Language Workshop in February 1989, Martha Palmer suggested three likely steps toward achieving more of a consensus on semantic representations:

1. Agreement on characterization of phenomena.
2. Agreement on mappings from one style of semantic representation to another.
3. Agreement on content of representations for a common domain.

An obvious choice for a common domain would be one of the MUCK domains, such as the OPREPS domain recently used for MUCK II. There are several state-of-the-art systems that are performing the same task for the same domain using quite different semantic representations. It would be useful to take four of these systems, say NYU, SRI, Unisys, and GE, and compare a selected subset of their semantic representations in depth. It should be possible to define a mapping from one style of semantic representation to another and pinpoint the various strengths and weaknesses of the different approaches. Another potential choice of domain is the Airline Guide domain. The Airline Guide task is a spoken language interface to the Official Airline Guide, where users can ask the system about flights, air fares, and other types of information about air travel.

5.3 PRAGMATICS AND DISCOURSE

The group's basic premise was that they would need a large corpus annotated with discourse phenomena. This would allow them to evaluate the effect of individual components upon the system as a whole and upon other components, such as syntax and semantics. It would also allow an individual component's behavior to be observed. They listed the discourse phenomena shown below, with the ones for which precise annotation instructions could be given marked with a *. The others might take a bit more thought. It was agreed that the topics for a subsequent meeting would include experimenting with text annotations and designing training sets and test sets.

- turn taking
* referring expressions, including anaphora, "do so," respectively
- multi-sentence text
- sensitivity to user's goals and plans
- model of user's beliefs, goals, intentions, etc.
- use of Gricean maxims
- use of speech acts
- interpretation and use of temporal and causal relationships
* part/whole, member/set relationships
- vague predicate specification
* determination of implicit arguments in predicate-argument relationships
- metaphor and analogy
- schema matching
- varying depth of processing based on certain criteria
- focus of attention and saliency of entities
* ellipsis
- style and social attitudes
- deixis

5.4 KNOWLEDGE REPRESENTATION FRAMEWORKS

This group began by pointing out that the *knowledge representation and reasoning* (KR&R) services provided for natural language systems fall into two classes: (1) providing a *meaning representation language* (MRL); and (2) providing inferential services in support of syntactic, semantic, and pragmatic processing. The group noted that the MRL class should probably be broadened to include languages for representing dialogues, lexical items, etc. In addition, the group laid out a spectrum of activities, which are included in a KR&R shown in Figure 4.

The group suggested three evaluation methodologies. The first was aimed at evaluating a KR&R system's suitability as a meaning representation language. One way to evaluate a potential MRL is to have a standard set of

- **theory** - Is there an underlying theory which gives meaning to the KR&R system? What is known about the expressiveness of the language and the computational complexity of its reasoning?
- **languages** - How does the KR&R system function as a practical language for expressing knowledge? How easy or difficult is it to define certain concepts or relations or to specify computations?
- **systems** - KR&R systems are more than just an implementation of an underlying theory. They require good development environments: knowledge acquisition tools, debugging tools, interface technology, integration aids, etc. How extensive and good is this environment?
- **basic models** - A KR&R system often comes with some basic, domain-independent modules or models, such as temporal reasoning, spatial reasoning, naive physics, etc. Are such models available and, if they are, how extensive and detailed are they?

Figure 4 There are several dimensions along which a knowledge representation and reasoning system might be evaluated.

natural language expressions to try to express in the MRL. This provides an evaluator with some idea of the expressiveness and conciseness of the KR&R system as an MRL. A second evaluation methodology follows the "Consumer's Reports" paradigm and involves developing a checklist of features. An extensive list of KR&R features could be developed for each of the dimensions given in Figure 4. Scoring how well KR&R systems provide each of these features provides a way to compare different systems. The final evaluation technique is to hold a MUCK-like workshop aimed at evaluating the performance of the NLP system's underlying KR&R system. The group outlined a proposal for organizing a workshop to do an evaluation of the KR&R aspects of a natural language processing system based on the MUCK Workshop models.

6 Workshop Conclusions

Several concerete results came out of the workshop. In particular, a consensus was reached on the black-box evaluation task for the second Message Understanding Conference (MUCK II), and a consensus was also reached on the desirability of a common corpus of annotated language, both written and spoken, that could be used for training and testing purposes. Since the workshop, MUCK II has been held with interesting and useful results, and the Treebank project at the University of Pennsylvania has received funding and has begun. This should eventually lead to a more formalized testing and comparisons of parsers. Evaluation is becoming a more prevalent topic at NL workshops, such as the one held at RADC in September 1989, and the Darpa Spoken Language Community is working hard to construct a general evaluation procedure for the various contractors. However, most of the other specific workshops suggested, such as Database Question-Answering, Generation, Knowledge Representation, and Pragmatics and Discourse do not have any funding sources available. The most difficult problems remain unresolved. There are still large classes of phenomena that have yet to be characterized in a scholarly fashion, and we do not have adequate methods for measuring progress of a system under development.

A fundamental underlying snag is the difficulty in arriving at a consensus on the nature of semantic representation. If the community was in agreement on what the representation of a sentence is supposed to be—whether it was a sentence from a dialogue with an expert system, a sentence fragment from a tactical message, or a database query—then the task of assessing a system's performance would be much more straightforward. Given input X, does the system produce Y as an internal data structure? Unfortunately, there are now as many Y's for X as there are systems, so finding a reliable method of assessing a system in isolation, or of comparing two systems, becomes much more difficult. It is necessary to define the evaluation in terms of a task that is being performed (Sundheim 1989; Napier 1989). Then the system's score with respect to natural language competence becomes dependent on how well the system as a whole can elicit information from the expert system or the database, or can summarize the information in the message. Task-oriented black-box evaluations are useful and valid, and are certainly of primary concern to the end users who need the information, and do not really care how it is produced. But there are drawbacks in depending solely on this approach. A system's capabilities cannot be measured or compared until it has been completely integrated with a target application. For any interesting application, this requires a major investment in a domain model and in a domain semantics, not to mention all of the application-specific needs around user friendliness and informative displays of information, etc. Designing the task-oriented test can require a major investment as well (Sundheim 1989). This is an extremely expensive and time-consuming enterprise that few organizations can indulge in. The result is that there are very few systems that are fully integrated with target applications in such a way that an appropriate task-oriented evaluation can be performed. There is no way to test whether or not a system is suitable for a particular application without actually building the application. There are no accepted guidelines that system developers can use to measure the progress being made by a fledgling system from month to month. Granted that a task-oriented evaluation is necessary and sufficient for a system that is ready for end-users, it does not solve the problem of charting a system's progress along the way toward a particular application.

References

Ball, Catherine N. 1989 Analyzing Explicitly-Structured Discourse in a Limited Domain: Trouble and Failure Reports. *Proceedings of the DARPA Speech and Natural Language Workshop* Philadelphia, PA.

Brennan, Susan E., Friedman, Marilyn W., and Pollard, Carl. J. 1987 A Centering Approach to Pronouns. *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, 155–162.

Grosz, Barbara, Joshi, Aravind, and Weinstein, Scott. 1983 Providing a Unified Account of Definite Noun Phrases in Discourse. *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics,* 44–50.

Joshi, A. K., and Weinstein, S. 1981 Control of Inference: Centering. *Proceedings of the 7th International Conference on Artificial Intelligence,* 385–387.

Napier, H. Albert. 1989 The Impact of a Restricted Natural Language Interface on Ease of Learning and Productivity. *Communications of the ACM,* 32(10):1190–1198.

Sundheim, B. M. 1989 Plans for a Task-Oriented Evaluation of Natural Language Understanding Systems. *Proceedings of the DARPA Speech and Natural Language Workshop.* 197–202.

Webber, Bonnie Lynn. 1988 Discourse Canon. Presented at the Mohonk Darpa Workshop, May, 1988.