THE TIPSTER/SHOGUN PROJECT *

Paul S. Jacobs, George Krupka, Lisa Rau,
Michael L. Mauldin, Teruko Mitamura, Tsuyoshi Kitani,
Ira Sider and Lois Childs

ABSTRACT

This paper presents an overview of the TIPSTER/SHOGUN project, the major results, and the SHOGUN data extraction system. TIPSTER/SHOGUN was a joint effort of GE Corporate Research and Development, Carnegie Mellon University, and Martin Marietta Management and Data Systems (formerly GE Aerospace), part of the ARPA TIPSTER Text program. Two of the main technical thrusts of the project were: (1) the development of a model of finite-state approximation, in which the accuracy of more detailed models of language interpretation could be realized in a simple, efficient framework, and (2) (3) experiments in automated knowledge acquisition, to minimize customization and ease the tuning and extension of the system. Innovations in each of these areas allowed the project to meet its goal of achieving advances in coverage and accuracy while showing consistently good performance across languages and domains.

1. SHOGUN SYSTEM DESCRIPTION

The GE-CMU TIPSTER/SHOGUN system is the result of a two-year research effort, part of the ARPA-sponsored TIPSTER data extraction program. The project's main goals were: (1) to develop algorithms that would advance the state of the art in coverage and accuracy in data extraction, and (2) to demonstrate high performance across languages and domains and to develop methods for easing the adaptation of the system to new languages and domains.

The system as used in MUC-5 (the final TIPSTER benchmark) represents a considerable shift from those used in earlier stages of the program and in previous MUC's. The original SHOGUN design integrated several different approaches by combining different knowledge sources, such as syntax, semantics, phrasal rules, and domain knowledge, at run-time. This allowed the system to achieve a good level of performance very quickly, and made it easy to test different modules and methods; however, it proved very difficult to make all the changes necessary to improve the system, especially across languages, when system knowledge was so distributed at run-time.

As a result, the team adopted a new approach, relying heavily on *finite-state approximation*. This method combines several earlier previous of work, including Pereira's research on grammar approximation [7], some of the original ideas on parser compilation from Tomita [12], and GE's representation of the dynamic lexicon [5, 3]. Like Pereira's model, the system uses a finite-state grammar as a loose version of a context free grammar, under the assumption that the finite state grammar will cover all the inputs that the general grammar would recognize but perhaps be more tolerant. However, the system also includes methods for compiling different knowledge sources into the finite state model, particularly emphasizing lexical knowledge and domain knowledge as reflected in a corpus.

This model, in which knowledge is combined at development time to be used by a finite-state pattern matching engine at run-time, makes it easier to tune the system to a new language or domain without sacrificing the benefit of having general linguistic and conceptual knowledge in the system.

While the GE systems, and more recently, the GE-CMU systems, have done well in all the MUC evaluations, our rate of progress has never been so great as it has been in the period before MUC-5. This is in spite of the fact that the team's diagnostic and debugging efforts had to be divided across languages and domains (handling Japanese, for example, presented a significant overhead in simply being able to follow the rules and analyze the results). We attribute this progress to the current focus on facilitating and automating the knowledge acquisition process, especially on the use of a corpus.

The TIPSTER/SHOGUN system as configured for the 24-month/ MUC-5 benchmark has roughly the same components as earlier versions of the system, but the system now performs linguistic analysis entirely using a finite-state pattern matcher, instead of LR parsing or chart-style parsing, both of which were part of the configuration in MUC-4.

Figure 1 shows the basic components of the SHOGUN system, using our own names for modules, where applicable, along with the labels used in Jerry Hobbs' paper "The Generic Information Extraction System" from the MUC-5 proceedings [11]. The core components of SHOGUN are a subset of the modules that Hobbs describes. However, the system differs from other current extraction systems in the use of the finite-

*For information contact Paul Jacobs, Information Technology Laboratory, GE Research and Development, Schenectady, NY 12301 state analyzer and the way that corpus-based knowledge is integrated into the lexico-syntactic rules.

Because many of the MUC-5 systems now perform much the same type of pre-processing, name recognition, and post processing that SHOGUN has, we will concentrate here on linguistic analysis, including parsing and lexical disambiguation, which were the main research areas of our work on SHOGUN.

About half of the MUC-5 systems still use linguistic analysis driven by "traditional" phrase structure rules, traditional in the sense that there is a clearly separable syntactic component whose knowledge consists mainly of rules for recognizing grammatical constituents based on word categories (like noun, verb) and word order. SHOGUN differs from all these systems in that it no longer has any purely syntactic component, and uses finite state rules in place of phrase structure rules.

The remaining systems divide roughly into those that emphasize pattern matching and those that emphasize fragment parsing. The fragment parsing systems, notably BBN's, work fairly close to the way our MUC-4 system did, taking advantage of partial parses by using a combination of syntactic and domain knowledge to guide the combination of syntactic chunks. The difference between this approach and SHOGUN's current processing is that fragment parsing is still a largely syntax-first method, while pattern matching tends to introduce specialized domain and corpus knowledge by combining this knowledge with syntactic knowledge in the system's declarative representation.

By this coarse characterization, the "pattern matching" group of systems includes, for example, SRI and Unisys as well as GE-CMU. We also consider UMass to be in this category, because their linguistic analysis emphasizes lexical and conceptual knowledge rather than constituent structure.

Among these approaches, we believe the main differentiator is not in the basic processing algorithms but in the way that knowledge ends up getting assigned to various system components. If there is one noteworthy trend among the MUC systems as they have evolved over time, it is that they have become more knowledge-based, especially emphasizing more corpus-based and lexical knowledge as well as automated knowledge acquisition methods. Within the emerging "generic" model, the main difference among systems is thus in the *content* of their knowledge bases. Here, the distinguishing characteristic of SHOGUN is probably the degree to which the system still includes sentence-level knowledge, assigning linguistic and conceptual roles much the way the TRUMP/TRUMPET combination did but using more detailed, lexically-driven knowledge. Many of the sentence-level rules, for example, include groupings like *start a facility* and *organization noun phrase*, which combine traditional syntactic phrases with lexical or domain knowledge.

As systems continue to become still broader in scope and more accurate, it is likely that the way knowledge is acquired will become the main differentiator.

The SHOGUN system is written in Common Lisp and runs on SUN workstations. Typical processing time is about 1000 words per minute. Components of the system have been ported to other languages and hardware and software environments; the core pattern matcher of the system has been re-engineered in "C" and runs on many platforms. In these other environments, throughput tends to be considerably higher.

The rest of this paper will discuss the overall results of SHOGUN on MUC-5 and describe how the system handles some of the system walkthrough examples. The analysis of the examples will highlight some of these characteristics and demonstrate the system's actions in various stages of processing.

2. PROJECT GOALS AND SYSTEM EVOLUTION

We have described the general architecture of SHOGUN as well as the experiments our project did with different control strategies. The end result of the project represented a shift from our original plan, and also somewhat of a surprise. The "new control strategy" we proposed began as relation-driven control [4]—a method of integrating different sources of knowledge at run-time—and ended up as finite-state approximation—a simpler method in which knowledge-sources are combined at development time. Although the major goals of the new control architecture—integrating knowledge sources, enabling corpus-based knowledge acquisition, and introducing domain and corpus knowledge early in processing—were all carried out, the final implemented method differed from the original plan significantly in that it combined knowledge sources mainly at development time rather than at run time.

This shift came from the results of two experiments, along with one important program goal. The experimental results were (1) the effect of different parsing strategies in MUC-4 [10, 8] and (2) the surprising success of Boolean retrieval strategies in text retrieval, as illustrated in the first Text Retrieval Conference (TREC) [2, 6]. Both of these experiments were carried out near the mid-point of our TIPSTER project.

The MUC-4 results combined several important achievements. First, in order to accommodate our objective of using the CMU generalized LR parser in the SHOGUN system, we integrated CMU's parser with GE's semantic interpreter, accomplishing what we believe is the first-ever successful combination of modules of this scope. Although the LR parser recovery strategies were not nearly as well developed, and

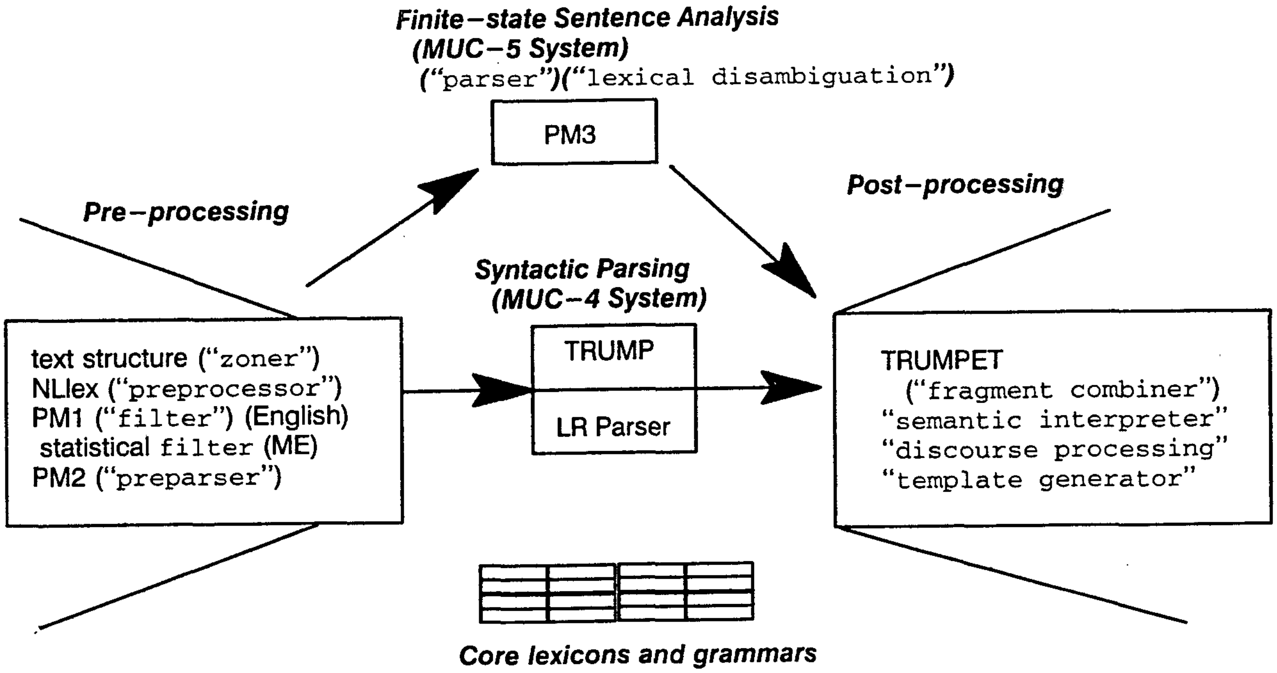

Figure 1: SHOGUN configuration in MUC-5

the LR parser was slower than the GE TRUMP parser, the accuracy of SHOGUN with the LR parser on MUC-4 was reasonably close to that of the GE system on MUC-4. At the same time, we were able to experiment with a variety of different relation-driven control strategies using both parsers, and the net result was the differences in control seemed to have very little effect on the system. We became certain that we would not achieve any significant advance with parser control as the central component of our method. Finally, as part of our MUC-4 system we had significantly improved the capabilities of our finite-state driven pre-processor, so we began to ask what more complex parsing strategies had to offer over finite-state driven strategies, and to take the major step of bypassing traditional parsing altogether.

The TREC results also presented a surprise. GE's TREC-1 experiment, carried on outside of the TIPSTER program, showed that finite-state pattern matching could apply on a broad scale to large volumes of text, but that the power of the finite state engine contributed very little beyond the content of the words and combinations of the words in the patterns. In other words, a Boolean engine recognizing complex combinations of words would do as well in a routing and retrieval task as a mechanism using the same combinations along with linguistic constraints such as part of speech, proximity, and word order.

The other important influence on the selection of the the new control strategy was the program goal of getting Japanese processing up to the level of English processing. At the 12-month point of the project, SHOGUN's Japanese performance was not only behind English, but it was behind some of the other contractors. Our original strategy had assumed the availability of resources such as broad-coverage lexicons and grammars, and while we had allowed for the development of these resources under our project, it looked like the Japanese resources would remain well behind the English resources. In addition, there was a problem with relying on the independent grammars and lexicons in each language: Although the lexicons tied together through a common ontology, there were many common aspects of the task across languages that were very hard to exploit with the emphasis on grammar-driven interpretation—and the leverage across languages could become greater with a new approach.

On the surface, these three influences—the relatively small difference among parsing strategies, the power of Boolean methods in classifying texts, and the need for synergy across languages—may seem to have little in common, but there is a common thread: They all suggest giving knowledge about the *content* of texts greater weight than knowledge about *structure*. For example, in selecting among parsing strategies in our MUC-4 system, the reason that different choices had relatively little impact was that the bulk of the information extracted depended on the labeling of key content elements—names of people, organizations, and event descriptions—at pre-processing time. If a terrorist organization appeared in connection with a terrorist event, it seemed to make little difference what parsing strategy was used, so long as terrorist organization ended up somehow in the final template. In fact, the more restrictive parsing strategies tended to do slightly worse than looser strategies, and the loosest approach—basically using the content elements of each sentence to "guess" the linguistic structure of the sentence—was as good as any.

The success of Booleans in TREC teaches the same lesson. The Booleans identify the key content elements of texts because they recognize words and combinations of words. If those words appear, enforcing structural constraints contributes very little (in general, it improves precision at the expense of recall), even where one might expect that the structure is critical.

The same observation applies to synergy across languages. It is easy to find commonalities between English and Japanese joint venture texts by emphasizing content—for example, *a joint venture description will usually include a joint venture verb, the description of two or more partners, and, optionally, a joint venture company.* Structurally, the texts have little in common. The same applies to the lexicon: words like the Japanese *kaihatsu* (開発) and the corresponding English verb *develop* differ in terms of how they are used syntactically and in general, but in the domain of TIPSTER joint ventures the key discriminations—whether it is land, business, or products that are being developed—are the same for the two verbs.

In retrospect, then, we believe the major lesson of our early experiments, punctuated by the success of finite state approximation in the final benchmark, is that a content focus in processing and acquisition does much better than a structure focus. This does not mean that structural analysis of texts can't make a difference. However, the relative impact of structural analysis is very small because the major challenge, especially in improving coverage, is acquiring and refining content knowledge (i.e. words, combinations of words, relationships between combinations of words and the "core" templates, and so forth).

The second major area of technical progress in SHOGUN was in knowledge acquisition. The ease with which we added knowledge to the system improved coverage, or recall. Even more importantly, it led to portability. It would be hard to attribute SHOGUN's consistently good performance across languages and domains to pre-existing resources—many of which we did not actually use in the final system—or to the time or effort spent working on each configuration— which was relatively small, especially given our shift in mid-stream from one strategy to another. We attribute this to the ease with which knowledge can be acquired, and to the degree of knowledge sharing across languages and domains.

The relationship between representation, i.e., the finite-state patterns in our system, and acquisition, i.e., the method by which new knowledge is added, is critical. Knowledge acquisition methods emphasize adding to knowledge resources that a system uses. If the main knowledge resource of a system is a lexicon, work focuses on lexical acquisition; if the main resource is domain knowledge, work tends to focus on domain knowledge acquisition. In our system, we chose to emphasize the finite-state patterns in part because they help to take advantage of the most critical source of knowledge we have available—the corpus.

Knowledge acquisition was one area in which we experimented heavily in TIPSTER, and found many methods that didn't work as well as some that did. As part of our initial experiments at improving control, we tried a variety of statistical approaches to part-of-speech labeling and grammar tuning, which are generally very popular because they give the appearance of improving parsing in the absence of other knowledge. In MUC-4, these techniques not only didn't help performance: they hurt overall performance. In fact, the only area where we got a benefit from acquiring knowledge for parser tuning was where the statistical method turned up an error in parsing (for example, the word *evening* was sometimes treated as a verb, but it was always a noun in the MUC-4 corpus). These bugs were always easily fixed.

The second acquisition method that didn't help much was learning from answer keys. There were some exceptions—places where the keys did help—but, in most cases, there was simply not enough data in the keys to train a system automatically. The timing was bad, also, because many of the keys were not available until late in the project, so these wasn't much we could do to experiment with them.

Thus, the main acquisition strategy we chose was to rely very heavily on the corpus. Before manually adding anything to the system's knowledge base, we used word frequency information and keyword-in-context lists to select common words and examine how they appeared in the corpus. With the help of a "collated" keyword-in-context browser in both languages, we tried to find common groupings that reflected the context in which words were used and helped to resolve ambiguities. For common slots that required large amounts of knowledge, particularly the product-service slot in the joint venture texts, we relied heavily on statistical methods to find sets of related words and phrases. We believe this accounts for the tremendous differences in coverage between SHOGUN and other systems on these slots.

In addition to helping coverage, the corpus-based acquisition strategy greatly eased portability, particularly across languages. Much of the work done in Japanese (almost all of the work for micro-electronics) was done by non-speakers of Japanese. In most cases, we did each English component first, then used the English as a way of bootstrapping the Japanese. For example, we would take each important "pivot" word in English, try to identify the corresponding "pivot" in Japanese, then use the corpus to identify the relevant contexts in which that word occurred in Japanese. This effectively matched the content of the Japanese patterns to that of the English patterns, making it very easy to acquire knowledge in Japanese. We believe this result is shown most by SHOGUN's performance on some of the "harder" slots in JJV, which was generally near the level of English performance, although most sites did much worse on these slots in Japanese (although we must exclude the time and revenue objects, which few sites attempted in either language).

We have concluded that coverage and portability are synergistic goals in natural language interpretation, and that a representation that emphasizes content rather than structure, in conjunction with an emphasis on corpus-based knowledge, is the secret of success in both.

3. ACCOMPLISHMENTS

The major accomplishment of SHOGUN, in terms of results, was to show significant advances in coverage of data extracted while also demonstrating good performance across languages and domains. However, we believe that the results of the project extend well beyond the benchmarks, including, for example, the proven value of the finite state method, the success of the knowledge acquisition methodology, and a host of experiments that showed what worked and what didn't. In addition to demonstrating high coverage across languages in TIPSTER, we believe that these innovations set the stage for future research.

3.1. Methodology - Correcting Mistakes

Our approach is experimentally-oriented. We do not see any competition between experimentalism and theory; however, there is often a competition between experimentalism and history. Experiments often prove that the way we have been doing things is wrong. This only rarely shows that our theories are wrong (because theories usually are quite diluted by the time they are implemented in broadly useful or functional systems). But, because there is a general reluctance to acknowledge or report negative results, historical methods—those which have never been shown to work but have become accepted as practice—often appear again and again even when experimental evidence points to different approaches.

The advances in TIPSTER, like some of our previous advances, have come from trying many different things, often departing from our intuitions and frequently from "conventional wisdom". However, the results are generally supported by theory. Furthermore, in the context of community efforts, the goal must be to expand the *polytheoretical* aspects and minimize the components of each system that are tied to a particular theory, because those will be the hardest to share and reuse.

SHOGUN has shown the power of focusing on the content of texts rather than the structure of sentences. Sentence structure has been the focus of most linguistic research, including computational linguistics. Within computational linguistics, there has been a great deal of work on grammatical formalisms, most of it aimed at developing notations that make it easy to represent linguistic generalizations, as well as to cover particular constructs. In theory, natural languages fall into a class of languages that are context-sensitive (or worse), thus requiring very powerful (and computationally expensive) methods. However, the phenomena that fall outside the scope of context-free languages are sufficiently unusual that it has become generally accepted that one should handle natural languages using a generally context-free notation, with some additions to deal with the special cases. This has come out with a recent maxim, *language is context-free plus epsilon*, meaning only a small portion of linguistic phenomena require anything more than a context free grammar.

Now, within the class of context free grammars, the choice of notation and parsing strategy is largely a matter of personal preference. Some popular systems have used "textbook" context free parsing algorithms [1], but others have relied on even more restrictive methods like shift-reduce parsing [9] and LR parsing [12]. These methods use the same grammars as the more general context free parsers, but rely on simpler computational strategies for resolving ambiguity.

The difference between finite state approximation, which can recognize the class of regular languages, and the least powerful context-free methods (LR and shift-reduce) is a very small step on the theoretical ladder of languages (the "Chomsky hierarchy"). While the difference between LR languages and general context free languages boils down to the manner in which ambiguity is treated, the difference between regular languages and LR languages, is, in essence, one phenomenon: *recursion*, or *center embedding*. Regular grammars cannot match arbitrary levels of parentheses, for example, or deal with embedded sentences of arbitrary complexity. However, people cannot cope with such constructs, either, at least not without working for a while with pencil and paper. So, in theory, it is quite reasonable to state that *language is regular plus epsilon*. The choice of whether to use a context free or finite state parsing model is simply a matter of convenience.

In speech recognition work, the finite state model has almost always been the convenient choice. As a result, an independent line of research on finite-state approximation [7], stemming from the speech community, showed that context free representations can be formally converted into finite state recognizers, thus guaranteeing that the finite state method would recognize every input that the context free system would. The finite state model may also admit some inputs that the context free model would reject, but the benefit is that the finite state method can use a simple linear-time algorithm, which is also compatible with those used in real time speech recognition. This method of mapping from context free to finite state models tends to be hard to use for large grammars because it creates a very large number of states, but it bears out, theoretically, the close relationship between finite state and context free models of parsing.

Therefore, we see finite state approximation as being experimentally motivated and theoretically sound. The main barrier to the acceptance of the method in computational linguistics is simply that it differs from the way linguists have grown accustomed to viewing language.

Generally speaking, finite state approximation departs from the traditional context free style of representation and parsing, but there is also one claim in the "conventional" wisdom that has been shown to be false. This claim is that because more powerful parsing models are able to handle a broader range of linguistic constructs and enforce more linguistic constraints, they will begin to overtake simpler models as data extraction tasks become broader and more complex. In fact, exactly the opposite has happened. In the four years between MUCK-II and MUC-5, the sites using powerful parsing models have slipped in their relative standings, and the sites that have moved from stricter to looser models (particularly, GE and SRI, but also Paramax and LSI) have improved significantly. The most significant data point here is that SRI improved very significantly between MUC-3 and MUC-4 while abandoning their traditional model in favor of a finite state strategy. This shows that even theoretically-minded groups will adopt a successful experimental strategy when faced with measurable objectives.

We must also note that, in the scope of the current emphasis on community-wide efforts, in which the aim is to create reusable resources and algorithms, the move to simpler models of parsing, as well as the focus on acquisition, are big steps forward. Because finite-state pattern matching is relatively easy to implement (in fact, these methods can run on most any platform and even have available hardware support), research can concentrate on how to develop the right patterns, how to compile knowledge resources, and how to use a corpus to help acquire finite-state rules.

3.2. Overall Coverage and Accuracy

Improved coverage was our strongest area. Recall, the main indicator of coverage, was extremely high relative to other systems in all four configurations. In addition, recall advanced 37% on average between the TIPSTER 18-month evaluation and the 24-month evaluation and was 10% higher in the TIPSTER final test than in SHOGUN's results on MUC-4 (a much easier test of coverage).

Accuracy, as measured by precision, was somewhat lower than some other systems. We attribute this mostly to the vastly larger amount of data that SHOGUN produced on much harder slots, at least in English joint ventures. In micro-electronics, we cannot characterize the results across objects and slots, but the differences between systems, in general, were much smaller in micro-electronics. SHOGUN's accuracy, in general, was higher than systems with comparable coverage on the components of the task where there were other systems with comparable coverage.

The main contributor to these advances, as discussed above, was the combination of finite-state level interpretation with corpus-driven knowledge acquisition methods. These methods effectively improved coverage, particularly in portions of the task requiring larger amounts of knowledge, without a sacrifice in accuracy.

What didn't work in advancing coverage and accuracy, generally speaking, were (1) methods of "fine tuning" system components, such as parser tuning and tagging, and (2) most automated acquisition strategies, especially those based on training on answer keys. The acquisition strategies that *did* work were those that relied mainly on raw corpus data or on manual intervention or bootstrapping.

We believe that the focus of future work in extending coverage and accuracy must be in acquiring knowledge from corpora. Ideally, a good portion of this knowledge, and certainly a good portion of the acquisition methodology, can be used across languages and domains. At the very least, corpus-based methods help to adapt and extend systems, but we believe we have only begun to explore the means by which the corpus can contribute to application development.

3.3. Portability to New Languages

We have emphasized SHOGUN's consistently high performance across languages. We can't argue that SHOGUN is the most consistent system across languages because one system, SRI's, had identical error rates in Japanese and English JV, and another system, BBN's, had error rates within a 6-point range (from 66 to 72) across the four configurations. SHOGUN's error rates were within an 11-point range (from 54 to 65), and SHOGUN's error rates were consistently lower in Japanese than in English. However, we believe that the Japanese configurations were somewhat easier than the English; thus SHOGUN's performance was consistently good.

In order to equalize resources and ease general portability across languages, CMU built built a 17,854 entry lexicon (with 15,984 different words) of Japanese words from the Tipster Corpus, supplemented by other sources to allow the lexicon to be used for domains other than Joint Ventures and Micro-electronics.

Sources for lists of Japanese words came from the BBN company name list, NTT Data's MAJESTY program run over the Japanese Joint Venture and Japanese Micro-electronics corpora, plus previous Japanese dictionaries from the CMT-SEMSYN project.

The next figure shows two Japanese lexicon entries, a noun and a verbal nominal. Each entry is headed by the Kana or Kanji as segmented by the MAJESTY segmenter, and then all senses with that particular Kana/Kanji string are stored as a list under the :SENSES field.

```
( データバンク
:POS n
:TOKENS ()
:G-DERIVS ()
:SENSES
(( de-tabanku
:EXAMPLES (中国統計データバンク)
:TYPE *primary*
:PAR (c-information)
:SYNONYMS (data-bank)
:NOTE (1 occurrences
:nttd-kana ("でーたばんく")
:jv-dom :me-dom)
:S-DERIVS ()
)))
```

```
( 物色
:POS nsa
:TOKENS ()
:G-DERIVS ()
:SENSES
(( busshoku
:EXAMPLES (新しい提携先を物色し、判断する)
:TYPE *primary*
:PAR (c-searching)
:SYNONYMS (look-for search-for)
:NOTE (1 occurrences
:nttd-kana ("ぶっしょく")
:jv-dom)
:S-DERIVS ()
)))
```

The Japanese lexicon development effort shows that it is reasonable and useful to develop core lexicons in new languages using a common framework and ontology. We believe that the Japanese lexicon is a useful resource for the community. However, the core lexicon, as we have explained, provides only a small part of the coverage necessary to do the data extraction task.

3.4. Portability to New Domains

The amount of effort required to adapt SHOGUN to new domains was quite small. Figure 2 shows about how much time went into each configuration, who spent the time, and what kinds of things were done.

Because development in the joint venture domain coincided with a large number of experiments, it was very hard to measure the degree of effort dedicated to joint ventures in particular. The micro-electronics task presented a more accurate test of porting to a new domain (as well as to a new language), although we believe that the amount of effort to achieve creditable performance in micro-electronics is much less than in the joint venture domain.

The Japanese micro-electronics task is our best data point in terms of ease of portability of the system. The main knowledge base development effort in micro-electronics was carried out by a single programmer, who did not know Japanese (but could read Chinese characters) and was a relative novice in the system (TIPSTER was his first experience with any AI project). He got a low level of help from two sources—native Japanese speakers, who helped somewhat with the vocabulary and to identify lists, phrases, and errors; and the main developer of the Japanese joint venture knowledge base (who also does not know Japanese). He got a somewhat higher level of assistance from the developer of the *English* micro-electronics knowledge base.

In some sense, this may show that it is easier to port the system to a new language than to a new domain, because the Japanese micro-electronics system required less effort than the English and clearly shared more with the English micro-electronics effort than from the Japanese joint venture effort. But it also shows the ease of porting in general, as both micro-electronics efforts for reasonably small.

Many factors influence the degree of portability of a system, and we should point out that portability from one domain to another is extremely sensitive to the degree of overlap between the two domains, as well as the difficulty of the target domain and the level of performance to be obtained. Portability across languages actually seems less variable, in that the apparent differences between English and Japanese seemed to have relatively less impact on the development effort than the differences among domains.

The following are some of the important factors influencing portability across domains:

- Sharing of knowledge base components based on task or corpus similarity, including, for example, name recognition, headline and dateline processing, special verbs (like manufacture, develop).
- Re-use of knowledge base components from a "library" of generic content elements that apply across a range of domains—for example, dates, locations, numbers, and monetary units.
- Tools for reducing the amount of manual effort in porting (including word in context access, statistical analysis,

Domain/Language	*Effort/Skill Level*	*Other Notes*
English joint ventures	*1 person−year, system developers, native English speakers*	*Some effort not reflected in results* *Difficult to measure because of many experiments*
Japanese joint ventures	*1.5 person−years, mostly Japanese college students with non−native developers*	*Least efficient, but most interesting effort* *Best overall results*
English micro−electronics	*3 person−months, system developers, native speakers, no knowledge of ME*	*Lowest overall results (but explained by sample variation)*
Japanese micro−electronics	*2 person−months, non−developers, non−native speakers (with some help from natives, developers)*	*Last configuration done, least work, good results (but not refined)*

Figure 2: Level of effort required to port SHOGUN

word frequency information) and for automatically augmenting the knowledge base.

- Generality of linguistic components, for example, lexicon coverage, morphology, Japanese segmentation, and ability of components to produce useful domain-independent information.

All of these factors contributed to SHOGUN's portability. In terms of what worked, we were especially pleased with the ability to re-use knowledge base components across tasks, and with the use of tools for reducing the manual effort in porting. The automatic acquisition effort, as we have discussed, was somewhat less successful. The use of "libraries" of recognizers that help to configure data extraction systems is a potentially very helpful resource, one which we have begun to exploit in other projects as well.

In terms of what didn't work, we were surprised and disappointed with the degree to which general linguistic components contributed (or didn't contribute) to portability. In previous work, we relied more heavily on lexical and grammar resources, and successfully ported to new domains (although not as complex as TIPSTER) with roughly the same level of effort. In this more grammar-oriented approach, we would generally start by making sure the "core" parser and semantic interpreter could cover most of the text in each new domain, then attach domain knowledge to the output of the semantic interpreter. Although this did not represent a very large effort, there were two major problems: (1) the first step, checking the linguistic components and tuning them to a new domain, usually required substantial expertise and involvement by system developers, and (2) there seemed to be very little hope of reducing the level of effort required to tie domain knowledge to semantic representations.

We are thus very optimistic about the prospects for improving corpus analysis and acquisition tools, and about the use of libraries to help to configure extraction systems to new domains. We also believe that considering available resources during the task design in an application can greatly reduce the effort in porting a system; sharing a "core" structure that applies to a related domain can make porting much easier. On the other hand, we are pessimistic about the prospects for reducing customization time by improving "core" linguistic resources; in our view, we have already tapped lexicons and grammars for much of what they have to offer in data extraction, and the best path we see in this respect is to use these resources to improve the results of automated corpus analysis.

4. EVALUATION SUMMARY

SHOGUN ran the official MUC-5 (TIPSTER 24-month) benchmark in all four configurations; in addition, the team ran an "optional" official system called TEXTRACT, developed mostly independently of SHOGUN, on the two Japanese configurations. After the benchmarks, the team did a number of experiments comparing the results of the two systems, and showing how these results could be combined to produce even better performance.

4.1. SHOGUN's Results

Figure 3 is a summary of SHOGUN's performance on all the official metrics. We put error rate first and F-measure last in this table because these are the only ones that can be used for overall system comparison (the goal being low error rate and high F-measure).

The overall results here are better, on average, than SHOGUN'S scores on the MUC-4 benchmark. While it is very difficult to compare results across domains across languages, it is clear that this shows substantial progress, as the MUC-5 tasks are certainly much harder and more detailed than MUC-4. In addition, the average improvement between the TIPSTER 18-month benchmark and the current point was over 20%, and there is certainly more room for further improvement. Thus, we are confident that our current methods and algorithms support continued progress toward high accuracy.

While it seems that there is substantial variation among the scores on the different language-domain pairs, this variation is reasonable given the differences among the task and the variations on the test samples. The EME result is worse than the others, but the EME MUC-5 test set seemed to be a very difficult one for our system. In fact, the system on a blind test using the same configuration scored 9 error rate points better in EME than on the test reported above. We are not sure what accounts for this variability in EME, which is much greater than on the other domain-language pairs.

With respect to achieving human performance, it is not clear where good human perform falls on these scales, but we are close. At the TIPSTER 12-month test, a study of trained human analysts placed individual analysts between 70 and 80 in F-measure. However, this test used a somewhat more generous scoring algorithm than the current one (there have been a number of important changes to the scoring since the 12-month point), and did not separate the analysts work from the preparation of the "ideal" answers—it is important in a blind test that the human subject have no impact on the answer key, because there are many texts that involve fine-grained interpretation.

The results on Japanese are, on average, somewhat higher than the English results. This is consistent with all our tests. We attribute this to the fact that the Japanese tests are considerably easier than the English (a factor that is somewhat difficult to weight, given that none of our system developers know Japanese). Some of the influences that make the Japanese easier are greater homogeneity in the text sources (for example, EME includes very different sources from EJV, while JJV and JME are quite consistent in style), shorter stories with fewer distinct events in Japanese, far fewer new joint venture companies in Japanese, and an emphasis in Japanese on research and sales rather than production (production activities are more difficult to assign to codes in the template design).

4.2. TEXTRACT and Combining Systems

In addition to the SHOGUN system, the GE-CMU team ran the Japanese benchmarks only using a system called TEXTRACT, which was developed in parallel to SHOGUN by Tsuyoshi Kitani, a visiting researcher at CMU from NTT Data. TEXTRACT, like SHOGUN, emphasizes lexically-driven pattern matching, and the two systems share a Japanese tagging/segmentation program from NTT Data, called MAJESTY. While there is little else that is directly shared between the two system's, additions to TEXTRACT's knowledge base were incrementally adapted, in functionality, to SHOGUN's knowledge base in JJV, thus it it not surprising that the systems had similar performance on this set. TEXTRACT generally had a better performance on company name recognition than SHOGUN, and a somewhat more effective method of splitting events. SHOGUN had better coverage of industry types and products (based, we think, on the heavy use of statistically-based training), and had higher recall (but lower precision) in JME.

For the Japanese Micro-electronics domain, the SHOGUN system scored the highest recall, while the TEXTRACT system scored the highest precision. The F-measure and error scores were almost exactly the same. We developed a statistical technique to combine these systems in a way to improve the F-measure, and as a by-product we determined the theoretical limits of combining the output of the two systems.

The combining algorithm works as follows: both SHOGUN and TEXTRACT are run on an input text, and the output templates are given as input to the combiner. The following methods were examined:

	Error	UND	OVG	SUB	Min-err	Max-err	Text	Rec	Pre	F-meas
EJV	61	30	39	19	0.8784	0.9026	96/92	57	49	52.8
JJV	54	36	27	12	0.6624	0.6794	99/98	57	64	60.1
EME	65	37	41	19	0.8354	0.8724	95/81	50	48	49.2
JME	58	30	38	14	0.7756	0.8152	97/86	60	53	56.3

Figure 3: SHOGUN Scores for MUC-5

SHOGUN this row just shows the scores for the SHOGUN system.

TEXTRACT this row shows the scores for the TEXTRACT system.

Theoretical max this row shows the scores for a system which chooses perfectly whether SHOGUN or TEXTRACT has the better answer for a particular text.

Entity weight D=T this row shows the results of using total entity weight to select the output template, using TEXTRACT output in case of ties.

Entity weight D=S same as above, but uses SHOGUN output to break ties.

Most names D=S this method chooses the output template with the most entity names.

Avg Entity weight D=T similar to entity weight, but the average is used instead of the total weight.

SHO + TEX this method uses SHOGUN's output unless it is empty, in which case TEXTRACT's output is used.

TEX + SHO this method uses TEXTRACT's output unless it is empty, it which case SHOGUN's output is used.

Avg Entity weight D=S average entity weight with SHOGUN output used in case of a tie.

Single capability D=T this method chooses the output with the number of capabilities closest to one, and chooses TEXTRACT's output in case of a tie.

Figure 5 gives the numeric values for the various combining methods, and Figure 6 shows the recall-precision performance of each method graphically.

Note that the best performing method was the total entity weight, which used statistics from the development corpus for the entity-name slot to determine which output template had more commonly found company names. Intuitively, if the output template had more companies that were associated with correct keys from the development corpus, that template is more likely to be correct. Note also that no knowledge-free combining method gave a better F-measure than either of the two systems alone.

4.3. Analysis of Benchmark Results

It is hard to consider the TIPSTER program without seriously analyzing the benchmark results. The extraction part of TIPSTER included four official benchmarks during a period of about 15 months, covering the MUC-4 evaluation and the TIPSTER 12-month, 18-month, and 24-month (MUC-5) tests. In addition to funding of individual contractors' efforts, the government devoted significant resources to the preparation of materials for these evaluations.

Benchmarks, above all else, are effective as a way of cultivating and comparing technologies. The evaluations are meant to show coverage and accuracy; in terms of system performance, these can translate into different characteristics. For example, coverage includes robustness (e.g. ability to handle misspellings and other errors in the input, unknown or unusual words, varieties of news style, etc.), vocabulary, grammatical coverage, and the extent of domain knowledge. Accuracy includes the ability to distinguish different senses of words, preciseness of interpretation, adherence to template fill rules, elimination of program bugs, and getting the overall construal of a text correct (i.e., putting the right information in the right place).

SHOGUN, on average, extracted 37% more information correctly (37% higher recall) than the second-best systems in each configuration on the final benchmarks (which were, of course, different systems in different configurations). On average, SHOGUN's precision in the information extracted was 13% lower than the next best system. Comparing with individual systems from other groups that did all four configurations, SHOGUN had 49% higher recall (with 5% lower precision) than one system, and 71% higher recall (with 8% lower precision) than the other, on average.

What is the source of these very significant differences in coverage? While there are many places where systems differ, the

	Error	UND	OVG	SUB	Min-err	Max-err	Text	Rec	Pre	F-meas
JJV	49.99	32	23	12	0.5877	0.6028	99/99	60	68	63.84
JME	58.64	43	28	12	0.6728	0.7072	96/85	51	63	56.35

Figure 4: Official TEXTRACT Scores for MUC-5

most obvious differences are on the more difficult portions of the task. In fact, in one configuration there was one class of information, the recognition of entities in micro-electronics, where SHOGUN did worse than the next best system. We believe that this is because the other system was doing very good name recognition, and had tested the name recognition component more carefully with the micro-electronics corpus. However, name recognition and top-level object recognition are certainly the easiest, least error-prone portions of the TIPSTER task.

In the harder extraction sub-tasks, SHOGUN had much higher recall, and generally much lower precision. For example, in EJV, SHOGUN correctly extracted 3 times as much industry information, 4.5 times as much facility information, and 3.3 times as much activity information as the next best system, while making many more errors than the next best system. It is very hard to compare systems when their performance is averaged across sub-tasks with very different degrees of difficulty, but when we look at performance on individual objects, slots, and messages, it would appear that the difference between SHOGUN and other systems stems largely from SHOGUN's higher coverage on harder parts of the task. As a result of doing more of these harder components, SHOGUN has lower precision and higher overgeneration than some other systems, but we do not see this as a real tradeoff, because the lower accuracy usually comes from extracting difficult information that other systems did not extract at all.

We attribute these substantive differences between SHOGUN and other systems to the combination of the finite-state approximation method with the corpus-based knowledge acquisition strategies. The finite-state method can be tuned for higher precision or higher recall; SHOGUN shows higher recall. The difference in recall was most pronounced on objects like the industry object, where word content far outweighs sentence structure. On these sorts of sub-tasks, the corpus-based knowledge acquisition strategy helped SHOGUN to obtain much better coverage of the task.

This analysis has tried to identify the salient differences among systems and approaches as shown in the MUC-5/TIPSTER 24-month benchmark. As we have discussed, SHOGUN had very good performance across languages and domains, and had a consistently lower error rate, mostly due to much higher recall than other systems. In addition, SHOGUN's advantage seemed to come more from "harder" slots in general, although there were only a few slots in any configuration where other systems did better, and these were concentrated in English micro-electronics. It is a safe conclusion that our project successfully produced a good, high coverage, portable system.

5. SUMMARY AND CONCLUSIONS

The TIPSTER/SHOGUN project advanced the state of the art in data extraction by developing a simplified model of text processing emphasizing finite state approximation, and by defining new methods for corpus-based knowledge acquisition. While demonstrating high coverage across languages in TIPSTER, these innovations open the way for future advances by focusing on corpus-based knowledge and content processing of texts.

6. REFERENCES

[1] Jay Earley. An efficient context-free parsing algorithm. *Communications of the Association for Computing Machinery*, 13(2):94–102, February 1970.

[2] Donna Harman, editor. *Proceedings of the Text Retrieval Conference (TREC)*. Morgan Kaufmann Publishers, San Mateo, CA, November 1992.

[3] P. S. Jacobs and L. F. Rau. Innovations in text interpretation. *Artificial Intelligence*, 63:143–191, 1993.

[4] Paul S. Jacobs. Parsing run amok: Relation-driven control for text analysis. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, 1992.

[5] Susan McRoy. Using multiple knowledge sources for word sense discrimination. *Computational Linguistics*, 18(1), March 1992.

[6] George R. Krupka Paul S. Jacobs and Lisa F. Rau. A Boolean approximation method for query construction and topic assignment in TREC. In *Second Annual Symposium on Document Analysis and Information Retrieval*, May 1993.

Method	Recall	Precision	F-Measure
SHOGUN	60.0306	53.0254	56.3110
TEXTRACT	50.6498	63.4988	56.3511
Theoretical max	61.0330	63.4371	62.2118
Entity weight D=T	56.0208	58.9768	57.4608
Entity weight D=S	60.1467	53.1031	56.4058
Most names D=S	61.7665	51.5824	56.2170
Avg Entity weight D=T	53.8203	58.7784	56.1902
SHO + TEX	60.7034	51.4782	55.7115
TEX + SHO	52.3476	58.6007	55.2979
Avg Entity weight D=S	55.2294	55.0946	55.1619
Single capability D=T	53.0257	57.0724	54.9747

Figure 5: Combining Two MUC-5 Systems: Table

[7] Fernando Pereira. Finite-state approximations of grammars. In *DARPA Speech and Natural Language Workshop*, pages 20–25, Hidden Valley, PA, 1990.

[8] Lisa F. Rau, George R. Krupka, and Paul S. Jacobs. GE NLToolset: MUC-4 test results and analysis. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, San Mateo, CA, June 1992. Morgan Kaufmann Publishers.

[9] S. Shieber. Sentence disambiguation by a shift-reduce parsing technique. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, 1983.

[10] Beth Sundheim, editor. *Proceedings of the Fourth Message Understanding Conference (MUC-4)*. Morgan Kaufmann Publishers, San Mateo, Ca, June 1992.

[11] Beth Sundheim, editor. *Proceedings of the Fifth Message Understanding Conference (MUC-5)*. Morgan Kaufmann Publishers, San Mateo, Ca, August 1993.

[12] M. Tomita. *Efficient Parsing for Natural Language*. Kluwer Academic Publishers, Hingham, Massachusetts, 1986.

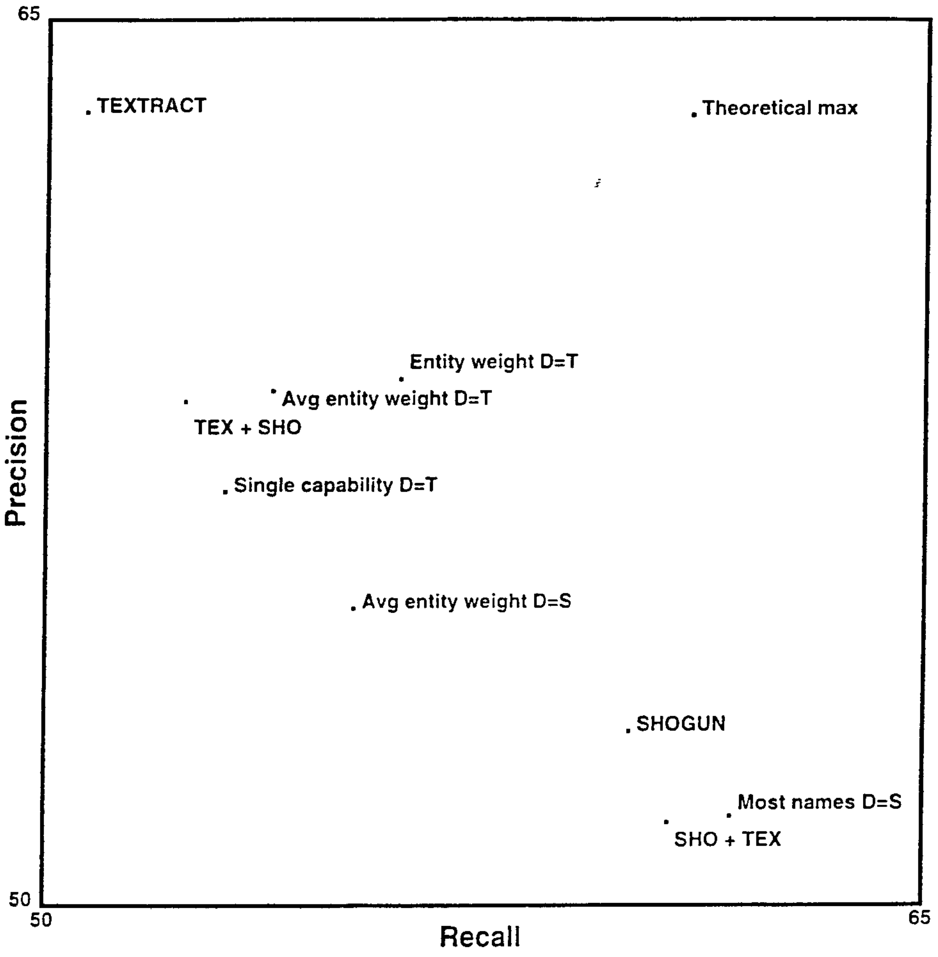

Figure 6: Combining Two MUC-5 Systems: Graph