A Test Environment for Natural Language Understanding Systems

Li Li, Deborah A. Dahl, Lewis M. Norton, Marcia C. Linebarger, Dongdong Chen
Unisys Corporation
2476 Swedesford Road
Malvern, PA 19355, U.S.A.
{Li.Li, Daborah.Dahl, Lewis.Norton, Marcia.Linebarger, Dong.Chen}@unisys.com

Abstract

The Natural Language Understanding Engine Test Environment (ETE) is a GUI software tool that aids in the development and maintenance of large, modular, natural language understanding (NLU) systems. Natural language understanding systems are composed of modules (such as part-of-speech taggers, parsers and semantic analyzers) which are difficult to test individually because of the complexity of their output data structures. Not only are the output data structures of the internal modules complex, but also many thousands of test items (messages or sentences) are required to provide a reasonable sample of the linguistic structures of a single human language, even if the language is restricted to a particular domain. The ETE assists in the management and analysis of the thousands of complex data structures created during natural language processing of a large corpus using relational database technology in a network environment.

Introduction

Because of the complexity of the internal data structures and the number of test cases involved in testing a natural language understanding system, evaluation of testing results by manual comparison of the internal data structures is very difficult. The difficulty of examining NLU systems in turn greatly increases the difficulty of developing and extending the coverage of these systems, both because as the system increases in coverage and complexity, extensions become progressively harder to assess and because loss of coverage of previously working test data becomes harder to detect.

The ETE addresses these problems by:

1. managing batch input of large numbers of test sentences or messages, whether spoken or written.
2. storing the NLU system output for a batch run into a database.
3. automatically comparing multiple levels of internal NLU data structures across batch runs of the same data with different engine versions. These data structures include part-of-speech tags, syntactic analyses, and semantic analyses.
4. flagging and displaying changed portions of these data structures for an analyst's attention.
5. providing access to a variety of database query options to allow an analyst to select inputs of potential interest, for example, those which took an abnormally long time to process, or those which contain certain words.
6. providing a means for the analyst to annotate and record the quality of the various intermediate data structures.
7. providing a basis for quantifying both regression and improvement in the NLU system.

1 Testing Natural Language Understanding Systems

Application level tests, in which the ability of the system to output the correct answer on a set of

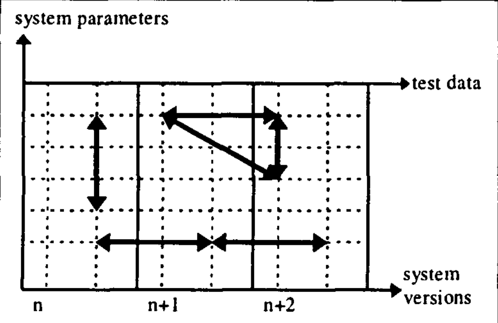

Figure 1: Matrix Comparison Analysis

inputs is measured, have been used in natural language processing for a number of years (ATIS-3 (1991), MUC-6 (1995), Harman and Voorhees (1996)). Although these tests were originally designed for comparing different systems, they can also be used to compare the performance of sequential versions of the same system. These kinds of *black-box* tests, while useful, do not provide insight into the correctness of the internal NLU data structures since they are only concerned with the end result, or answer provided by the system. They also require the implementation of a particular application against which to test. This can be time-consuming and also can give rise to the concern that the NLU processing will become slanted toward the particular test application as the developers attempt to improve the system's performance on that application.

The Parseval effort (Black (1991)) attempted to compare parsing performance across systems using the Treebank as a basis for comparison. Although Parseval was very useful for comparing parses, it did not enable developers to compare other data structures, such as semantic representations. In addition, in order to accommodate many different parsing formalisms for evaluation, it does not attempt to compare every aspect of the parses. Finally, Treebank data is not always available for domains which need to be tested.

King (1996) discusses the general issues in NLU system evaluations from a software engineering point of view. Flickinger et al. (1987) describe in very general terms a method for evaluation of NLU systems in a single application domain (database query) with a number of different measures, such as accuracy of lexical analysis, parsing, semantics, and correctness of query, based on a large collection of annotated English sentences. Neal et al. (1992) report on an effort to develop a more general evaluation tool for NLU systems. These approaches either focus on application level tests or presuppose the availability of large annotated test collections, which in fact are very expensive to create and maintain. For the purpose of diagnostic evaluation of different versions of the same system, an annotated test corpus is not absolutely necessary because defects and regressions of the system can be discovered from its internal data structures and the differences between them.

2 Matrix Comparison Analysis of NLU Systems

A typical NLU system takes an input of certain form and produces a final as well as a set of intermediate analyses, for instance, parse trees, represented by a variety of data structures ranging from list to graph. These intermediate data can be used as "milestones" to measure the behavior of the underlying system and provide clues for determining the types and scopes of problems.

The intermediate data can be further compared systematically to reveal the behavior changes of a system. In a synchronic comparison, different tests are conducted for a version of the system by changing its parameters, such as the presence or absence of the lexical server, to determine the impact of the module to the system. In a diachronic comparison, tests are conducted for different versions of the system with the same parameters, to gauge the improvements of development effort. In practice, any two tests can be compared to determine the effect of certain factors on the performance of a NLU system. Conceptually, this type of matrix analysis can be represented in a coordinate system (Figure 1) in which a test is represented as a point and a comparison between two tests as an arrowhead line connecting the points. In theory, n-way and second order comparisons are possible, but in practice 2-way first order comparisons are most useful.

ETE is designed for the Unisys natural language engine (NLE), a NL system implemented in Quintus Prolog. NLE can take as input text (sentences or paragraphs) or nbest speech output and produce the following intermediate data structures as Prolog constructs:

- tokens (flat list)
- words (flat list)
- part-of-speech tags (flat list)
- lexical entries (nested attribute-value list)
- parse trees (tree)
- syntactic representation (graph and tree derived from graph)
- semantic representation (graph and tree derived from graph)
- processing time of different stages of analyses

The trees in this case are lines of text where parent-child relationships are implied by line indentations. A graph is expressed as a Prolog list of terms, in which two terms are linked if they have the same (constant) argument in a particular position. In addition to these data structures, NLE also generates a set of diagnostic flags, such as backup parse (failure to achieve a full-span parse) and incomplete semantic analysis.

A special command in NLE can be called to produce the above data in a predefined format on a given corpus.

3 The Engine Test Environment

ETE is comprised of two components: a common relational database that houses the test results and a GUI program that manages and displays the test resources and results. The central database is stored on a file server PC and shared by the analysts through ETE in a Windows NT network environment . ETE communicates with NLE through a TCP/IP socket and the Access database with Visual Basic 5.0. Large and time-consuming batch runs can be carried out on several machines and imported into the database simultaneously. Tests conducted on other platforms, such as Unix, can be transferred into the ETE database and analyzed as well.

The key functions of ETE are described below:

- **Manage test resources**: ETE provides an graphical interface to manage various resources needed for tests, including corpora, NLE versions and parameter settings, and connections to linguistic servers (Norton et al. (1998)). The interface also enforces the constraints on each test. For example, two tests with different corpora cannot be compared.
- **Compare various types of analysis data**. ETE employs different algorithms to compute the difference between different types of data and display the disparate regions graphically The comparison routines are implemented in Prolog except for trees. Lists comparisons are trivial in Prolog. Graph comparison is achieved in two steps. First, all linkage arguments in graph terms are substituted by variables such that links are maintained by unification. Second, set operations are applied to compute the differences. Let $U(G)$ denote the variable substitution of a graph G and $diff(Gx, Gy)$ the set of different terms between Gx and Gy, then $diff(Gx, Gy) = Gx - U(Gy)$ and $diff(Gy, Gx) = Gy - U(Gx)$, where (-) is the Prolog set difference operation. Under this definition, differences in node ordering and link labeling of two graphs are discounted in comparison. For instance, $Gx = [f(a, e1), g(e1, e2)]$, for which $U(Gx) = [f(a, X), g(X, Y)]$, is deemed identical to $Gy = [g(e3, e4), f(a, e3)]$, where ei are linkage arguments. It is easy to see the time complexity of $diff$ is $O(mn)$ for two graphs of size m and n

respectively. Trees are treated as text files and the DOS command *fc* (file comparison) is utilized to compare the differences. Since *fc* has several limits, we are considering replacing it with a tree matching algorithm that is more accurate and sensitive to linguistic structures.

- **Present a hierarchical view of batch analyses.** We base our approach to visual information management upon the notion of "overview, filter, detail-on-demand." For each test, ETE displays a diagnostic report and a table of sentence analyses. The diagnostic report is an overview to direct an analyst's attention to the problem areas which come either from the system's own diagnostics, or from comparisons. ETE is therefore still useful even without every sentence being annotated. The sentence analyses table presents the intermediate data in their logical order and shows on demand the details of each type of data.
- **Enable access to a variety of database query capabilities**. ETE stores all types of intermediate data as strings in the database and provides regular-expression based text search for various data. A unique feature of ETE is *in-report query*, which enables query options on various reports to allow an analyst to quickly zoom in to interesting data based on the diagnostic information. Compared with Tgrep (1992) which works only on Treebank trees, ETE provides a more general and powerful search mechanism for a complex database.
- **Provide graphical and contextual information for annotation**. Annotation is a problem because it still takes a human. ETE offers flexible and easy access to the intermediate data within and across batch runs. For instance, when grading a semantic analysis, the analyst can bring up the lexical and syntactic analyses of the same sentence, or look at the analyses of the sentence in other tests at the same time, all with a few mouse clicks. This context information helps analysts to maintain consistency within and between themselves during annotation.
- **Facilitate access to other resources and applications**. Within ETE, an analyst can execute other applications, such as Microsoft Excel (spreadsheet), and interact with other databases, such as a Problem Database which tracks linguistic problems and an Application Database which records test results for specific applications, to offer an integrated development, test and diagnosis environment for a complex NLU system. The integration of these databases will provide a foundation to evaluate overall system performance. For instance, it would be possible to determine whether more accurate semantic analyses increase the application accuracy.

4 Using the Engine Test Environment

So far ETE has been used in the Unisys NLU group for the following tasks:

- Analyze and quantify system improvements and regressions due to modifications to the system, such as expanding lexicon, grammar and knowledge base. In these diachronic analyses, we use a baseline system and compare subsequent versions against the baseline performance, as well as the previous version. ETE is used to filter out sentences with changed syntactic and semantic analyses so that the analyst can determine the types of the changes in the light of other diagnostic information. A new system can be characterized by percentage of regression and improvement in accuracy as well as time speedup.
- Test the effects of new analysis strategies. For instance, ETE has been used to study if our system can benefit from a part-of-speech tagger. With ETE, we were able quantify the system's accuracy and speed improvements with different tagging options easily and

quickly on test corpora and modify the system and the tagger accordingly.

- Annotate parses and semantic analyses for quality analysis and future reference. We have so far used corrective and grading annotations. In corrective annotation, the analyst corrects a wrong analysis, for example, a part-of-speech tag, with the correct one. In grading annotation, the analyst assigns proper categories to the analyses. In the tests we found that both absolute grading (i.e. a parse is perfect, mediocre or terrible in a test) and relative grading (i.e. a parse is better, same or worse in a comparison) are very useful.

The corpora used in these tests are drawn from various domains of English language, ranging from single sentence questions to e-mail messages. The performance of ETE on batch tests depends largely on NLE, which in turn depends on the size and complexity of a corpus. The tests therefore range from 20 hours to 30 minutes with various corpora in a Pentium Pro PC (200 Mhz, 256 MB memory). A comparison of two batch test results is independent of linguistic analysis and is linear to the size of the corpus. So far we have accumulated 209 MB of test data in the ETE database. The tests show that ETE is capable of dealing with large sets of test items (at an average of 1,000 records per test) in a network environment with fast database access responses. ETE assists analysts to identify problems and debug the system on large data sets. Without ETE, it would be difficult, if not impossible, to perform tasks of this complexity and scale. ETE not only serves as a software tool for large scale tests of a system, but also helps to enforce a sound and systematic development strategy for the NLU system. An issue to be further studied is whether the presence of ETE skews the performance of NLE as they compete for computer resources.

Conclusion

We have described ETE, a software tool for NLU systems and its application in our NL development project. Even though ETE is tied to the current NLU system architecture, its core concepts and techniques, we believe, could be applicable to the testing of other NLU systems. ETE is still undergoing constant improvements, driven both by the underlying NLU system and by users' requests for new features. The experiments with ETE so far show that the tool is of great benefit for advancing Unisys NLU technology

References

ATIS-3 (1991) *Proceedings of the DARPA Speech and Natural Language Workshops*, Morgan Kaufmann

Black E. et al. (1991) *A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars*, Proceedings of Speech and Natural Language Workshop, DARPA, pp. 306 - 311

Flickinger D., Nerbounne J., Sag I., and Wasow T. (1987) *Toward Evaluation of NLP Systems.* Hewlett Packard Laboratories, Palo Alto, California

Harman D.K., Voorhees E.M. (1996) *Proceedings of the Fifth Text Retrieval Conference (TREC-5),* Department of Commerce and NIST

King Margaret (1996) *Evaluating Natural Language Processing Systems.* Communication of ACM, Vol. 39, No. 1, January 1996, pp. 73 - 79

MUC-6 (1995) *Proceedings of the Sixth Message Understanding Conference*, Columbia, Maryland, Morgan Kaufmann

Neal J., Feit, E.L., Funke D.J., and Montgomery C.A. (1992) *An Evaluation Methodology for Natural Language Processing Systems.* Rome Laboratory Technical Report RL-TR-92-308

Norton M.L., Dahl D.A., Li Li, Beals K.P. (1998) *Integration of Large-Scale Linguistic Resources in a Natural Language Understanding System.* to be presented in COLING 98, August 10-14, 1998, Universite de Montreal, Montreal, Quebec, Canada

Tgrep Documentation (1992) http://www.ldc.upenn.edu/ldc/online/treebank/README.long