# GE-CMU: DESCRIPTION OF THE TIPSTER/SHOGUN SYSTEM AS USED FOR MUC-4 [1]

*Paul Jacobs, George Krupka and Lisa Rau*
Artificial Intelligence Laboratory
GE Research and Development
Schenectady, NY 12301 USA
E-mail: rau@crd.ge.com
Phone: (518) 387 - 5059
and
*Todd Kaufmann and Michael Mauldin*
Center for Machine Translation
Carnegie Mellon University

**Abstract**

*The GE-CMU team is developing the* TIPSTER/SHOGUN *system under the government-sponsored TIPSTER program, which aims to advance coverage, accuracy, and portability in text interpretation. The system will soon be tested on Japanese and English news stories in two new domains. MUC-4 served as the first substantial test of the combined system. Because the* SHOGUN *system takes advantage of most of the components of the* GE NLTOOLSET *except for the parser, this paper supplements the* NLTOOLSET *system description by explaining the relationship between the two systems and comparing their performance on the examples from MUC-4.*

## INTRODUCTION

Work on the GE-CMU TIPSTER-SHOGUN system began in the fall of 1991. The first stage of the project involved integrating the resources and algorithms of the existing GE and CMU systems, in order that the combined team could effectively explore new data extraction methods. This integration was barely completed in time for MUC-4: There are still some loose ends, and the combined system is still an infant. The system performed well on MUC-4 because it takes advantage of most of the capabilities of the GE system, and because of the overall system architecture in which different parsers can be easily interchanged. This paper describes this architecture and compares the results of the two parsers on MUC-4.

## SYSTEM OVERVIEW

The TIPSTER-SHOGUN system as configured for MUC-4 uses exactly the same architecture as the GE system, except that it uses the CMU generalized LR parser instead of the TRUMP analyzer. The core lexicon and grammar of the GE system, as well as MUC-specific restrictions and additions to the knowledge base, have been converted to a form that the LR parser can use, and the system interfaces allow the two parsers to be interchanged at the flip of a switch. In fact, it occasionally proved useful in MUC to test the system by toggling between parsers.

[1]This research was sponsored (in part) by the Defense Advanced Research Project Agency (DOD) and other government agencies. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Project Agency or the US Government.

While the integration of the system required some person-months and is still not refined, the effort for MUC-4 was very limited. Most of the time spent on MUC went into testing and executing the final test, and fixing several important system problems that doubled the system's recall between the interim and final test.

In addition to sharing lexicons, grammars, ontologies, and run-time data structures, the two parsers share an evolving control mechanism, which allows for the insertion of new control strategies into both systems. To enable this, the LR parser was augmented to provide dynamic data that look just like TRUMP's, and to receive signals and preferences in the same way that TRUMP does. This helped the parser (which uses an all-paths style algorithm) to manage the complexity of the MUC corpus, and allowed the re-use of the recovery control algorithm. These additions were required to run the LR parser in MUC, because the system was designed for machine translation applications, which have very different requirements (more detailed processing and less ambiguity).

The two parsers, while drastically different algorithmically, are quite similar when used in MUC, with a common lexicon and grammar. The LR parser explores many more parses than TRUMP, but this seems to make very little difference other than to consume more CPU cycles. When the parsers fail, they exhibit different patterns of behavior, but the same recovery strategies work well. The main difference in recall between the two systems seems to come from the lack of certain recovery strategies in the LR parser, leading to a greater number of failures, in addition to a variety of system problems that come from having a less mature system at this point.

The analysis of the example that follows will illustrate some of these similarities and differences.

## ANALYSIS OF TST2-0048

### Overview of Example

Processing of this example proceeds much as in the NLTOOLSET system; the only difference between the two systems is that the LR parser fails in one case where TRUMP does not, and fails to recover in another case. As a result, the GE-CMU system missed 6 total slots that the GE system got (four from one parse and two from the other), for .53 recall at .59 precision and .40 overgeneration.

### Detail of Message Run

The two sentences where the parsers differed in TST2-0048 were S1 and S13. In both cases, the GE systems had heuristics that covered up potential problems, while the combined system, which had not been tested as extensively, slipped up.

On S1, the LR parser failed because of the construct "THE FARABUNDO MARTI NATIONAL LIBERATION FRONT". Names of organizations are treated quite restrictively in the grammar—they are not allowed to have modifiers before or after, because this might lead to misinterpretation. To compensate for this, the pre-processor recognizes names of organizations and usually deletes determiners or brackets out modifiers that appear beforehand. In this case, the pre-processor failed to delete the determiner because it appeared inside of a larger bracketed construct (ACCUSED...OF). TRUMP corrected for this by applying one of a handful of parser-specific recovery strategies, which assumes only as it is about to fail that a name could have been interpreted as a noun. The LR parser, with no such strategy, failed catastrophically in the second clause of the sentence, though it still got the first clause.

In S13, both parsers failed on the phrase "A 15-YEAR-OLD NIECE OF MERINO'S" because of the unusual use of the possessive as a noun (rather than a modifier). TRUMP recovered the whole sentence by attaching the phrase without the apostrophe-s to the verb phrase after; the LR parser failed to execute the same strategy because it could not, for some reason, complete the noun phrase without the apostrophe-s. While this, of course, is a rare case, it is fairly common that the LR parser misses a reduction at the point of failure, and it is thus somewhat less robust on recovery.

### Interpretation of Key Sentences

Because the two parsers produce almost identical output, we will not review the sentences here.

### Comparison of Program Answers with Answer Key

As we have explained, the answers of the GE-CMU system were the same as those of the GE system, except for 6 additional missing slots that came from two parser failures.

## SUMMARY AND CONCLUSION

The experience of testing the TIPSTER-SHOGUN system on MUC-4 was surprisingly rewarding. It provided motivation for pulling together the pieces of the system quickly, and gave the system its first real test, without the extra overhead of developing new code and knowledge for a particular task. We were astonished that each system could benefit from the other to the degree that this experiment proved, and pleasantly surprised at the ability of one parser to emulate another as well as the success with which we shared algorithms and even code.

The rewards come with a price: Maintaining two parsers is more work than maintaining one, and adapting and testing code and data structures is a tedious process.

While the major benefit of the experiment was to prove the degree to which resources could be shared, the ability to compare differences has advantages as well. The GE-CMU system exposed some bugs outside of the parser that for some reason didn't turn up in the GE-only system; likewise, the comparison of the two allowed the new strategies to advance much more rapidly than they would have without a reference point. This suggests a wide range of new things to try, from broadly defining the architecture to accommodate other parsers, to running multiple parsers in parallel as a way of improving accuracy and fault-tolerance. The plug-compatible integration of systems takes task-driven evaluation in MUC to its extreme, allowing a controlled comparison of modules based on their influence on the system as a whole.