Verification and Validation of Language Processing Systems: Is It Evaluation?

Valerie B. Barr
Department of Computer Science
Hofstra University
Hempstead, NY 11549-1030 USA
vbarr@hofstra.edu

Judith L. Klavans
Center for Research on Information Access
Columbia University
535 West 114th Street, MC 1103
New York, NY 10027 USA
klavans@cs.columbia.edu

Abstract

If Natural Language Processing (NLP) systems are viewed as intelligent systems then we should be able to make use of verification and validation (V&V) approaches and methods that have been developed in the intelligent systems community. This paper addresses language engineering infrastructure issues by considering whether standard V&V methods are fundamentally different than the evaluation practices commonly used for NLP systems, and proposes practical approaches for applying V&V in the context of language processing systems. We argue that evaluation, as it is performed in the NL community, can be improved by supplementing it with methods from the V&V community.

1 NLP Systems as Intelligent Systems

Language engineering research is carried out in areas such as speech recognition, natural language understanding, natural language generation, speech synthesis, information retrieval, information extraction, and inference (Jurafsky & Martin, 2000). In practice this means building systems which model human activities in various language processing tasks. Therefore, we can quite clearly view language processing systems as forms of intelligent systems. This view allows us to draw on work that has been done within the intelligent systems community within computer science on verification and validation of systems. It also allows us to consider V&V in the context of NL systems, and evaluation, as carried out on NL systems, in the context of software engineering methodologies. This research extends the first author's earlier work on software testing methodologies in the context of expert systems (Barr, 1995; Barr, 1999).

2 Verification and Validation of Intelligent Systems

The area of verification and validation of software systems has suffered from a multiplicity of definitions (Barr, 2001; Gonzalez and Barr, 2000). However, the most commonly used definitions are :

- Verification – ensuring that software correctly implements specific functions, that it satisfies its specification.
- Validation – determining that the system satisfies customer requirements.

These definitions have been re-examined in order to account for the differences between 'conventional' software and intelligent systems. An intelligent system is built based on an interpretation of the problem domain, with the expectation that the system will behave in a fashion that is equivalent to the behavior of an expert in the domain. It follows that human performance is often the benchmark we use to evaluate an intelligent system.

The usual definitions of verification and validation can be applied to intelligent systems with slight modifications to take into account the presence of a knowledge base and the necessity of comparing system performance to that of humans in the problem domain. The core issue in validation and verification of an intelligent system boils down to one simple objective: ensuring that the resulting system will

provide an answer, solution or behavior equivalent to what an expert in the field would say if given the same inputs.

Therefore the definitions of verification and validation have been refined (Gonzalez and Barr, 2000) in order to account for the differing aspects of intelligent systems :

- Verification – the process of ensuring 1) that the intelligent system conforms to specifications, and 2) its knowledge base is consistent and complete within itself.
- Validation – the process of ensuring that the output of the intelligent system is equivalent to those of human experts when given the same inputs.

Consistency of the knowledge base means that there are no redundancies, conflicts or cycles. *Completeness* means that all facts are used, there are no unreachable conclusions, missing rules (in the rule-based expert systems context), there are no dangling conditions. These definitions of verification and validation retain the standard definitions used in software engineering, while also requiring that the knowledge base be free of internal errors, and letting human performance be the standard for 'customer requirements'.

In the realm of language processing, the 'expert' can often be any user of language in the context for which the system has been developed.

3 Evaluation of NLP Systems

The previous section presented definitions for V&V. In this section the general paradigms for evaluation of NLP system is presented.

3.1 Taxonomies of Evaluation within NLP

Our review of the evaluation literature indicates that NLP systems have largely been evaluated using a **black-box, functional**, approach. Evaluation is often subdivided into **formative** evaluation and **summative** evaluation (Sparck Jones & Galliers, 1996). The former determines if the system meets the objectives that were set for it. It can be diagnostic, indicating areas in which the system does not perform well, or predictive, indicating the performance that can be expected in actual use. Summative evaluation is a comparison of different systems or approaches for solving a single problem.

In a somewhat different taxonomy (Hirschman and Thompson, 1998), evaluation is subdivided into

- **Adequacy evaluation** – determination of the fitness of a system for its intended purpose. Will it do what is required by the user, how well, and at what cost?
- **Diagnostic evaluation** – exposure of system failures and production of a system performance profile.
- **Performance evaluation** – measurement of system performance in one or more specific areas. Can be used to compare alternative implementations or successive generations of a system.

We can see that performance evaluation overlaps with summative evaluation, while adequacy evaluation corresponds to formative evaluation.

While the evaluation process must consider the results generated by an NLP system, it also considers the **usability** of the system (Hirschman and Thompson, 1998; White and Taylor, 1998), its features, and how easily it can be enhanced. For example, a translation system may appear to work well in a testbed situation, but may not function well when embedded into a larger system. Or it may perform acceptably when its output is intended for a general audience, but not when an expert uses the output.

Sparck Jones and Galliers (1996) discuss how the evaluation process should take into account whether the NLP task is part of a larger system with both linguistic and non-linguistic components, and determine the impact on overall performance of each of the subparts. We call this **component performance** evaluation. Additional complexity arises in the evaluation of component performance within multi-faceted systems, such as embodied conversational agents, where assessment of how well the system works is based on more than strict language aspects, considering also more subtle features such as gesture and tone (Cassell *et al.*, 2000). Furthermore, whether or not a system response is considered to be correct or acceptable may depend on who is judging it.

In general, NLP systems for various kinds of tasks require differing views of the evaluation process, with different criteria, measures, and

methods. For example, consider the ways in which evaluation of machine translation (MT) systems is carried out. Notice that not all aspects of validation and verification, as discussed in section 2, are represented. Evaluation of machine translation (MT) systems has to consider the pre-processing of input and the post-editing of output. Black-box evaluation of MT systems can measure the percentage of words that are incorrect in the entire output text (based on how post-editing changes raw output text to fix it). But whether or not a word is considered incorrect in the output may depend on the task of the system. So functional evaluation of an MT system may have to be augmented by a subjective determination of whether the output text carries the same information as the input text, and whether the output is intelligible (Sparck Jones and Galliers 1996). Another example is the case of speech interfaces and spoken dialogue systems. the evaluation process typically focuses on the accuracy, coverage, and speed of the system, with increasing attention paid to user satisfaction (James et al. 2000, Walker and Hirschman 1999). Notice that in just these two examples, various kinds of evaluation are called into play. We will argue in section 4 that V&V techniques extend these evaluation methods, providing system coverage analysis that assesses completeness and consistency.

3.2 Factors which impact evaluation

Evaluation of NLP systems must also take into account the kinds of inputs we expect a system to work on after its testing phase is complete. Wacholder (1997) demonstrates the extent to which the linguistic complexity of documents is one of the factors responsible for the weakness of applications that process natural language texts. The ability to categorize test data by complexity will help distinguish between a failure of an NLP system that results from extraordinary document complexity (beyond that of the data on which the system was tested) and a failure that results from inadequate testing of the NLP tool. The former should be predictable, while the latter should rarely happen if a system has been adequately tested. It is certainly possible that a tool may be very well tested, functionally and with regard to consistency and completeness, on text of certain degree of complexity, but still fail on text that is more complex or from a different domain.

3.3 Comparative evaluation efforts

There are NLP evaluation methods that, although in a different problem domain, closely mirror the approach typically used with expert systems, comparing machine results to human results. For example, the TAUM-AVIATION machine translation system was evaluated in 1980, in part by comparing the raw translation produced by the system to several human translations. Then revised and post-edited translations (human and machine) were rated and ranked by a number of potential users (Sparck Jones and Galliers, 1996). This is essentially the same testing method that was used for the MYCIN expert system (Yu, 1985) and many additional systems. However, within the expert systems area several methods have been developed in subsequent years that address the weaknesses of strictly functional evaluation approaches (e.g. Barr, 1999; Grossner, 1993).

There are also well-known evaluation efforts such as EAGLES (Sparck Jones and Galliers, 1996) and the Paradise evaluation framework (Walker *et al.*, 1997). In addition, many researchers have participated in the comparative evaluation efforts characterized by the Text Retrieval Conferences (TREC)[1], the Message Understanding Conferences (MUC)[2] and Document Understanding Conferences (DUC)[3], the Cross-Language Evaluation Forum (CLEF)[4], and the summarization evaluation effort (SUMMAC) (for a very comprehensive list of evaluation related links, see http://www.limsi.fr/TLP/CLASS/prj.eval.links.html).

Evaluation of NLP systems is aided by the fact that there is considerable test data available. There are substantial repositories of data, such as the TREC collection that includes, among other data, Associated Press wire feeds; Department of Energy documents; Federal Register documents; Wall Street Journal full texts; and sources from Ziff-Davis Publishing.

---

[1] http://trec.nist.gov/

[2] http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/muc.htm

[3] http://www-nlpir.nist.gov/projects/duc/index.html

[4] http://www.iei.i.cnr.it/DELOS/CLEF/

It is important to note that the DARPA/ARPA sponsored conferences (MUC, TIPSTER, and TREC, for example), while making considerable data available, promote functional testing by stressing black-box performance of a system. The metrics used in the MUC program are oriented toward functional testing, focusing on the number of spots in a template that are correctly filled in by a particular MUC system, along with various error-based measures. For database query systems the emphasis has been on functional testing, supplemented with evaluations of the system by users, given the desire to create marketable systems.

An issue that arises in comparative evaluation efforts, particularly because there is so much test data available, is what it means to compare the behavior of two systems designed to carry out the same task, based on their performance on a common set of test data. Allen (1995) argues that evaluation results for individual systems, and any comparison of results across systems, should not be given much credence until they reach "some reasonably high level of performance." Certainly the MUC and TREC programs are based on comparing performance of multiple systems on a common task. One of the purposes of our research is to show that without assessment of consistency and completeness, the quality of the functional testing alone may not be sufficient for predicting reliability of an NLP system and V&V methods will improve the situation.

3.4 Additional comments on functional testing

We have referred to functional testing in prior paragraphs in the context of various aspects of evaluation. Recent literature (Declerk *et al*., 1998; Rubio *et al*., 1998; Klavans *et al*., 1998; Jing *et al*., 1998) shows that functional testing is still very much in use for evaluation of NLP systems and larger systems of which NLP components are a part. Where other evaluation mechanisms are in use, they are still based on the behavior of the system under test, not based on an analysis of how test case execution exercises the system. For example, White and Taylor (1998) propose evaluating machine translation (MT) systems based on what kind of text handling tasks could be supported by the output of the MT system. They examine the text handling tasks (publishing, gisting, extraction, triage, detection, filtering) to determine how good a translation has to be in order for it to be useful for each task. They then rank the text handling tasks in such a way that if an MT system's output can facilitate a task, it can also facilitate tasks lower on the scale, but is unlikely to facilitate tasks higher on the scale. This kind of evaluation is functional in nature, though the assessment of the quality of the MT system's output is based not on an examination of the output but on a subsequent use of the output.

The notion of functional glass-box testing does not assess coverage of the system itself, but is essentially an assessment of how well a system carries out its task. It relies on the programmer or tester's idea of how a component should carry out its task for a particular test input (Sparck Jones and Galliers 1996). At the same time, black-box evaluation is a very important and powerful testing approach, particularly because it works from the perspective of the user, without concern for implementation.

4 Applying V&V to NLP – Is it Evaluation?

In the previous section we outlined many different types of evaluation that are performed on NL systems. Our claim at the beginning of the paper was that evaluation, as it is performed in the NL community, can be improved by adopting V&V approaches. In this section we show specifically what the relationship is between V&V, as it is typically applied in software development, and evaluation as it is carreid out in the context of NLP systems.

In considering whether V&V and evaluation are equivalent, we need to consider whether the evaluation process achieves the goals of verification and validation. That is, does the evaluation process demonstrate that

- the system is correct and conforms to its specification
- the knowledge inherent in the system is consistent and complete
- the output is equivalent to that of human 'experts'.

It is apparent that summative, adequacy and diagnostic evaluation are all in some way equivalent to validation. The evaluation steps

involve black-box exercise of test data through the system, which then allows for a comparison of actual results to expected results. This facilitates an assessment of whether the output is equivalent to that of human experts (who provide the expected results).

The usual evaluation processes, through formative evaluation, also facilitate one aspect of verification, in that they allow us to determine if a system conforms to its specification. That is, based on the specification for a system, a domain-based test set can be constructed for evaluation which will then demonstrate whether or not a system meets the specification.

It is the second aspect of verification, determining whether the knowledge represented within the system is consistent and complete, that seems not to be taken into account by the evaluation processes in NLP. The difficulty lies in the fact that a domain based test set can never completely test the actual system as built. Rather, it tests the linguistic assumptions that motivated construction of the system. A domain based test set can determine if the system behaves correctly over the test data, but may not adequately test the full system. In particular, any inconsistencies in the knowledge represented within the system, or missing knowledge, may not be identified by an evaluation process that relies on domain-based test data.

To address this issue, we need to apply additional testing techniques, based on coverage of the actual system, in order to achieve the full breadth of verification activities on a language processing system. Furthermore, we may not need larger test sets, but we may need different test cases in the test set.

5 Applying V&V to NLP – How Do We Do It?

In our research we are in the early stages of experiments wherein we apply existing V&V tools to a number of NL systems for indexing and significant topics detection. We expect the results of these experiments will support our claim that V&V techniques will positively enhance the evaluation process.

The actual software testing tools we have chosen are based on the implementation paradigms that are used in the specific NL systems. For example, for a C based system for automatic indexing (Wacholder *et al*., 2001), we have selected the Panorama C/C++ package[5]. Various features of this tool facilitate testing of the system as built, based on code coverage rather than domain coverage. This approach effectively tests the knowledge base for consistency and completeness, which we cannot do as successfully with a domain based test set.

Regular expressions are frequently used to implement components of NL systems. We are studying a component (Evans *et al*., 2000) of a significant topics identification system that uses regular expressions. The software testing community has not yet developed tools for addressing coverage (completeness and consistency) testing of regular expressions. In this case we will construct a tool, building on theoretical work (Yannakakis and Lee, 1995) that has been carried out in the network protocol research community for testing finite-state machines.

Clearly we propose adding additional steps to the testing process. However, this does not necessarily imply that huge amounts of additional test data will be necessary. In a typical testing protocol, a developer can start the V&V phase with a domain based test set. Additional test cases are then added incrementally as needed until the test set is adequate for coverage of the system as built, and for assessment of the consistency and completeness of the system's knowledge.

6 Open Questions

The question we intend to address in future research is whether different natural language application areas can profitably benefit from different techniques utilized in the software testing/V&V world. For example, the issues involved with grammars and parsers are undoubtedly quite different from those that come into play in machine translation systems. With grammars and parsers it is quite tempting to test the grammar by running the parser and vice versa. Yet the grammar and parser are essentially built concurrently, and an error in one would easily carry over as an error in the other. Typical testing strategies make it quite

[5] http://www.softwareautomation.com

difficult to expose these sorts of errors. A testing approach is necessary which will help expose errors or incompletenesses that exist in both a grammar and its parser. An effort by Bröker (2000) applies code instrumentation techniques to grammar analysis. However, the kinds of errors that may occur in a translation system or a language generation system are of a different nature and will require different testing strategies to expose.

7 Acknowledgements

This research was partially supported by the National Science Foundation under NSF POWRE grant #9973855. We also thank Bonnie Webber, of the University of Edinburgh, and the Columbia University Natural Language Processing Group, particularly Kathy McKeown and Nina Wacholder, for their helpful discussions.

References

Allen, James (1995). Natural Language Understanding. Benjamin/Cummings, Redwood City, CA.

Barr, Valerie (1995). TRUBAC: A Tool for Testing Expert Systems with Rule-Base Coverage Measures. *Proceedings of the 13th Annual Pacific Northwest Software Quality Conference*, Portland, OR.

Barr, Valerie (1999). Applications of Rule-Base Coverage Measures to Expert System Evaluation. *Journal of Knowledge Based Systems*, Volume 12 (1999), pp. 27-35.

Barr, Valerie (2001). A quagmire of terminology. *Proceedings of Florida Artificial Intelligence Research Symposium 2001.*

Bröker, Norbert (2000). The use of instrumentation in grammar engineering. *Proceedings of COLING 2000.*

Cassell, Justine, Joseph Sullivan, Scott Prevost, and Elizabeth Churchill (2000). Embodied Conversational Agents. MIT Press, Cambridge, MA.

Declerk, Thierry *et al.* (1998). Evaluation of the NLP Components of an Information Extraction System for German. *Proceedings of the 1st International Conference on Language Resources and Evaluation*, Granada, Spain, May 1998, pgs. 293-297.

Evans, David K. *et al.* (2000). Document Processing with LinkIT, *Proceedings of RIAO 2000* (Recherche d'Informations Assistee par Ordinateur), Paris.

Gonzalez, Avelino and Valerie Barr (2000). Validation and verification of intelligent systems – what are they and how are they different? *Journal of Experimental and Theoretical Artificial Intelligence*, 12(4).

Grossner, C. *et al.* (1993). Exploring the structure of rule based systems. *Proceedings, AAAI-93*, Washington, D.C., pp. 704-709.

Hirschman, Lynette and Henry S. Thompson (1998). Overview of Evaluation in Speech and Natural Language Processing in *Survey of the State of the Art in Human Language Technology*, Giovanni Varile and Antonio Zampolli, eds., Cambridge University Press, New York.

James, Frankie *et al.* (2000). Accuracy, Coverage, and Speed: What Do They Mean to Users? See http://www.riacs.edu/doc/2000/htmlo/chi_nl_worksh op.html

Jing, Hongyan *et al.* (1998). Summarization Evaluation Methods: Experiments and Analysis. *AAAI Symposium on Intelligent Summarization*, March 1998, Stanford University.

Jurafsky, Daniel and James Martin (2000). *Speech and Language Processing.* Prentice-Hall, NJ.

Klavans, Judith L., Kathleen McKeown, Min-Yen Kan, and Susan Lee (1998). Resources for Evaluation of Summarization Techniques. *Proceedings of the First International Conference on Language Resources and Evaluation*, Granada, Spain, 1998, pgs. 899-902.

Rubio, A. *et.al.* (1998). On the Comparison of Speech Recognition Tasks. *Proceedings of the First International Conference on Language Resources and Evaluation*, Granada, Spain, 1998.

Sparck Jones, Karen and Julia Galliers (1996). *Evaluating Natural Language Processing Systems*. Springer-Verlag, Berlin.

Wacholder, Nina. (1997). POWRE: Computationally Tractable Methods for Document Analysis. CS Report, Dept. of Computer Science, Columbia University (NSF funded project).

Wacholder, Nina, *et al.* (2001). Automatic Generation of Indexes for Digital Libraries. *IEEE-ACM Joint Conference on Digital Libraries.*

Walker, M., *et.al.* (1997). PARADISE: A Framework for evaluating spoken dialogue agents. *Proceedings of Association of Computational Linguists 35th Annual Meeting*.

Walker, M. and L. Hirschman (1999). *DARPA Communicator Evaluation Proposal*. www.research.att.com/~walker/eval/evalplan6.rtf

White, John S. and Kathryn B. Taylor (1998). A Task-Oriented Evaluation Metric for Machine Translation. *Proceedings of the First International Conference on Language Resources and Evaluation*, Granada, Spain, May 1998, pgs. 21-25.

Yannakakis, Mihalis and David Lee (1995). Testing Finite State Machines : Fault Detection . *Journal of Computer and Systems Sciences*, Volume 50, pages 209-227.

Yu, V.L. *et.al.* (1985). An evaluation of MYCIN's advice. In *Rule-Based Expert Systems*, Bruce Buchanan and Edward Shortliffe (Eds.), Addison-Wesley, Reading, MA.