GE-CMU: DESCRIPTION OF THE SHOGUN SYSTEM USED FOR MUC-5 [1]

Paul S. Jacobs, George Krupka, and Lisa Rau
Information Technology Laboratory
GE Research and Development
Schenectady, NY 12301 USA

Michael L. Mauldin, Teruko Mitamura, and Tsuyoshi Kitani
Center for Machine Translation
Carnegie Mellon University
Pittsburgh, PA 15213 USA

Ira Sider and Lois Childs
Management Data Systems
Martin Marietta
Philadelphia, PA 19101 USA

Abstract

This paper describes the GE-CMU TIPSTER/SHOGUN system as configured for the TIPSTER 24-month (MUC-5) benchmark, and gives details of the system's performance on the selected Japanese and English texts. The SHOGUN system is a distillation of some of the key ideas that emerged from previous benchmarks and experiments, emphasizing a simple architecture in which the focus is on detailed corpus-based knowledge. This design allowed the project to meet its goal of achieving advances in coverage and accuracy while showing consistently good performance across languages and domains.

INTRODUCTION

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 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 [4], some of the original ideas on parser compilation from Tomita [5], and GE's representation of the dynamic lexicon [3, 1]. Like Pereira's model, the system uses a finite-state grammar as a loose version of a context free

[1] This research was sponsored (in part) by the 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 Advanced Research Project Agency or the US Government.

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.

This paper will give a very brief overview of the configuration of the system, followed by the analysis of the examples, and some conclusions about the results.

SYSTEM OVERVIEW

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". 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-state analyzer and the way that corpus-based knowledge is integrated into the lexico-syntactic rules.

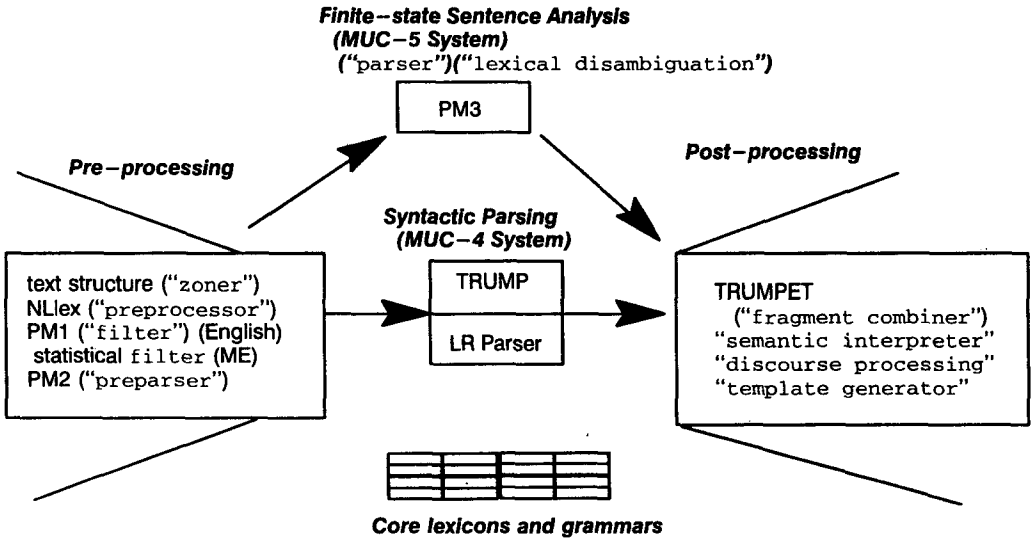

Figure 1: SHOGUN configuration in MUC-5

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 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.

OVERALL RESULTS

The SHOGUN system did very well on MUC-5. The team's specific goals were to achieve results on the MUC-5/TIPSTER tasks that were above the level of the simpler MUC-4 task, to attain comparable performance across languages and domains, and to reduce customization time as much as possible. In addition, the aim was to produce near-human accuracy at a throughput orders of magnitude faster than human beings. These goals seemed rather ambitious, but SHOGUN reached all of them.

The following 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).

	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 2: SHOGUN Scores for MUC-5

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 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).

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.

Figure 3 shows the results of both systems on the recall/precision scale on the various MUC-5 sets.

ANALYSIS OF WALKTHROUGH MESSAGES

Overview of Examples

The examples are in many ways typical of the TIPSTER-SHOGUN system. These are relatively easy messages, but the problems the system encountered are illustrative. In the English message, the system made a few minor mistakes, some of which may even have been matters of fine-grained interpretation, and had an error rate of 15 for EJV0592. This is much better than the average message; on the whole, the EJV performance is pulled down by "tangled tie-up" messages in which the system has a great deal of difficulty determining who is doing what with whom.

JJV0002 was much harder, because it requires information to be split across two tie-ups. The system correctly determined that there were two tie-ups (which it did not do when it ran this message at the 12-month point), but it failed to recognize "Toukyou kaijou" as an alias for "Toukyou kaijou kasai hoken", and as a result ended up getting a whole bunch of aliases and entity pointers wrong. In addition, SHOGUN made the very typical mistake of almost getting the product service information but losing most of the points, anyway. In this case, the Japanese text says that the tie-up will be selling a new product called "hyu-man".

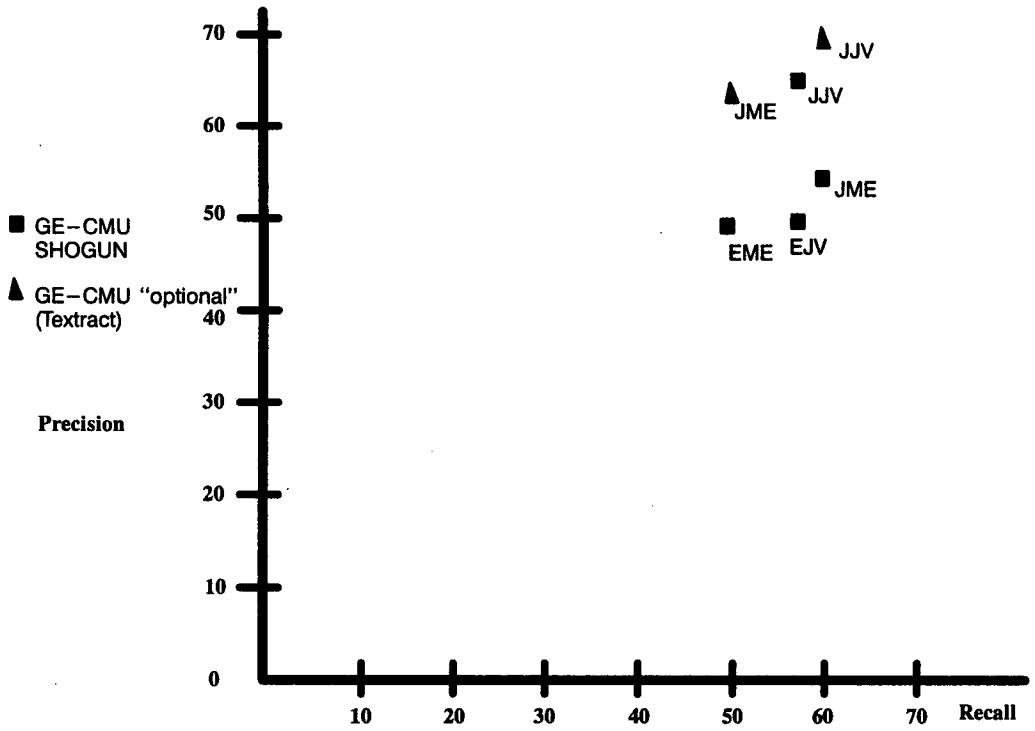

Figure 3: GE-CMU Results for MUC-5/TIPSTER 24-month benchmark

SHOGUN correctly spots this and assumes that whatever "hyu-man" is will be wholesale sales with code 50. The analyst infers from the context that "hyu-man" is an insurance product, so the actual industry type is "finance" rather than "sales". Finally, the answer key contains an error in the string fill, so SHOGUN gets scored completely wrong on this object.

We emphasize these minor mistakes because it helps to show, for one thing, how hard it is to get extremely high accuracy, and, for another, the relative effects of easy and hard objects. SHOGUN was, by far, the most accurate system in determining industry information, probably because our efforts on automated knowledge acquisition used this object as a test case for both English and Japanese. However, the net effect of the industry object in SHOGUN was a reduction in error of .2 in English and 1.2 in Japanese over what the system would have produced by leaving the product service slot blank. This is because potentially spurious information on hard objects and slots dilutes the good scores produced on the easier objects and slots. Hence it is very difficult to show improvement by getting more information; the easiest improvements are to get higher and higher performance on the "critical" slots and objects.

In addition, the system made many technical errors with the location and alias slots, some of which are illustrated here. Often these were due to bugs, but there are many other problems. The location slot(s) proved much more difficult than expected, because many forms of subtle inferences often affect location information, such as inferring that one site subsumes another or inferring location by process of elimination (especially in Japanese).

We will now show, very briefly, the results of each stage in processing of SHOGUN on the EJV and JJV examples.

Pre-processing

Pre-processing identifies names, dates, locations, and other special phrases, and handles certain morphological rules in Japanese. For example, the following gives some of the results of pre-processing on one sentence from each example:

```
EJV0592 Sentence 0:
 [CNAME{1}: BRIDGESTONE SPORTS CO.  ]  SAID FRIDAY IT HAS SET UP A  JOINT VENTURE
```

```
 [IGNORE{41}: IN  TAIWAN  ] WITH A LOCAL CONCERN AND A  JAPANESE  TRADING HOUSE
 TO PRODUCE GOLF CLUBS TO BE SHIPPED TO  JAPAN.

JJV0002 Sentence 0:
 [CNAME{24}: 東京海上火災保険 ]  は今月から大和証券と [MORPH{8}: 提携して ]
 保険と年金の機能を兼ね [MORPH{4}: 備えた ]  新商品「ヒューマン」を
 [MORPH{5}: 売り出した ]  。
```

Where a company name is marked in pre-processing, this means that the name is "learned" rather than recognized as a known name. In JJV0002, Daiwashouken (大和証券) is a known name, so it is not marked above.

Linguistic analysis

Linguistic analysis uses the same pattern matcher and same knowledge base notation as pre-processing, but relies on a mixture of syntactic and lexical information to perform sentence-level interpretation. For example, the following is one rule for marking verb phrases with activity information in English:

```
44::
;; JV ACTIVITY-VP
;; ACTIVITY
{< ?START-TIME=*date* * >}
[ $jventure
  ?ENTITY=(and org-name (not *partner* *venture*))
  < ?VENTURE=*venture* {< (member *apostrophe-s* *apostrophe*) rights >} >
  < (and *venture-org-np* (not $ventureterm)) {$loc} >
  it
  $facilityphr ]
{$np-postmod}
{which}
{$helperphr}
$verb-premod*
{to}
[ ?ACTIVITY-TEXT=
  < ?TIE-UP-ACTIVITY=$actverb
    < *comma* ?TIE-UP-ACTIVITY=$actverb >*
    {< {*comma*} and ?TIE-UP-ACTIVITY=$actverb>}
    $ps-text-list >
  < ?ACTIVITY-TEXT=< ?TIE-UP-ACTIVITY=$actverb $ps-text-list >
    ?ACTIVITY-TEXT=< *comma* ?TIE-UP-ACTIVITY=$actverb $ps-text-list >*
    {*comma*} and
    ?ACTIVITY-TEXT=< ?TIE-UP-ACTIVITY=$actverb $ps-text-list > > ]
{< (not *date*)* $loc >}
{< (not *date*)* ?START-TIME=*date* {$loc} >}
<=> (mark-jv-activator c-joint-venture-template) ;
```

In linguistic analysis, the pattern matcher annotates the text, much like it does during pre-processing, but these annotations can be very close to the roles that portions of text will play in the template. For example, where pre-processing finds company names and organization descriptions, sentence analysis will often find partners and ventures.

The following are examples of this analysis from the walkthrough

```
EJV0592 Sentence 0:
```

```
[C-JOINT-VENTURE-TEMPLATE{45,44,16,2,0} ?CONJ=<
?ENTITY=?PARTNER=?HEAD=BRIDGESTONE SPORTS CO. SAID FRIDAY IT ?HEAD=HAS
?HEAD=SET UP [C-JOINT-VENTURE-TEMPLATE{36,13}?HEAD=A ?HEAD=JOINT
VENTURE IN ?LOCATION=TAIWAN WITH A LOCAL ?PARTNER=CONCERN AND A
JAPANESE ?ACTIVITY-TEXT=< ?TIE-UP-ACTIVITY=TRADING
?TIE-UP-PRODSERV=?PS-TEXT=?PARTNER=HOUSE >=?ACTIVITY-TEXT >=?CONJ
{45}] ?ACTIVITY-TEXT=< TO ?ACTIVITY-TEXT=<
?TIE-UP-ACTIVITY=?HEAD=PRODUCE ?PS-TEXT=< GOLF
?TIE-UP-PRODSERV=?TIE-UP-ACTIVITY=CLUBS >=?PS-TEXT >=?ACTIVITY-TEXT
{44,36,16,13,2,0}]TO BE SHIPPED TO JAPAN.

JJV0002 Sentence 1 :

また [C-JOINT-VENTURE-TEMPLATE{12,0}
?PARTNER=?HEAD= 日新火災海上保険、 ?PARTNER= 同和火災海上保険の両社は
?PARTNER= 山一証券と結んで [C-JOINT-VENTURE-TEMPLATE{9} ?PS-TEXT=<
?ACTIVITY-TEXT=< ?HEAD= 年金型 商品 >=?PS-TEXT を近く、
?TIE-UP-ACTIVITY= 発売する >=?ACTIVITY-TEXT {12,9}] 予定で、損害保険業界
の [C-JOINT-VENTURE-TEMPLATE{1} ?PARTNER=?HEAD= 証券会社との提携
{1,0}][C-JOINT-VENTURE-TEMPLATE{8}?HEAD= 商品づくりは ?PS-TEXT= 一段
{8}] と広がって。
```

Each set of annotations from sentence-level analysis goes through semantic interpretation, top-down analysis (using TRUMPET), and discourse processing, just as full parses and fragment parses were used in TRUMP and the LR parser. The input to TRUMPET now, however, is a set of annotations instead of full or partial syntactic trees.

```
Calling Trumpet with SENSE Interpretation:
(C-JOINT-VENTURE-TEMPLATE (R-TIE-UP-ACTIVITY (PRODUCE))
 (R-LOCATION (TAIWAN (R-NAME TAIWAN)))
 (R-PARTNER
  (CNAME_BRIDGESTONE-SPORTS-CO1 (R-NAME BRIDGESTONE-SPORTS-CO)
   (R-PART (C-ENTITY))))
 (R-PARTNER (CONCERN)) (R-PARTNER (HOUSE)))


Calling Trumpet with SENSE Interpretation:
(C-CAP-TEMPLATE
 (R-CAP
  (C-MONEY (R-QUANTITY (C-NUMBER (R-VALUE |20000000|))) (R-UNIT (DOLLAR))))
 (R-OWNED
  (CNAME_BRIDGESTONE-SPORTS-TAIWAN-CO1 (R-NAME BRIDGESTONE-SPORTS-TAIWAN-CO)
   (R-PART (C-ENTITY)))))

;; Top-down processing
Linking (special) C-CAP-TEMPLATE as filler for R-OWNERSHIP of C-JOINT-VENTURE-TEMPLATE

Creating objects in sentence 3 for C-OWN-PERCENT-TEMPLATE marker{17} with
 (?OWNER ?PERCENT) variables

Calling Trumpet with SENSE Interpretation:
(C-OWN-PERCENT-TEMPLATE
 (R-OWNER (CNAME_TAGA-CO1 (R-NAME TAGA-CO) (R-PART (C-ENTITY))))
```

```
(R-PERCENT (REMAINDER)))
```

TRUMPET then takes these pieces of semantic interpretation and tries to map them onto a final template, applying domain constraints, reference resolution, and heuristics for merging and splitting information from multiple sentences and paragraphs.

Discourse Processing

Before producing the final template, SHOGUN must take all the references to objects and events and try to resolve them. Often the resolution of object references affects the resolution of event references, because the objects become the only tie-in from one description of an event to the next.

The discourse processing knowledge of the system is considerably more developed in English than in Japanese. This is a case where it was difficult to do all the experiments we would have liked because our developers were not bilingual, and discourse cues in Japanese are often fairly subtle.

In EJV 0592, the system correctly resolves most of the event and object references, but still does badly on the location and activity site slots because it assumes that the location of the joint venture company is the location of the production activity, and it fails to guess that "Kaohsiung" is in Taiwan. In addition, there is a very subtle inference here that the production of clubs in Japan is not an additional location for the production of clubs by the Taiwan unit; SHOGUN treats both Japan and Taiwan as production bases.

```
;;; Removing nations (TAIWAN) which conflict with the organizations
;;; Replacing R-VENTURE references (COMPANY) with (CNAME_BRIDGESTONE-SPORTS-TAIWAN-CO1)
;;; Removing references (CONCERN HOUSE) from R-PARTNER (CNAME_TAGA-CO1
    CNAME_UNION-PRECISION-CASTING-CO1 CNAME_BRIDGESTONE-SPORTS-CO1)
....
;;; Creating ACTIVITY template with 1 industries
;;; Resolving "THE  TAIWAN  UNIT" to PARTNER CNAME_UNION-PRECISION-CASTING-CO1
    for nationality "Taiwan (COUNTRY)" using location
;;; Resolving "THE  JAPANESE  SPORTS GOODS  MAKER" to PARTNER
    CNAME_BRIDGESTONE-SPORTS-CO1 for nationality "Japan (COUNTRY)"
```

TEXTRACT "OPTIONAL" SYSTEM

In order to process Japanese, the SHOGUN system uses a morphological analyzer called MAJESTY developed at NTT Data. As part of our early efforts in the Joint Venture domain, Tusyoshi Kitani of NTT Data (who was then a visiting scientist at Carnegie Mellon) wrote several AWK scripts to identify Japanese company names in the segmented output. Later, rules for identifying other kinds of text fields including proper names, locations, numbers and times were added. This year, he has extended this set of finite-state rules and augmented it with other modules to perform the entire TIPSTER task on Japanese texts. For the MUC-5 evaluation, we have submitted TEXTRACT's results on the JJV and JME texts as optional scores. These were officially scored by the government, and the results appear in the table.

	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

TEXTRACT: Overview

TEXTRACT is comprised of four major components: preprocessing, pattern matching, discourse processing and template generation. Although only the first of these modules is shared with the SHOGUN system,

both systems share the basic method of using finite-state pattern matching instead of full natural language parsing.

In the preprocessor, Japanese text is segmented into primitive words and they are tagged parts of speech by a Japanese segmenter called MAJESTY. Then, proper nouns, monetary, numeric and temporal expressions are identified by the proper noun recognizer. The comprising segments are grouped together to provide meaningful sets of segments to the succeeding processes [2]. The pattern matcher searches all possible patterns of interest in a sentence that match defined patterns such as tie-up relationships and economic activities. In the discourse processor, company names are identified uniquely throughout a text, allowing recognition of company relationships and correct merging of information within a text. Finally, the template generator puts extracted information together to create the required template format.

The JJV configuration of TEXTRACT has been under development since the Spring, and during the TIPSTER 18 month evaluation it achieved a recall of 29 and a precision of 70 (for an F-measure of 40.9). With 5 months of additional work, TEXTRACT now has a recall of 60 and a precision of 68, giving an F-measure of 63.8.

The JJV Textract system was ported to the microelectronics domain in three (3) weeks by one person. This was possible because most of the system's modules were shared across both domains (and because identifying company names is a key element to performance in both domains). Most of the development time was spent identifying key expressions from the corpus. The JME configuration of the TEXTRACT system performed about the same as the base SHOGUN system on JME, but had higher precision compared to the higher recall of SHOGUN.

Our experience with TEXTRACT confirms that finite-state pattern matching allows for very rapid development of high performance text extraction for new domains.

TEXTRACT: Company name identification throughout a text

Unifying multiple references to the same company throughout a text is key to achieving a high performance in the template structure of joint venture. A notion of "topic companies," which are the main concern in the sentence, was introduced. Topic companies are identified where subject case markers such as "が" and "は" appear. When a subject is missing in a sentence, which is often the case in Japanese, the subject is automatically assumed as the topic companies taken from the previous sentence.

Company aliases are identified by applying a substring matching algorithm called the longest common subsequence (LCS) method. References of three kinds of company name pronouns, "同社" (dousha; the company), "自社" (jisha; the company itself), and "両社" (ryousha; both companies) are also identified using the topic companies and some heuristic rules.

Every company name in the text, including company aliases and pronouns, is given a unique number by the discourse process. Using topic companies and the unique number, individual pieces of information identified by the preprocessor and the pattern matcher are merged together to generate a relevant template structure.

TEXTRACT: Analysis of a Walkthrough Message

In JJV0002, all five entities were correctly identified by the preprocessor. The pattern matcher also recognized two tie-ups correctly, although the pattern selected from four matched patterns was incorrect in Sentence 2 as shown in the traces below. TEXTRACT found one tie-up from Sentence 2, only because it cannot identify multiple tie-ups in a sentence with the current design.

```
Sentence no. = 1
@CNAME_PARTNER_SUBJ:0.97: = 東京海上火災保険
defined = は
@CNAME_PARTNER_WITH:0.94: = 今月から大和証券
defined = と
@SKIP =
defined = 提携
して保険と年金の機能を兼ね備えた新商品「ヒューマン」を売り出した。
```

```
Sentence no. = 2
@CNAME_PARTNER_WITH1:0.50: = また日新火災海上保険、同和火災海上保険の両社は山一証券
defined = と
@CNAME_PARTNER_WITH2:0.50: = 結んで年金型商品を近く、発売する予定で、損害保険業界の証券会社
defined = と
defined = の
@SKIP =
defined = 提携
商品づくりは一段と広がっている。
0002 ryosha = 同和火災海上保険, category = 1, distance = 3
0002 ryosha = 日新火災海上保険, category = 2, distance = 6
```

An alias "東京海上" (Toukyou kaijou) was found by the LCS method as a substring of the entity name "東京海上火災保険" (Toukyou Kaijou Kasai Hoken). References of "両社" (ryousha; both companies) were correctly resolved as "日新火災海上保険" (Nisshin Kasai Kaijou Hoken) and "同和火災海上保険" (Douwa Kasai Kaijou Hoken). After the discourse processing, entities were given unique numbers (unique_id) as follows:

```
gid = 1,  unique_id = 1, partner = 1, string = 東京海上火災保険
gid = 4,  unique_id = 4, partner = 1, string = 大和証券
gid = 5,  unique_id = 5, partner = 2, string = 日新火災海上保険
gid = 7,  unique_id = 7, partner = 2, string = 同和火災海上保険
gid = 9,  unique_id = 9, partner = 2, string = 山一証券
gid = 12, unique_id = 1, partner = 0, string = 東京海上
gid = 14, unique_id = 1, partner = 0, string = 東京海上
gid = 15, unique_id = 4, partner = 0, string = 大和証券
```

Industry objects and the product service slot were completely wrong due to the following reasons: (1) TEXTRACT did not find the Product/Service1 string, and (2) although it did spot the Product/Service2 string, it gave a wrong pointer to Activity1 due to a system bug. Another observation regarding the industry object was that TEXTRACT gave the industry type "sales" with SIC 50 to Product/Service2 as the SHOGUN system did.

COMBINING SYSTEMS: SHOGUN + TEXTRACT

For the Japanese Microelectronics 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:

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.

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

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

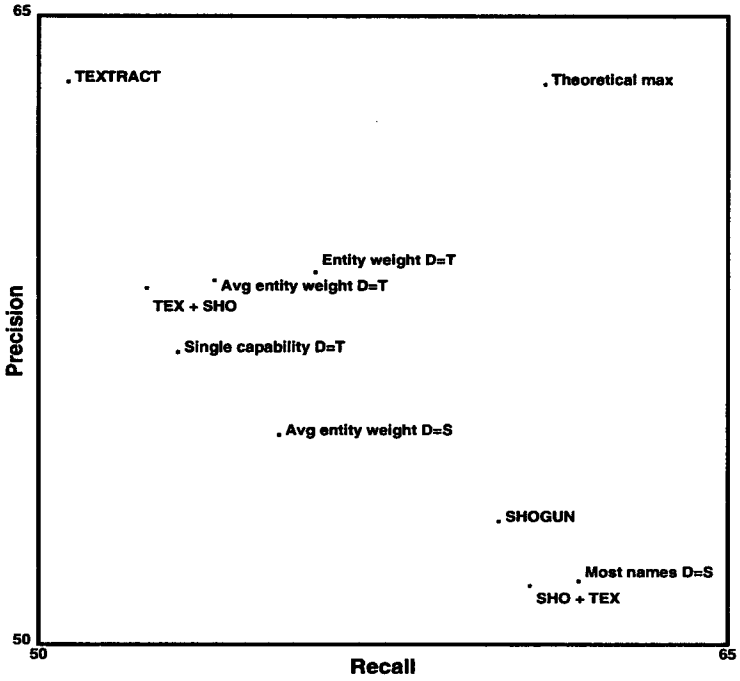

Figure 6: Combining Two MUC-5 Systems: Graph

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.

SUMMARY AND CONCLUSION

The examples and the analysis here are illustrative of the performance of the TIPSTER/SHOGUN system on MUC-5. While the system has done well and continued to improve significantly, there are still quite a number of problems that could be fixed to achieve better accuracy. On the other hand, the steady improvement of the system and the high performance across languages are very gratifying, and the fact that we already seem close to human performance seems to bode well for the deployment of this technology.

While research up to this point has emphasized interpretation and control issues, we see corpus analysis and knowledge acquisition algorithms as being the key topics for further research and further progress. In this way, MUC-5 may represent a turning point from matters of structure to matters of scale, with most of the necessary work on this type of task being broadening scope and scale. At the same time, we expect that simple but very challenging tasks will emerge that test some of the key algorithms that are required for data extraction.

References

[1] P. S. Jacobs and L. F. Rau. Innovations in text interpretation. *Artificial Intelligence (Special Issue on Natural Language Processing)*, 48, To Appear 1993.

[2] T. Kitani and T. Mitamura. A Japanese Preprocessor for Syntactic and Semantic Parsing. In *Ninth IEEE Conference on Artificial Intelligence for Applications*. IEEE, 1993.

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

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

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