A Proposed Model for Effective Verification of Natural Language Generation Systems

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

Introduction

Natural language processing (NLP) research is carried out in areas such as speech recognition, natural language understanding (NLU), natural language generation (NLG), speech synthesis, information retrieval, information extraction, and inference (Jurafsky & Martin 2000). NLP components are being incorporated into a variety of systems, and NLP methods are being used in new application areas. There is increasing interest in dialogue systems and language generation systems (e.g. the ELVIS system for voice access to email and voicemail (Walker 2000), the embodied conversational agent REA (Cassell, Stone, & Yan 2000; Cassell 2000), as well as (Litman & Pan 1999; McKeown *et al.* 1997)), text analysis systems (Klavans & Wacholder 1997), summarization systems (McKeown & Klavans 1997), information extraction systems (Declerk, Klein, & Neumann 1998), and text-to-speech systems (Sproat 1997).

In practice these activities require building systems that model human activities in various language processing tasks. Therefore, we can view language processing systems as intelligent systems. These uses, furthermore, increase the need for thorough testing of NLP systems and individual NLP components that are embedded in larger systems.

Language processing researchers have not generally carried out the sorts of verification and validation activities that are typically attempted in the intelligent systems research area. The research presented here is part of a larger project that is considering how verification and validation can be carried out for language processing systems in different application areas. In specific, research is needed to achieve the following:

- develop a clear definition of what is encompassed by the verification task for language processing software, building on foundations in both the intelligent systems and software engineering/software testing areas.
- develop techniques and tools for carrying out verification of language processing software, as appropriate for various application areas within language processing (such as generation, understanding, dialogue).
- develop a theory of how to apply the results of verification of language processing software to the task of validating of language processing systems, which will facilitate the development of more robust and predictable systems.

In this paper we present a first stage for effective verification of natural language generation (NLG) systems, focusing on a specific strategy for the verification of tree adjoining grammars (TAGs) as used in NLG systems.

Overview of Verification and Software Testing

Earlier research (Barr 2001; Barr & Klavans 2001) showed that software testing, as it is usually carried out, and evaluation, as typically carried out within the context of NLP systems development, are largely distinct activities that can both be used during the development of NLP systems. We propose to combine traditional NLP system evaluation methods with new verification approaches that are based on traditional software testing approaches as well as the verification and validation of intelligent systems.

In the intelligent systems community the overall goal of verification is to ensure that the system conforms to specifications and is consistent and complete within itself (Gonzalez & Barr 2000). In the software engineering community testing activities are incorporated into the verification process (Voas & Miller 1995). Verification is defined as comprising three general activities: dynamic software testing, software testability, and formal verification (typically using a static theorem prover). Here we focus primarily on dynamic software testing. We use the term "testing" to refer only to those testing activities that are a part of the larger verification process.

Dynamic software testing is made up of black-box (functional) testing and white box (structural) testing. Black-box testing consists of a testing-debugging cycle, where the internals of a system are considered only to the extent necessary to determine how to correct faulty behavior on some test case. Typical evaluation of NLP systems is essentially black box testing, as it is highly functional in nature, with test data based solely on the application domain.

By contrast, white-box testing uses test-adequacy criteria (based on coverage of code or system properties) to assess the quality of the test suite and the thoroughness of the testing. A complete approach to software testing will incorporate both black-box and white-box analysis of a program, which allows us to identify incompleteness in the test data and potential errors in the program itself.

Typical NLP system evaluation does not include the adequacy criteria that are commonly part of a systematic verification process. Furthermore, the nature of NLP systems introduces additional complexity into the verification problem. For language processing systems it is not easy to specify what should and should not be acceptable input, and what should and should not be acceptable output. In addition, an NLP system can potentially be used in multiple user contexts, and must be judged accordingly in order to gauge application dependent usability.

Verification of NLP Systems

We would like to derive the benefits of white-box testing within the context of verification of NLP systems. The tasks undertaken by NLP systems differ from those undertaken by other kinds of software systems, and the operation of NLP systems is different as well, which limits the applicability of programming language dependent testing tools. With conventional software a dataflow scheme can be used to determine the precise set of operations that lead to a particular computation or result. However, in NLP systems it is not always clear to the designers *a priori* what information is relevant for a particular decision. In fact, understanding this may be an intended side effect of building the system. Moreover, in some situations, there may be multiple correct results that a system could produce. This is particularly the case in language generation systems where there may be numerous correct realizations of a text specification.

Certainly for all software systems we would like to determine whether each possible input leads to some output, which can be determined by a black-box testing approach. In the realm of language processing we are also interested in determining if, for a given output, there is a single input or set of inputs that can produce it. This task corresponds to the creation of equivalence classes among the input domain, as might be done within the context of software reliability engineering (Lyu 1996).

A point of concern in the language processing realm is whether the system will place into the same equivalence class entities that should not generate the same output realization. How we view this issue also depends on which aspect of NLP we are considering. For example, in language generation a given input to a system comprises several parts: the knowledge source, the communicative goal, the user model, and the discourse history (Reiter & Dale 2000). The knowledge source will remain fixed over all uses of the NLG system (though it should be verified separately). The user model may be fixed or not, depending on whether the system is designed to generate text for multiple user groups or only a single monolithic user. However, the communicative goal and discourse history change for each invocation of the system. Therefore, the set of possible input configurations is based on the set of communicative goals that the system can handle as well as the different interaction sequences that can be represented by the discourse history.

There are additional questions of interest about the relationship of input to output in the language processing realm that do not occur in other areas of software systems. For example, can we determine whether there will be more than one possible output for a given input, and do the multiple outputs form an equivalence class? How we view this also depends on whether the system is doing analysis, as in language understanding, or synthesis, as in language generation. In an analysis situation the existence of multiple outputs for a single input usually implies ambiguity on the part of the input. In synthesis multiple outputs can be equally acceptable. In language generation, multiple outputs could represent equivalent acceptable paraphrases generated from a single input, based on a single set of underlying concepts from the knowledge source and the discourse history. We would also like to be able to determine which parts of a structured input actually play a role in producing the corresponding output, which parts play no role, and what we can infer about system quality from that information.

Developing a Model For Effective Verification of NLG Systems

Dale and Mellish (Dale & Mellish 1998) have focused on the problem of natural language generation (NLG), as distinct from natural language understanding (NLU), and suggest a direction for improving evaluation of NLG systems. Their proposal is that, rather than attempt to evaluate a complete system, the evaluation effort address the component tasks of the NLG process. They suggest a breakdown of the NLG process (Reiter & Dale 2000; Dale & Mellish 1998) into the six tasks of content determination, document structuring, lexical selection, referring expression generation, aggregation, and surface realization.

This breakdown, combined with traditional approaches to verification, leads to several sub-areas in which we can address the issue of effective verification of NLG systems. We note that the task of surface realization involves language-specific linearization of a representation, and will not be considered in the context of this work on verification.

Content determination

Dale and Mellish suggest that the area of content determination may well benefit from work done on the evaluation of expert systems. The fundamental question of content determination is that of what information should be conveyed in a generated text, which is really the question of what an expert in the domain area would convey in the given situation. In many respects this is a validation task, not a verification task, and it may benefit from work done on validation of intelligent systems.

Document structuring

In document structuring we are concerned with the extent to which the coherence of generated texts reflects the coherence found in human generated text. The document structuring component of a generation system represents an implementation of a theory of text coherence. The verification task is to determine whether the implementation of the coherence theory is correct. This cannot be set up as a completely mechanical task, as it must also take into account the discourse relations between elements of text, and the hierarchical relations that hold between more general and more specific elements within the knowledge source of the system input.

Lexical selection

Lexical selection is the component of NLG that most directly leads to the possibility of an output equivalence class, multiple acceptable outputs for a given input. The process of choosing the words that convey intended content can also involve linguistic structures and the user model. From the verification standpoint, we must determine whether all possible lexicalizations of the selected content for a communicative goal form an equivalence class, associated with the same set of underlying concepts from the knowledge source and responding to the same collocation factors within the discourse history.

Generation of Referring Expressions (GRE)

At issue in GRE is how the system uses information about an entity to refer to it in order to generate text that is not unnecessarily awkward or redundant. As with lexical selection, GRE involves discourse history as well as the knowledge source. While it is important in this case, as well, to ensure that all possible referring expressions for an entity form an equivalence class, it is also important that the referring expression(s) chosen is sufficiently unambiguous and brief.

Aggregation

Aggregation involves developing and ordering linguistic structures and textual elements based on the information provided from the document structuring task. In many NLG system architectures, aggregation is combined with lexical selection and referring expression generation to form a single sentence planning task. Certainly it is the case that there are multiple ways that information can be aggregated to form a coherent text that meets the communicative goal. Dale and Mellish argue (Dale & Mellish 1998) that the interactions between these three subtasks are so poorly understood that researchers are not yet ready to do evaluation of the separate subtasks. However, precisely because all three subtasks involve the knowledge source, discourse history, as well as the communicative goal, it may be possible to develop a verification approach for each of them, based on the concept of equivalence classes of results. Verification of sentence planning as a higher level task may be computationally complex because it effectively involves the cross product of all possible lexicalizations, referring expressions, and aggregations (recognizing that not all combinations are possible but that each lexicalization allows for certain referring expressions, and each allowable lexicalization-referring expression pair then allows certain aggregations).

Resource Use

Another area of interest, which cannot be directly assessed by traditional testing approaches, is that of how linguistic resources are utilized by a language processing system. Typically a language processing system has numerous resources within it, such as a lexicon, a grammar, morphological rules, a pragmatics component, and semantic knowledge (both formal and lexical). We would like to assess completeness and consistency of the grammar alone. Beyond the grammar, we are interested in the role of other linguistic resources as well. Ultimately we intend to define what it means to evaluate all the linguistic resources for completeness and consistency. For example, it would be useful to clarify exactly how an incomplete lexicon affects system behavior. There may be sub-processes, within a language generation system, for example, that should be verified separately because they utilize only a subset of the available resources. We are also interested in how the various resources participate in the input-output relationship. For example, can we determine which of a system's linguistic resources contributes to the transformation of an input to an output? Can we pinpoint exactly how each element of an output is affected by each linguistic resource? If the grammar in a generation system is capable of parsing a sentence, is there some context in which the system will generate that sentence? Developing mechanisms for addressing these issues will enable us to more accurately assess the overarching verification issue, which is whether the system does the task, and only the task, for which it was intended.

Grammar Verification

As a first concrete step in developing a verification approach, we address the verification of a grammar, analyzing for consistency and completeness. We cannot necessarily do this by applying existing methods. How we do it depends on the kind of grammar used. We have, from the expert systems' realm, methods and tools that are suitable for rule-based systems (for example, the TRUBAC tool (Barr 1999)). However, the rule formalism, while used in some aspects of NLP, is frequently not used for grammar representation. Yet adapting to the grammar of an NLP system the underlying approach used for rule-based systems may give us the ability to determine which parts of a grammar have not been exercised by a given input test set, and also to generate input that will exercise parts of the grammar and combinations of grammatical constructions that were not exploited by a particular test suite.

We focus on the use of Lexicalized Tree Adjoining Grammars (LTAG), based on the original TAG formalism (Joshi, Levy, & Takahashi 1975; Joshi 1987; Joshi & Schabes 1997). TAG is a mathematical formalism that can be used, in conjunction with a linguistic theory, for linguistic descriptions. TAG was developed as a more powerful formalism than context free grammars (CFGs), and is based on tree rewriting, not string rewriting. In a TAG the elementary structure is a tree, not the flat rules found in CFGs. TAG allows for an extended domain of locality, thereby facilitating agreement between different parts of a tree structure (such as subject-verb agreement) that cannot be easily enforced with a CFG.

Formally, a TAG (Joshi & Schabes 1997) consists of a quintuple (Σ, NT, I, A, S), made up, respectively, of a finite set of terminal symbols, a finite set of nonterminal symbols, a finite set of finite *initial trees*, a finite set of finite *auxiliary trees*, and a distinguished nonterminal symbol S.

An auxiliary tree has nonterminal symbols labeling interior nodes and one "foot node" on the frontier which allows for adjunction (other nonterminal frontier nodes are marked for substitution). An initial tree has interior nodes labeled by nonterminal symbols, and nonterminal nodes on the frontier are labeled for substitution.

When an LTAG is used, each lexical item (a word and its part-of-speech) has an associated family of LTAG trees, each of which has the lexical item as its "anchor", and each elementary structure in the grammar is associated with a lexical item (terminal item, or word of the language). The TAG rules for adjunction and substitution operations determine the ways in which these trees can be combined together to make complex structures that represent sentences. The adjunction operation builds a new tree from an auxiliary tree and another tree (of any type). This operation is used to add modifiers that provide syntactically-optional elaborations. Substitution takes place only at nonterminal nodes of the frontier of a tree and provides complements or arguments that are syntactically required. In a language generation system, at each step the generation algorithm will choose one out of the adjunction or substitution possibilities.

In natural language generation the goal is to go from an internal representation to a string of text. The input to the system will include a knowledge source, a communicative goal, a user model and the discourse history. LTAG is a good choice for generation systems because of the way the grammar is organized, the full lexicalization, and the extended domain of locality. These features allow the LTAG to be a factor in word choice and the mapping of semantic relations to syntactic relations. It is also possible, as is the case in the SPUD system (Stone *et al.* 2001), to use the LTAG to handle lexical choice, referring expression generation, and sentence planning. Therefore, if we can verify the LTAG in some way then we will gain valuable information about the quality of several steps of the generation process.

There are numerous questions that we hope to eventually address as part of the verification of generation systems, particularly those based on LTAGs. Initially, we propose analyzing the grammar for consistency and completeness. Given a grammar made up of trees (rather than rules), we cannot directly use the characteristics that are used in evaluating rule-bases for consistency and completeness (conflict, redundancy, circularity, subsumption, unreachability, dead-ends, etc.), but rather must adapt the concepts of completeness and consistency for the LTAG realm. We propose the analysis of the consistency and completeness of an LTAG as a suitable first step to carrying out verification and validation of the generation system in which the LTAG is used.

We propose checking the following characteristics of an LTAG:

- all trees have an anchor (the LTAG is properly lexicalized)
- there are no adjunction points in initial trees
- every auxiliary tree can participate in some adjunction operation with other tree(s) in the LTAG
- every initial tree can participate in some substitution operation
- every tree has a use in some adjunction or substitution operation

This set of checks will serve as the first stage of verification analysis for an LTAG. We note, however, that all of these checks are strictly structural in nature, and do not in any way guarantee semantic coherence in the sentences that could be generated by the LTAG under analysis. In the next stage of this work we will extend the approach to feature-based LTAG, where elements of semantic coherence are enforced within the structure of the grammar components, so that a verified grammar is more likely to generate semantically coherent sentences.

We are currently pursuing this approach in the context of the SPUD language generation system(Stone *et al.* 2001). We will start with a simple grammar, and then move on to a grammar that includes generation of text combined with gestures for an embodied conversational agent(Cassell, Stone, & Yan 2000).

Conclusion

In this paper we proposed an approach for effective verification of NLG systems, based on conventional evaluation of language processing systems combined with software testing and system verification concepts drawn from the intelligent systems community. We presented a first step in the implementation of the verification strategy in the form of a specific strategy for the verification of lexicalized tree adjoining grammars (LTAGs) as used in natural language generation systems. In future work we will determine the usefulness of this approach by testing it on a number of LTAGs that have been used in conjunction with a specific language generation system.

Acknowledgments

Thanks to Bonnie Webber and Matthew Stone for their thoughts about and comments on this work, and their continued interest in it.

References

Barr, V., and Klavans, J. 2001. Verification and validation of language processing systems: Is it evaluation? In *Proceedings of the Workshop on Evaluation Methodologies for Language and Dialogue Systems, ACL2001*. Toulouse, France: Association of Computational Linguists.

Barr, V. 1999. Applications of rule-base coverage measures to expert system evaluation. *Journal of Knowledge Based Systems* 12:27–35.

Barr, V. 2001. A quagmire of terminology: Verification & validation, testing, and evaluation. In *Proceedings of Florida Artificial Intelligence Research Symposium 2001*.

Cassell, J.; Stone, M.; and Yan, H. 2000. Coordination and context-dependence in the generation of embodied conversation. In *Proceedings of First International Conference on Natural Language Generation*, 171–178.

Cassell, J. 2000. Nudge nudge wink wink: Elements of face-to-face conversation for embodied conversational agents. In Cassell, J.; Sullivan, J.; Prevost, S.;

and Churchill, E., eds., *Embodied Conversational Agents*. Cambridge, MA: MIT Press. 1–27.

Dale, R., and Mellish, C. 1998. Towards evaluation in natural language generation. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*.

Declerk, T.; Klein, J.; and Neumann, G. 1998. Evaluation of the nlp components of an information extraction system for german. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, 293–297.

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

Joshi, A., and Schabes, Y. 1997. Tree-adjoining grammars. In Rozenberg, G., and Salomaa, A., eds., *Handbook of Formal Languages*, volume 3. Berlin: Springer. 69–124.

Joshi, A. K.; Levy, L. S.; and Takahashi, M. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences* 10:136–163.

Joshi, A. 1987. An introduction to tree adjoining grammars. In ManasterRamer, A., ed., *Mathematics of Language*. Amsterdam: John Benjamins. 87–113.

Jurafsky, D., and Martin, J. 2000. *Speech and Language Processing*. New Jersey: Prentice-Hall.

Klavans, J. L., and Wacholder, N. 1997. Automatic identification of significant topics in domain independent full text analysis. Technical report, Columbia University Dept. of Computer Science.

Litman, D., and Pan, S. 1999. Empirically evaluating an adaptable spoken dialogue system. In *Proceedings of the International Conference on User Modeling*.

Lyu, M. 1996. *Software Reliability Engineering*. New York, NY: McGraw-Hill.

McKeown, K. R., and Klavans, J. L. 1997. Stimulate: Generating coherent summaries of on-line documents: Combining statistical and symbolic techniques. Technical report, Columbia University Dept. of Computer Science.

McKeown, K.; Pan, S.; Shaw, J.; Jordan, D.; and Allen, B. 1997. Language generation for multimedia healthcare briefings. In *Proceedings of the Applied Natural Language Processing Conference (ANLP'97).*

Reiter, E., and Dale, R. 2000. *Building Natural Language Generation Systems*. Cambridge, UK: Cambridge University Press.

Sproat, R. 1997. *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*. New York: Kluwer.

Stone, M.; Doran, C.; Webber, B.; Bleam, T.; and Palmer, M. 2001. Microplanning with communicative intentions: The spud system. Technical report, arXiv:cs.CL/0104022.

Voas, J. M., and Miller, K. W. 1995. Software testability: The new verification. *IEEE Software* 12(3):17–28.

Walker, M. 2000. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research* 12:387–416.