UMASS/HUGHES: DESCRIPTION OF THE CIRCUS SYSTEM USED FOR TIPSTER TEXT

W. Lehnert, J. McCarthy, S. Soderland, E. Riloff, C. Cardie, J. Peterson, F. Feng
University of Massachusetts
Department of Computer Science
Box 34610
Amherst, MA 01003-4610
lehnert@cs.umass.edu

C. Dolan, S. Goldman
Hughes Research Laboratories
3011 Malibu Canyon Road M/S RL96
Malibu, CA 90265
cpd@aic.hrl.hac.com

INTRODUCTION

The primary goal of our effort is the development of robust and portable language processing capabilities for information extraction applications. The system under evaluation here is based on language processing components that have demonstrated strong performance capabilities in previous evaluations [Lehnert et al. 1992a]. Having demonstrated the general viability of these techniques, we are now concentrating on the practicality of our technology by creating trainable system components to replace hand-coded data and manually-engineered software.

Our general strategy is to automate the construction of domain-specific dictionaries and other language-related resources so that information extraction can be customized for specific applications with a minimal amount of human assistance. We employ a hybrid system architecture that combines selective concept extraction [Lehnert 1991] technologies developed at UMass with trainable classifier technologies developed at Hughes [Dolan et al. 1991]. Our Tipster system incorporates seven trainable language components to handle (1) lexical recognition and part-of-speech tagging, (2) knowledge of semantic/syntactic interactions, (3) semantic feature tagging, (4) noun phrase analysis, (5) limited coreference resolution, (6) domain object recognition, and (7) relational link recognition. Our trainable components have been developed so domain experts who have no background in natural language or machine learning can train individual system components in the space of a few hours.

Many critical aspects of a complete information extraction are not appropriate for customization or trainable knowledge acquisition. For example, our system uses low-level text specialists designed to recognize dates, locations, revenue objects, and other common constructions that involve knowledge of conventional language. Resources of this type are portable across domains (although not all domains require all specialists) and should be developed as sharable language resources. The UMass/Hughes focus has been on other aspects of information extraction that can benefit from corpus-based knowledge acquisition. For example, in any given information extraction application, some sentences are more important than others, and within a single sentence some phrases are more important than others. When a dictionary is customized for a specific application, vocabulary coverage can be sensitive to the fact that a lot of words contribute little or no information to the final extraction task: full dictionary coverage is not needed for information extraction applications.

In this paper we will overview our hybrid architecture and trainable system components. We will look at examples taken from our official test runs, discuss the test results obtained in our official and optional test runs, and identify promising opportunities for additional research.

TRAINABLE LANGUAGE PROCESSING

Our Tipster system relies on two major tools that support automated dictionary construction: (1) OTB, a trainable part-of-speech tagger, and (2) AutoSlog, a dictionary construction tool that operates in conjunction with the CIRCUS sentence analyzer. We trained OTB for EJV on a subset of EJV texts and then again for EME using only EME texts. OTB is notable for the high hit rates it obtains on the basis of relatively little training. We found that OTB attained overall hit rates of 97% after training on only 1009 sentences for EJV. OTB crossed the 97% threshold in EME after only 621 training sentences.

Incremental OTB training requires human interaction with a point-and-click interface. Our EJV training was completed after 16 hours with the interface; our EME training required 10 hours.

AutoSlog is a dictionary construction tool that analyzes source texts in conjunction with associated key templates (or text annotations) in order to propose concept node (CN) definitions for CIRCUS [Riloff & Lehnert 1993; Riloff 1993]. A special interface is then used for a manual review of the AutoSlog definitions in order to separate the good ones from the bad ones. Of 3167 AutoSlog CN definitions proposed in response to 1100 EJV key templates, 944 (30%) were retained after manual inspection. For EME, AutoSlog proposed 2952 CN definitions in response to 1000 key templates and 2275 (77%) of these were retained after manual inspection. After generalizing the original definitions with active/passive transformations, verb tense generalizations, and singular/plural generalizations, our final EJV dictionary contained 3017 CN definitions and our final EME dictionary contained 4220 CN definitions. It took 20 hours to manually inspect and filter the full EJV dictionary; the full EME dictionary was completed in 17 hours. The CIRCUS dictionary used in our official run was based exclusively on AutoSlog CN definitions. No hand-coded or manually altered definitions were added to the CN dictionary.

When CIRCUS processes a sentence it can invoke a semantic feature tagger (MayTag) that dynamically assigns features to nouns and noun modifiers. MayTag uses a feature taxonomy based on the semantics of our target templates, and it dynamically assigns context-sensitive tags using a corpus-driven case-based reasoning algorithm [Cardie 93]. MayTag operates as an optional enhancement to CIRCUS sentence analysis. We ran CIRCUS with MayTag for EJV, but did not use it for EME (we'll return to a discussion of this and other domain differences later). MayTag was trained on 174 EJV sentences containing 5591 words (3060 open class words and 2531 closed class words). Our tests indicate that MayTag achieves a 74% hit rate on general semantic features (covering 14 possible tags) and a 75% hit rate on specific semantic features (covering 42 additional tags). Interactive training for MayTag took 14 hours using a text editor.

An important aspect of the Tipster task concerns information extraction at the level of noun phrases. Important set fill information is often found in modifiers, such as adjectives and prepositional phrases. Part-of-speech tags help us identify basic noun phrase components, but higher-level processes are needed to determine if a prepositional phrase should be attached, how a conjunction should be scoped, or if a comma should be crossed. Noun phrase recognition is a non-trivial problem at this higher level. To address the more complicated aspects of noun phrase recognition, we use a trainable classifier that attempts to find the best termination point for a relevant noun phrase. This component was trained exclusively on the EJV corpus and then used without alteration for both EJV and EME. Experiments indicate that the noun phrase classifier terminates EJV noun phrases perfectly 87% of the time. 7% of its noun phrases pick up spurious text (they are extended too far), and 6% are truncated (they are not extended or extended far enough). Similar hit rates are found with EME test data: 86% for exact NP recognition, with 6% picking up spurious text and 8% being truncated. The noun phrase classifier was trained on 1350 EJV noun phrases examined in context. It took 14 hours to manually mark these 1350 instances using a text editor.

Before we can go from CIRCUS output to template instantiations, we create intermediate structures called memory tokens. Memory tokens incorporate coreference decisions and structure relevant information to facilitate template generation. Memory tokens record source strings from the original input text, OTB tags, MayTag features, and pointers to concept nodes that extracted individual noun phrases.

Discourse analysis contributes to critical decisions associated with memory tokens. Here we find the greatest challenges to trainable language systems. Thus far, we have implemented one trainable component that contributes to coreference resolution in limited contexts. We isolate compound noun phrases that are syntactically consistent with appositive constructions and pass these NP pairs on to a coreference classifier. Since adjacent NPs may be separated by a comma if they occur in a list or at a clause boundary, it is easy to confuse legitimate appositives with pairings of unrelated (but adjacent) NPs. Appositive recognition is therefore treated as a binary classification problem that can be handled with corpus-driven training. For our official Tipster runs we trained a classifier to handle appositive recognition using EJV development texts and then used the resulting classifier for both EJV and EME. Our best test results with this classifier showed an 87% hit rate on EJV appositives. It took 10 hours to manually classify 2276 training instances for the appositive classifier using a training interface.

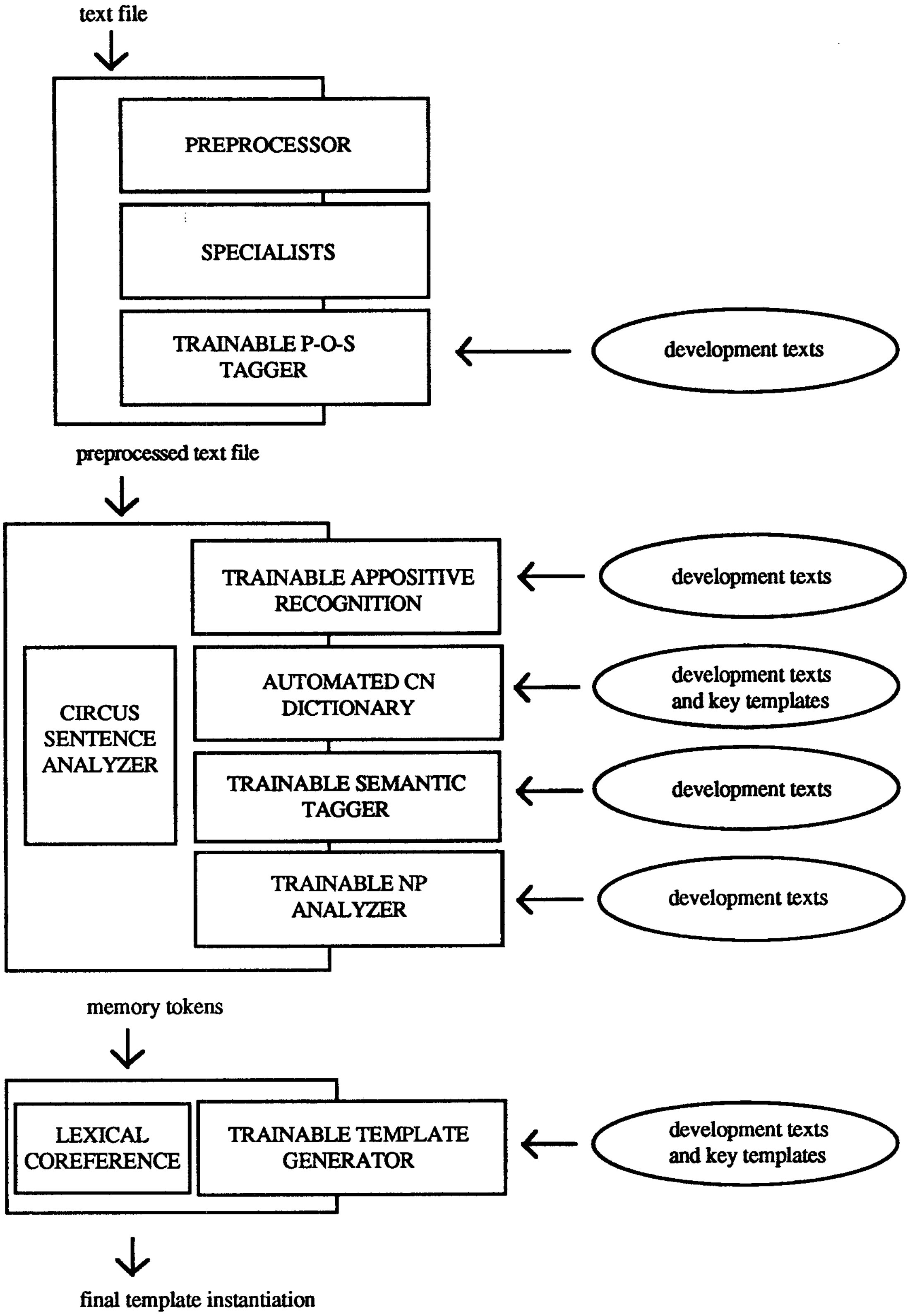

Figure 1: System Architecture

Our final tool, TTG, is responsible for the creation of template generators that map CIRCUS output into final template instantiations. TTG template generators are responsible for the recognition and creation of domain objects as well as the insertion of relational links between domain objects. TTG is corpus-driven and requires no human intervention during training. Application-specific access methods (pathing functions) must be hand-coded for a new domain but these can be added to TTG in a few days by a knowledgeable technician working with adequate domain documentation. Once these adjustments are in place, TTG uses memory tokens and key templates to train classifiers for template generation. No further human intervention is required to create the template generators, although additional testing, tuning and adjustments are needed for optimal performance.

Our hybrid architecture demonstrates how machine learning capabilities can be utilized to acquire many different kinds of knowledge from a corpus. These same acquisition techniques also make it easy to exploit the resulting knowledge without additional knowledge engineering or sophisticated reasoning. The knowledge we can readily acquire from a corpus of representative texts is limited with respect to reusability, but nevertheless cost-effective in a system development scenario predicated on customized software. The trainable components used for ***both*** EJV and EME were completed after 101 hours of interactive work by a human-in-the-loop. Moreover, most of our training interfaces can be effectively operated by domain experts: programming knowledge or familiarity with computational linguistics is generally not required. (Although one technical background is needed to train OTB.). Near the end of this paper will report the results of a system development experiment that supports this claim.

There will always be a need for some amount of manual programming during the system development cycle for a new information extraction application. Even so, significant amounts of system development that used to rely on experienced programmers have been shifted over to trainable language components. The ability to automate knowledge acquisition on the basis of key templates represents a significant redistribution of labor away from skilled knowledge engineers, who need access to domain knowledge, directly to the domain experts themselves. By putting domain experts into the role of the human-in-the-loop we can reduce dependence on software technicians. When significant amounts of system development work is being handled by automated knowledge acquisition and expert-assisted knowledge acquisition, it will become increasingly cost-effective to customize and maintain a variety of information extraction applications. We have only just begun to explore the range of possibilities associated with trainable language processing systems.

The hybrid architecture underlying our official Tipster systems was less than six months old at the time of the evaluation, and most of the trainable language components that we utilized were less than a year old. Less than 24 person/months were expended for both of the EJV and EME systems, although this estimate is confounded by the fact that trainable components and their associated interfaces were being designed, implemented, and tested by the same people responsible for our Tipster system development. The creation of a trainable system component represents a one-time system development investment that can be applied to subsequent systems at much less overhead.

Figure 1 outlines the basic flow-of-control through the major components of the UMass/Hughes Tipster system. Note that most of the trainable components depend only on the texts from the development corpus. The concept node dictionary and the trainable template generator also rely on answer keys during training. In the case of the concept node dictionary, we have been able to drive our dictionary construction process on the basis of annotated texts created by using a point-and-click text marking interface. So the substantial overhead associated with creating a large collection of key templates is not needed to support automated dictionary construction. However, we do not see how to support trainable template generation without a set of key templates, so this one trainable component requires a significant investment with respect to labor.

SYSTEM EVOLUTION AND PROJECT GOALS

Our preparation for Tipster was somewhat limited relative to other Tipster extraction sites. We began our effort one year later than other sites and we were not funded to work with Japanese. Our funding (and system development) began October 1, 1992 and the 24-month evaluation took place in July 1993. During that period we redesigned the overall system architecture initially described in our proposal, and designed a number of trainable system components from scratch. OTB, AutoSlog, and MayTag were already available, but needed to be trained and applied to the new domains. Two project personnel attended the 12-month Tipster meeting in September 1992, and that was our first introduction to the domain guidelines for EJV and EME.

Our initial system design was based on a simplistic model of how the UMass and Hughes technologies demonstrated at MUC-3 and MUC4 might be brought together into a hybrid information extraction system. The original idea was to pass CIRCUS output to TTG (the Hughes Trainable Template Generator) without further alteration to either CIRCUS or TTG. This approach was put into place for the 18-month Tipster evaluation in February 1993 and found to be inadequate for a number of reasons. First, the output of CIRCUS was highly fragmented and completed unstructured at the discourse level. CIRCUS was extracting but not organizing related information even when that information all came from the same sentence. So TTG was being asked to consolidate information that required reorganization at many levels. This problem was aggravated by the fact that CIRCUS tended to create a lot of output for a given text. Some of this output was irrelevant or off-target, and some of it was legitimate but redundant. TTG was expected to sort out the good from the bad and not get tangled up in the redundancies. In fact, TTG was designed to handle noise in the training data, but the amount of data being generated by CIRCUS created massive training runs for TTG. TTG had never been pushed this hard before, so we found ourselves contending with memory limitation problems and long runtimes. Since TTG is not just one decision tree but a collection of about 30 decision trees, these additional complications produced significant stumbling blocks.

We barely had a working system in place for the 18-month evaluation in February and there was no opportunity to adjust this initial implementation before the evaluation. As painful as it was to subject a first-pass system to a formal evaluation, the exercise left no doubt about the problems inherent in our system design. The hand-coded consolidation heuristics developed at UMass for MUC-3 and MUC-4 could not be functionally duplicated by TTG alone. Additional processing needed to be inserted between CIRCUS and TTG. We were also losing a lot of information that should have been recognized during preprocessing by low-level specialists designed to pick up dates and locations. These specialists were not particularly challenging, but they did require a lot of programming time and we had not made an adequate investment in our preprocessing specialists.

At the same time, we were pleased with the trainable components that allowed us to customize CIRCUS for two new domains. The OTB part-of-speech tagger needed some adjustments but seemed quite promising. The AutoSlog dictionary construction tool had worked very well for EJV and reasonably well for EME. We came to understand that EME was heavily dependent on keyword recognition for its technical vocabulary. AutoSlog had not been designed to collect or organize extensive synonym lists, but this was not an inherently difficult task.

Six months prior to the final Tipster evaluation, we had to make some major decisions. One of our options was to hand-code new consolidation heuristics for EJV and EME. We knew how to do this based on similar efforts for MUC-3 and MUC-4, and we probably would have been able to improve the performance of our system dramatically in the six months remaining to us by following this route. But we had made a research commitment to investigate trainable technologies for information extraction system development: we were interested in finding alternatives to hand-coded heuristics. So we began to look at the gap between CIRCUS and TTG with an eye for the problems that might be managed by trainable decision trees.

We identified three phenomena that seemed to be causing significant difficulties for us at the 18-month evaluation: (1) noun phrase termination, (2) appositive recognition, and (3) coreference resolution. Noun phrase termination refers to the problem of knowing when an NP can be extended across potential boundary markers like commas, conjunctions and prepositional phrases. Appositive recognition is a subset of the general coreference resolution problem, but we found it useful to separate out appositives because appositive candidates can be reliably recognized on the basis of a syntactic pattern, which makes appositives somewhat easier to tackle than the general coreference problem.

We worked for a few weeks on the design of feature vectors for the appositive problem and the noun phrase termination problem. Training instances were then collected and our first decision trees for these problems were running in March. Baseline comparisons with hand-coded heuristics for noun phrase termination showed that a hand-coded component was able to make noun phrase termination decisions correctly about 65% of the time. ID3 decision trees were showing us hit rates in the 80-85% range. We found a similar level of success for our ID3 appositive d-trees. These trees were generally able to categorize potential appositive candidates correctly about 85% of the time. Subsequent experimentation with these modules during March and April failed to improve our initial performance levels.

During this same period we were also first coming to appreciate the difficulties inherent in a system based on many trainable components. With changes still occurring to critical upstream components like the processing specialists and OTB, we knew that older training data for AutoSlog and our ID3-based components was slowly falling out of sync with the rest of the system. It wasn't a good idea to train a tree on one set of data and then test the tree on a substantially different data set. If a training set incorporated a certain amount of noise (e.g. false hits with respect to locations), and the test data contained no noise or a much reduced noise level, an appositive tree or noun phrase termination tree might not operate as effectively as it would if the noise levels were consistent across both data sets. But it wasn't practical for us to continually update the training sets in order to keep everything in phase. So we were constantly working with components that were either outdated or out of phase, and it was difficult to know how significant that complication would prove to be. We were also dealing with a lot of software engineering complexity in trying to manage all the data sets, decision trees, and various configurations of the system under inspection. Our Tipster system development effort was proving to be much more complicated than our MUC-3 and MUC-4 efforts had been.

In all system development efforts, downstream development depends on upstream stability, and this is especially true when a number of trainable components are involved. The introduction of major new components six months before the final Tipster evaluation was far from ideal. In particular, we were up against the fact that we could not retrain TTG until the new components were producing output. We knew that TTG would benefit from a lot of experimentation, but we couldn't do anything about that until we had new capabilities in place to handle noun phrases, appositives, and coreference. Noun phrase termination was very important for AutoSlog, so we held off on our final dictionary construction until May in order to benefit from our progress on noun phrase termination.

In retrospect, we can also see that we were too slow to get started on the coreference module. We knew that coreference would benefit from noun phrase termination and appositives, so it made sense to delay the coreference work until we had made at least some progress on these other components. But we did not appreciate how much more complicated data collection for coreference would prove to be, and we did not allow enough time to design feature vectors for coreference. Coreference data collection was not tackled before May, and progress on coreference went much more slowly than expected. We did eventually train some decision trees for general coreference resolution, but our preliminary test results were not strong, so we were forced to abandon trainable coreference in the eleventh hour. Manually-coded heuristics were then created in an effort to manage at least some coreference decisions prior to template generation.

The difficulties associated with coreference and the last minute drive to pull together manual coreference heuristics prevented us from experimenting with TTG as much as we would have liked. Some improvements were made to make TTG more efficient and less memory intensive, but nothing could be done that depended on memory token input. In the end, we were not able to begin TTG training and experimentation until July, at which time we were struggling to produce a working system in time for the final evaluation.

The lack of complete system throughput until July meant that we couldn't run the scoring program in order to obtain internal benchmarks during the six months prior to the 24-month evaluation. We ran the scoring program on EME output for the first time on July 21, and on EJV output for the first time on July 25. The official test runs were executed on July 31.

Our previous experience with MUC-3 and MUC-4 taught us that significant improvements to score reports can be made by studying high-frequency slots and looking for simple adjustments that improve performance for these slots. There was no time to experiment with any such adjustments in EJV although some effort was made to optimize the EME system in response to some internal testing based on the 18-month test set. Unfortunately, these EME adjustments may have backfired for reasons that we will describe in the next section.

Despite our abbreviated development schedule and difficulties establishing upstream system stability, we learned a lot about the integration of machine learning techniques into natural language processing systems. Although Tipster was originally conceived to obtain a comparative evaluation of mature text processing technologies, we always viewed our Tipster effort as being somewhat more exploratory in nature. No one had previously attempted to integrate NLP techniques with machine learning techniques at any of the previous MUC conferences. The UMass/Hughes system represented an ambitious undertaking that would have benefited greatly from another year of collaboration.

THE OFFICIAL TEST RUNS

Our official test runs were conducted on three DECstations running Allegro Common Lisp. The runs went smoothly in EME but we encountered one fatal error in one of the EJV test sets. Portions of our official EJV and EME score reports are shown in figures 2 and 3.

AO = all objects
MO = matched objects
TF = text filtering
FM = F-measures

AO	ERR	SUB	REC	PRE	UND	OVG
MO						
TF	77	25	26	54	65	28
FM	48	17	55	76	34	8
			65	94	35	6

	P&R	2P&R	P&2R
F-measures	35.18	44.39	29.13

Figure 2: Official EJV Score Report Summaries

	ERR	SUB	REC	PRE	UND	OVG
AO	77	24	31	39	59	48
MO	54	15	60	59	30	31
TF			85	76	15	23

	P&R	2P&R	P&2R
F-measures	34.84	37.42	32.59

Figure 3: Official EME Score Report Summaries

	ERR	SUB	REC	PRE	UND	OVG
AO	67	17	44	52	47	37
MO	45	11	65	70	27	20
TF			81	84	19	16

	P&R	2P&R	P&2R
F-measures	47.47	50.08	45.11

Figure 4: Unofficial Tips2 EME Score Reports

Based on the minimal amount of internal testing conducted prior to the official test runs, we were surprised to see that our EME scores were about the same as our EJV scores. We had expected to do much better on the EME test set. In the week prior to the final evaluation, we made adjustments to the EME system based on internal testing with the Tips2 test set from the 18-month evaluation. Subsequent testing on Tips2 suggested that we should have done much better on the official test runs.

We had restricted our feedback loop to this one test set because we knew that answer keys compiled for official test sets tend to be more reliable than answer keys in the general development corpus. We also expected the test sets from the 18-month evaluation to be predictive of the test sets used for the 24-month evaluation. Unfortunately, this turned out not to be the case with the Tips2 EME test set. After the final evaluation, we discovered that Tips2 contained a disproportionate number of short texts (97% of the Tips2 texts were less than 2200 bytes long). We had inadvertently tuned our EME system for short texts without realizing it. The EME test set used for the 24-month evaluation contained a large number of texts that were significantly longer than 2200 bytes.

	the complete test set					the short texts only			
	E	R	P	F		E	R	P	F
Tips3/1	79	27	39	32		70	38	53	44
Tips3/2	77	33	37	35		70	43	45	44
Tips3/3	75	35	40	38		71	38	48	43
Tips2	67	44	52	47					

E = Error rate R = Recall P = Precision F = P&R F-score
(all objects scores only)

Figure 5: EME Tuning Effects for Short Texts

We can see the effects of this disparity between training materials and test materials if we compare the overall EME score reports with score reports based on only a subset of the shorter EME test texts. Figure 5 shows how our system would have performed if the EME test set had been restricted to shorter texts that were consistent with the Tips2 test set. When we test the system on only the shorter texts from the Tips3 test set, we see significantly higher test scores. The Tips3 subset scores are a little weaker than the Tips2 scores, but that difference is probably due to the fact that we were tuning the system in response to Tips2 and we would therefore expect to see our strongest possible performance on those same texts.

THE OPTIONAL TEST RUNS

We ran optional tests to see what sort of recall/precision trade-offs were available from the system. Since the template generator is a set of classifiers, and each classifier outputs a certainty associated with a hypothesized template fragment, we have many parameters that can be manipulated. Raising the threshold on the certainty for a hypothesis will, in most cases, increase precision and reduce recall. In the experiments reported here, we have varied the parameters over broad classes of discrimination trees. There are three important classes of decision tree: (1) trees that filter the creation of objects based on string fills, (2) trees that filter the creation of objects based on set fills, and (3) trees that hypothesize relations among objects. An example of the first class is the tree that filters the CIRCUS output for entity names in the EJV domain. An example of the second class is the tree that filters possible lithography objects based on evidence of the type of lithography process. The trees that hypothesize TIE_UP_RELATIONSHIP's and ME_CAPABILITY's are examples of the third class.

For these experiments we have varied the certainty thresholds for all trees of a given class. Figure 6 shows the trade-off achieved for EME.

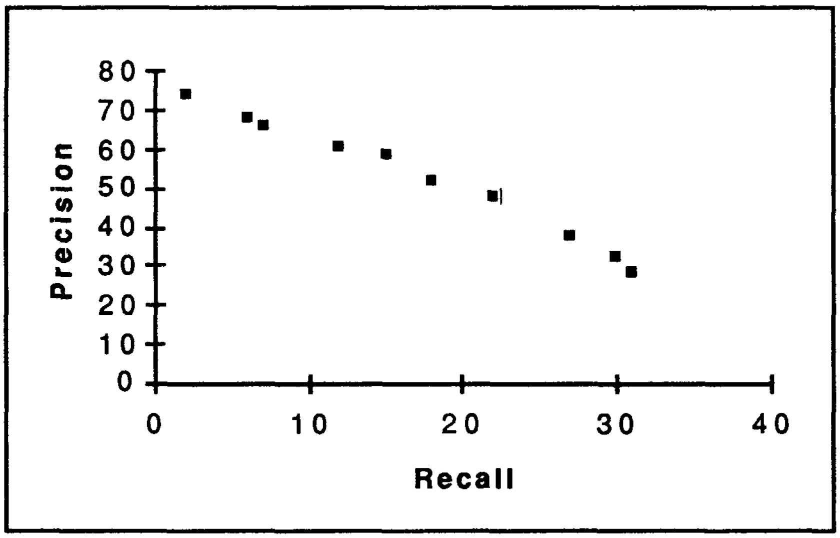

Figure 6: Trade-off curve for EME

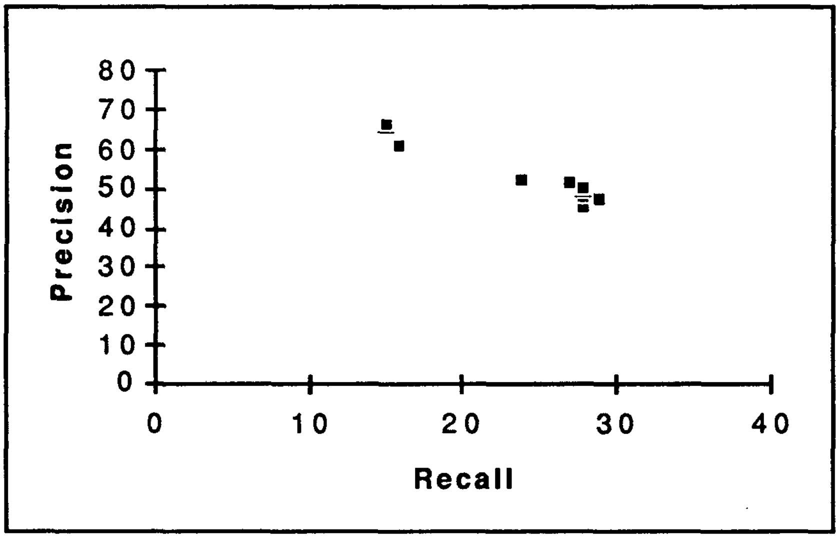

Figure 7: Trade-off curve for EJV

This trade-off curve was achieved by varying, in concert the thresholds on all three classes of discrimination tree from 0.0 to 0.9. Figure 7 shows the trade-off curve achieved in EJV. The difference between the two curves highlights difference between the two domains and between the system configurations used for the two domains. The EME curve shows a much more dramatic trade-off. The EJV curve shows that only modest varying of recall and precision is achievable. Part of this is a reflection of the two domains. In EJV, most relationships were found via two noun phrases that shared a common CN trigger. This method proved to be effective at detecting relationships. Therefore the only real difference in the trade-off comes from varying the thresholds for the string-fill and set-fill trees, which generate the objects that are then composed into relationships. In EME, there not nearly as many shared triggers and so the template generator must attempt intelligent guesses for relations. The probabilistic guesses made in EME are much more amenable to threshold manipulation than the more structured information used in EJV. Also, in EJV the system ran with a slot masseur that embodied some domain knowledge. In EJV, TTG was configured to only hypothesize objects if the slot masseur had found a reasonable slot-fill or set-fill. This use of domain knowledge further limited the efficacy of changing certainty thresholds.

TRAINABLE INFORMATION EXTRACTION IN ACTION

Before CIRCUS can tackle an input sentence, we have to pass the source text through a preprocessor that locates sentence boundaries and reworks the source text into a list structure. The preprocessor replaces punctuation marks with special symbols and applies text processing specialists to pick up dates, locations, and other objects of interest to the target domain. We use the same preprocessing specialists for both EJV and EME: many specialists will apply to multiple domains. A subset of the Gazetteer was used to support the location specialist, but no other MRDs are used by the preprocessing specialists. We do not have a specialist that attempts to recognize company names. Figure 8 shows sample output from the preprocessor along with subsequent processing.

Note that the date specialist had to consult the dateline of the source text in order to determine that

Source Text:

```
BRIDGESTONE SPORTS CO. SAID FRIDAY IT HAS SET UP A JOINT VENTURE IN TAIWAN WITH A LOCAL
CONCERN AND A JAPANESE TRADING HOUSE TO PRODUCE GOLF CLUBS TO BE SHIPPED TO JAPAN.
```

Preprocessed Text:

```
(*START* BRIDGESTONE SPORTS CO= SAID **ON_241189 IT HAS SET UP A JOINT VENTURE IN @@_Taiwan WITH A LOCAL
CONCERN AND A JAPANESE TRADING HOUSE TO PRODUCE GOLF CLUBS TO BE SHIPPED TO @@_Japan *END*)
```

OTB Tags:

```
(*START* BRIDGESTONE  SPORTS  CO=   SAID  **ON_241189  IT    HAS  SET   UP   A    JOINT
 strt    nm           nm      noun  verb  $date$       noun  aux  pasp  ptcl art  nm

VENTURE  IN    @@_Taiwan   WITH  A    LOCAL  CONCERN  AND   A    JAPANESE  TRADING
noun     prep  $location$  prep  art  nm     noun     conj  art  nm        nm

HOUSE  TO   PRODUCE  GOLF  CLUBS  TO   BE   SHIPPED  TO    @@_Japan    *END*)
noun   inf  verb     nm    noun   inf  aux  pasp     prep  $location$  stop
```

extracted NPs	rejected CN features:	accepted CN features:
BRIDGESTONE SPORTS CO=	jv-person	jv-entity, jv-parent, jv
IT	jv-person	
A JOINT VENTURE		jv-entity, company
GOLF CLUBS		jv-prod_serv, production

Figure 8: Preprocessing and CIRCUS analysis

"Friday" must refer to November 24, 1989. Once the preprocessor has completed its analysis, the OTB part-of-speech tagger identifies parts of speech: OTB tagged 97.1% of the words in EJV 0592 correctly. One error associated with "... A COMPANY ACTIVE IN TRADING WITH TAIWAN ..." led to a truncated noun phrase when "active" was tagged as a head noun instead of a nominative predicate.

With part-of-speech tags in place, CIRCUS can begin selective concept extraction. On this first sentence from EJV 0592, CIRCUS triggers 18 CN definitions triggered by the words "said" (3 CNs), "set" (3 CNs), "venture" (9 CNs), "produce" (1 CN), and "shipped" (2 CNs). These CNs extract a number of key noun phrases, and assign semantic features to these noun phrases based on soft constraints in the CN definition. Some of these features were recognized to be inconsistent with the slot fill and others were deemed acceptable. For example, in Figure 8 we see that different CNs picked up "BRIDGESTONE SPORTS CO." with incompatible semantic features (it was associated with both a *joint venture* and a *joint venture parent* feature).

As we can see from this sentence, CN feature types are not always reliable, and CIRCUS does not always recognize the violation of a soft feature constraint. An independent set of semantic features are obtained from MayTag. In the first sentence of EJV 0592, MayTag only missed marking "golf clubs" as a product/ service. An independent set of semantic features are obtained from MayTag as shown in Figure 9.

In addition to extracting some noun phrases and assigning semantic features to those noun phrases, we also call the noun phrase classifier to see if any of the simple NPs picked up by the CN definitions should be extended to longer NPs. For this sentence, the noun phrase classifier extended only one NP: it decided that "A JOINT VENTURE" should be extended to pick up "A JOINT VENTURE IN TAIWAN WITH A LOCAL CONCERN AND A JAPANESE TRADING HOUSE". The second prepositional phrase should not have been included - this is an NP expansion that was overextended.

Each noun phrase extracted by a CIRCUS concept node will eventually be preserved in a memory token that records the CN features, MayTag features, any NP extensions, and other information associated with CN definitions. But before we look at the memory tokens, let's briefly review the other NPs that are extracted from the remainder of the text. For each preprocessed sentence produced in response to EJV 0592, we will put the noun phrases extracted by CIRCUS into boldface and use underlines to indicate how the noun phrase classifier extends some of these NPs.

```
words              MayTag semantic features                          hits & misses

*START*
BRIDGESTONE        ((WS-JV-ENTITY)  (WS-COMPANY-NAME))               ; correct
SPORTS             ((WS-JV-ENTITY)  (WS-COMPANY-NAME))               ; correct
CO=                ((WS-JV-ENTITY)  (WS-GENERIC-COMPANY))            ; correct
SAID
IT                 ((WS-ENTITY) NIL)                                 ; correct
HAS
SET
UP
A
JOINT              ((WS-JV-ENTITY) NIL)                              ; correct
VENTURE            ((WS-JV-ENTITY) NIL)                              ; correct
IN
JV-LOCATION        ((WS-LOCATION) NIL)                               ; correct
WITH
A
LOCAL              ((WS-ENTITY) NIL)                                 ; correct
CONCERN            ((WS-JV-ENTITY) NIL)                              ; correct
AND
A
JAPANESE           ((WS-NATIONALITY) NIL)                            ; correct
TRADING            ((WS-PRODUCT-SERVICE)  (WS-SALES))                ; correct
HOUSE              ((WS-JV-ENTITY) NIL)                              ; correct
TO
PRODUCE
GOLF               ((WS-ENTITY) NIL)                                 ; incorrect
CLUBS              ((WS-ENTITY) NIL)                                 ; incorrect
TO
BE
SHIPPED
TO
JV-LOCATION        ((WS-LOCATION) NIL)                               ; correct
```

Figure 9: MayTag semantic feature tags

```
*START* BRIDGESTONE SPORTS CO= SAID **ON_241189 IT HAS SET UP A JOINT VENTURE IN @@_Taiwan WITH A LOCAL
CONCERN AND A JAPANESE TRADING HOUSE TO PRODUCE GOLF CLUBS TO BE SHIPPED TO @@_Japan *END*) (*START*
THE JOINT VENTURE $COMMA$ BRIDGESTONE SPORTS TAIWAN CO= $COMMA$ CAPITALIZED AT $$20000000_TWD
$COMMA$ WILL START PRODUCTION **DURING_0190 WITH PRODUCTION OF &&20000 IRON AND METAL WOOD CLUBS A
MONTH *END*) (*START* THE MONTHLY OUTPUT WILL BE LATER RAISED TO &&50000 UNITS $COMMA$ BRIDGESTONE
SPORTS OFFICIALS SAID *END*) (*START* THE NEW COMPANY $COMMA$ BASED IN KAOHSIUNG $COMMA$ SOUTHERN
TAIWAN $COMMA$ IS OWNED %%75 BY BRIDGESTONE SPORTS $COMMA$ %%15 BY UNION PRECISION CASTING CO= OF
@@_Taiwan AND THE REMAINDER BY TAGA CO= $COMMA$ A COMPANY ACTIVE IN TRADING WITH TAIWAN $COMMA$
THE OFFICIALS SAID *END*) (*START* BRIDGESTONE SPORTS HAS SO_FAR BEEN ENTRUSTING PRODUCTION OF GOLF
CLUB PARTS WITH UNION PRECISION CASTING AND OTHER TAIWAN COMPANIES *END*) (*START* WITH THE
ESTABLISHMENT OF THE TAIWAN UNIT $COMMA$ THE JAPANESE SPORTS GOODS MAKER PLANS TO INCREASE
PRODUCTION OF LUXURY CLUBS IN @@_Japan *END*)
```

Figure 10: Noun phrase analysis

As far as our CN dictionary coverage is concerned, we were able to identify all of the relevant noun phrases needed with the exception of "A LOCAL CONCERN AND A JAPANESE TRADING HOUSE" which should have been picked up by a JV parent CN. In fact, our AutoSlog dictionary had two such definitions in place for exactly this type of construction, but neither definition was able to complete its instantiation because of a previously unknown problem with time stamps inside CIRCUS. This was a processing failure—not a dictionary failure.

Trainable noun phrase analysis processes 13 of the 17 NP instances shown in Figure 10 correctly. Three of the NPs were expanded too far, and one was expanded but not quite far enough due to a tagging error by OTB ("a company active ..."). An inspection of the 13 correct instances reveals that 7 of these would have been correctly terminated by simple heuristics based on part-of-speech tags. It is important to note that the trainable NP analyzer had to deduce these more "obvious" heuristics in the same way that it deduces decisions for more complicated decisions. It is encouraging to see that straightforward heuristics can be acquired automatically by trainable classifiers. When our analyzer makes a mistake, it generally happens with the more complicated noun phrases (which is where hand-coded heuristics tend to break down as well).

After the noun phrase classifier has attempted to find the best termination points for the relevant NPs, we then call the coreference classifier to consider pairs of adjacent NPs separated by a comma. In this text we find three such appositive candidates (the second of which contains an extended NP that was not properly terminated):

```
THE JOINT VENTURE, BRIDGESTONE SPORTS TAIWAN CO.
TAGA CO., A COMPANY ACTIVE
THE NEW COMPANY, BASED IN KAOHSIUNG, SOUTHERN TAIWAN
```

In the third case, the location specialist failed to recognize either Kaohsiung or Southern Taiwan as names of locations. On the other hand, the fragment "based in Kaohsiung" was recognized as a location description and therefore reformatted it as "THE NEW COMPANY (%BASED-IN% KAOHSIUNG), SOUTHERN TAIWAN" which set up the entire construct as an appositive candidate. The coreference classifier then went on to accept each of these three instances as valid appositive constructions. This was the right decision in the first two cases, but wrong in the third. If full location recognition had been working, this last instance would have never been handed to the coreference classifier in the first place.

The coreference classifier tells us when adjacent noun phrases should be merged into a single memory token. We also invoke some hand-coded heuristics for coreference decisions that can be handled on the basis of lexical features alone. These heuristics determine that Bridgestone Sports Co. is coreferent with Bridgestone Sports, and that "THE JOINT VENTURE,, BRIDGESTONE SPORTS TAIWAN CO." is coreferent with "A JOINT VENTURE IN TAIWAN ..." Our lexical coreference heuristics are nevertheless very conservative, so they fail to merge our four product service instances in spite of the fact that "clubs" appears in three of these string fills. In effect, we pass the following memory token output to TTG:

5 recognized companies (#4 and #5 should have been merged):

```
(1) "TAGA CO=" aka "A COMPANY ACTIVE"
(2) "UNION PRECISION CASTING CO= OF @@_TAIWAN"
(3) "BRIDGESTONE SPORTS CO="  aka "BRIDGESTONE SPORTS"
(4) "THE NEW COMPANY (%BASED-IN% KAOHSIUNG)" aka "SOUTHERN TAIWAN"
(5) "THE JOINT VENTURE" aka "BRIDGESTONE SPORTS TAIWAN CO=" aka
        "A JOINT VENTURE IN @@_TAIWAN WITH A LOCAL CONCERN AND A JAPANESE TRADING HOUSE"
```

4 product service strings (all of these should have been merged):

```
(6) "GOLF CLUBS"
(7) "&&20000 IRON AND METAL WOOD CLUBS"
(8) "GOLF CLUB PARTS WITH UNION PRECISION CASTING AND OTHER TAIWAN COMPANIES"
(9) "LUXURY CLUBS IN @@_JAPAN"
```

1 ownership and 2 percent objects :

```
(10) "$$20000000_TWD"
     (11) "%%15"
     (12) "%%75"
```

We failed to extract "the remainder by ..." for the third ownership object because our percentage specialist was not watching for non-numeric referents in a percentage context - this could be fixed with an adjustment to the specialist.

Figure 11: Extracted Text Strings

When TTG receives memory tokens as input, the object existence classifiers try to filter out spurious information picked up by overzealous CN definitions. Unfortunately, in the case of 0592, TTG filtered out two good memory tokens: (#1 describing the parent Tago Co.), and (#5 describing the joint venture). It was particularly damaging to throw away #5 because that memory token contained the correct company name (Bridgestone Sports Taiwan Co.). Of the 3 remaining memory tokens describing companies, TTG correctly identified the two parent companies on the basis of semantic features, but then it was forced to pick up #4 as the child company. Our pathing function was smart enough to know that "THE NEW COMPANY" was probably not a good company name, but that left us with "SOUTHERN TAIWAN" for the company name. So a failure that started with location recognition led to a mistake in trainable appositive recognition, which then combined with a failure in lexical coreference recognition and a filtering error by TTG in order to give us a joint venture named "SOUTHERN TAIWAN" instead of "BRIDGESTONE SPORTS TAIWAN." Overly aggressive filtering by TTG resulted in the loss of our 4 product service memory tokens.

Our CN instantiations do not explicitly represent relational information, but CNs that share a common trigger word can be counted on to link two CN instantiations in some kind of a relationship. Trigger families can reliably tell us when two entities are related, but they can't tell us what that relationship is. We relied on TTG to deduce specific relationships on the basis of its training. In cases like "75% BY BRIDGESTONE SPORTS", TTG had no trouble linking extracted percentage objects with companies. But our trainable link recognition ran into more difficulties when trigger families contained multiple companies Among the features that TTG had available for discrimination where closed class features, such as memory token types, semantic features, and CN patterns, and open class features (i.e. trigger words). However, although there exist heuristics for discriminating relationships based on particular words, the combination of the algorithms used (ID3) and the amount of data (600 stories) failed to induce these heuristics. There may be other algorithms, however, that can use the same or less data and external knowledge to derive such heuristics from the training data.

The processing for EME proceeds very similarly to EJV, with the exception that MayTag is not used in our EME configuration, and in the EME system we used our standard CN mechanism and an additional keyword CN mechanism (KCN). The KCN mechanism was used to recognize specific types of processing, equipment, and devices that have one or only a few possible manifestations. In Figure 12 we see the OTB tags for the first sentence, all of which are correct. In fact, for EME text 2789568, OTB had 100% hit rate.

The memory token structure in Figure 12 illustrates the processing of the text prior to TTG. Two NPs are identified as the same entity, "Nikon" and "Nikon Corp." The two NPs are merged into one memory token based on name merging heuristics. The second NP demonstrates how multiple recognition mechanisms can add robustness to the processing.

```
*start*  **DURING_2Q91      $COMMA$     Nikon  Corp=  $LPAREN$      &&7731
strt     $date$            punc        nm     noun   punc          noun
$RPAREN$  plans   to    market   the   NSR-1755EX8A  $COMMA$    a    new    stepper
punc      verb    inf   verb     art   noun          punc       art  nm     noun
intended  for   use   in    the   production   of     &&64-   Mbit   DRAMs    *end*
pasp      prep  noun  prep  art   noun         prep   nm      nm     noun     stop


 (TOKEN
  (TYPE (ME-ENTITY))
  (SUBTYPE NIL)
  (RELATION NIL)
  (SLOT-FILLS
    (TYPE COMPANY)
    (NAME (:SYM-LIST NIKON CORP=)))
  (NPS
    (NP 2 1 (NIKON)
      (CNS %ME-ENTITY-NAME-SUBJECT-VERB-AND-DO-STEPPER%))
    (NP 0 3 (NIKON CORP=)
      (CNS %ME-ENTITY-NAME-SUBJECT-VERB-AND-INFINITIVE-PLANS-TO-MARKET%)
      (KCNS %KEYWORD-ME-ENTITY-CORP=%
            %LEAD-NP%))))
```

Figure 12: EME processing

FEATURE	RELATION CERTAINTY AFTER FEATURE
	0.36
The process is not X-RAY	0.23
The entity is not triggered off "developed"	0.14
The process is not CVD	0.03
The process is not LITHOGRAPHY of UKN type	0.04
The process is not ETCHING	0.06
The entity is not triggered off "from"	0.12

Figure 13: ME-CAPABILITY developer features

FEATURE	RELATION CERTAINTY AFTER FEATURE
	0.38
The process is not packaging	0.47
The entity is not in a PP	0.58
A CN marked the entity as an entity	0.55
The process is not layering type sputtering	0.40

Figure 14: ME-CAPABILITY distributor features

"Nikon Corp." is picked up by both a CN triggered off of "plans to market" and by two KCNs, one that looks for "Corp." and another that looks for the lead NP in the story.

Unfortunately, our system did not get any lithography objects for this story. On our list of things to get to if time permitted was creating a lithography object for an otherwise orphaned stepper. We would have only gotten one lithography object since we merged all mentions of "stepper" into one memory token.

We created a synthetic version of the system that inserted a lithography memory token corresponding to each stepper. One was discard by TTG and another was created because there were two different equipment objects attached to the remaining lithography object. The features that TTG used to hypothesize a new ME_CAPABILITY are illustrative of one of the weaknesses of this particular method. TTG used the features in Figure 13 to decide not to generate an ME_CAPABILITY developer.

All of the features are negative, and the absence of each feature reduces the certainty that the relation holds, because each feature's presence, broadly speaking, is positive evidence of a relation. Therefore, the node of the decision tree that is found is a grouping of cases that have no particular positive evidence to support the relation, but also no negative evidence. With the relation threshold set at 0.3, this yields a negative identification of a relation. However, there are strong indications of a relation here. For example, the trigger "plans to market" is good evidence of a relation, however, the nature decision tree algorithms (recursively splitting the training data) causes us to lose that feature (in favor of other, better features). Figure 14 shows what features TTG used to generate an ME_CAPABILITY distributor.

Again, we do not see here the features that we would expect, given the text. A human generating rules would say that "plans to market" is a good indication of a ME_CAPABILITY distributor.

WHAT WORKS AND WHAT NEEDS WORK

When we look at individual texts and work up a walk through analysis of what is and is not working, we find that many of our trainable language components are working very well. The dictionary coverage provided by AutoSlog appears to be quite adequate. OTB is operating reliably enough for subsequent sentence analysis. When we run into difficulties with our trainable components, we often find that many of these difficulties stem from a mismatch of training data with test data. For example, when we trained the coreference interface for appositive recognition, we eliminated from the training data all candidate pairs involving locations because the location specialist should be identifying locations for us. If the coreference classifier were operating in an ideal environment, it would never encounter unrecognized locations. Unfortunately, as we saw with EJV 0592, the location specialist does not trap all the locations, and this led to a bad coreference decision. In an earlier version of the coreference classifier we had trained it on imperfect data containing unrecognized locations, but as the location specialist improved, we felt that the training for the coreference classifier was falling increasingly out of sync with the rest of the system so we updated it by eliminating all the location instances. Then when the coreference classifier was confronted with an unrecognized location, it failed to classify it correctly. When upstream system components are continually evolving (as they were during our TIPSTER development cycle), it is difficult to synchronize downstream dependencies in training data. A better system development cycle would stabilize upstream components before training downstream components in order to maintain the best possible synchronization across trainable components.

The importance of reliable representative training materials was demonstrated to us with even more impact after the final 24-month EME evaluation when we discovered that the EME test materials used for the 18-month evaluation were not representative of the test materials used for the 24-month evaluation. As explained earlier, our EME system was tuned for shorter texts than we encountered in the official test sets. With a trainable system, it is crucial to use representative texts for all parameter tuning. TTG was easy to tune by straightforward experimentation with different parameter settings, but we failed to demonstrate its full utility in EME because of the differences between Tips2 and Tips3.

TTG nevertheless enhanced the output of CIRCUS and other discourse processing modules. In module-specific testing TTG typically added 6-12% of accuracy in identifying domain objects and relationships. That added value is measured against picking that most likely class (yes or no) for a particular domain object (e.g. JV-ENTITY or ME-LITHOGRAPHY) or relationship (e.g. JV-TIE-UP or ME-MICROELECTRONICS-CAPABILITY). However, TTG fell far below our expectations for correctly filtering and connecting the parser's output. We find two reasons for this short fall. First, some amount of the deficit can be attributed to the system development cycle since TTG sits at the end of the cycle of training and testing various modules.

The second, and by far the dominant effect comes from the combination of the training algorithm (ID3) and the amount of data. As mentioned previously, there are two types of features used by TTG: (1) closed class (e.g. token type, semantic features, and CN patterns) and (2) open class features (i.e. CN trigger words). Using open class features can be difficult, because most algorithms cannot detect reliable discriminating features if there are too many features—reliable features cannot be separated from noise. Using trigger words in conjunction relations between memory token results in 3,000-5,000 binary features. With no noise suppression added to the algorithm and given a large number of features, ID3 will create very deep decision trees that classify stories in the training set based on noise.

We ran two sets of decision trees in deciding how to configure our system for the final test run. MIN-TREES using only closed class features and no noise suppression and MAX-TREE using closed class and open class features and a noise suppression rule. The noise suppression was a termination condition on the recursion of the ID3 algorithm. Recursion was terminated when all features resulted in creating a node that classified examples from few than 10 different source texts. Using closed class features rarely resulted in a terminal node that classified examples from fewer than 10 stories. In all tests the MAX-trees performed better. However, as a result of the noise suppression, no decision tree contained very many discriminations on a trigger. The performance of the MAX-trees indicated that individual words are good discriminators, however their scarcity in the decision trees indicates that we are not using the appropriate algorithm. We believe that data-lean algorithms (such as explanation-based learning) in concert with shared knowledge bases might be effective.

In attributing performance to various components, we measured 25 random texts in EME. At the memory token stage we found that CIRCUS had extracted string-fills and set-fills with a recall/precision of 68/54. However our score output for those slots was 32/45 (measured only on the slots we attempted). Even when the thresholds for TTG were lowered to 0.0, so that all output came through, the recall was not anywhere near 68. Therefore it would appear that the difficult part of the template task is not finding good things to put in the template, but figuring how to split and merge objects. We do not (yet) have a trainable component that handles splitting and merging decisions in general.

The EJV and EME systems that we tested in our official evaluation were in many ways incomplete systems. Although our upstream components were operating reasonably well, additional feedback cycles were badly needed for other components operating downstream. In particular, trainable coreference and trainable template generation did not received the time and attention they deserve. We are generally encouraged by the success of our trainable components for part-of-speech tagging, dictionary generation, noun phrase analysis, semantic feature tagging, and coreference based on appositive recognition. But we encountered substantial difficulties with general coreference prior to template generation. This appears to be the greatest challenge remaining for trainable components supporting information extraction. We know from our earlier work in the domain of terrorism that coreference resolution can be reasonably well-managed on the basis of hand-coded heuristics [Lehnert et al. 1992b]. But this type of solution does not port across domains and therefore represents a significant system development bottleneck. True portability will only be achieved with trainable coreference capabilities.

We believe that trainable discourse analysis was the major stumbling block standing between our Tipster system and the performance levels attained by systems incorporating hand-coded discourse analysis. We remain optimistic that state-of-the-art performance will be obtained by corpus-driven machine learning techniques but it is clear that more research is needed to meet this very important challenge. To facilitate research in this area by other sites, UMass will make concept extraction training data (CIRCUS output) for the full EJV and EME corpora available to research laboratories with internet access. When paired with Tipster key templates available from the Linguistic Data Consortium, this data will allow a wide range of researchers who may not be experts in natural language to tackle the challenge of trainable coreference and template generation as problems in machine learning. We believe it is important for the NLP community to encourage and support the involvement of a wider research community in our quest for practical information extraction technologies.

SYSTEM REQUIREMENTS

The final 24-month evaluation was conducted on one DECstation 5000/240 and two DECstation 5000/133s. Each machine was configured with 64MB RAM and a 300 MB internal disk. Two of these machines also had 1.35 GB external disks. The 5000/240 was running at 40 MHz and the 5000/133s ran at 33 MHz. All of our code was written in Allegro Common Lisp 4.1 and ran under the Ultrix 4.2 operating system.

Our source code occupies about 3.6 MB of disk space. The data used in the evaluation (including response templates and trace files for both EJV and EME) takes up 26.9 MB. It took 27 hours of run time (distributed across the three machines) to complete the EJV test sets. It took 12 hours to complete the EME test sets. The difference in these run times is due to the fact that we used the MayTag semantic feature tagger on the EJV texts but not the EME texts. No attempt was made to optimize runtimes in either system.

Additional system development was also conducted on a number of Mac IIs configured with Macintosh Common Lisp 2.0. Both of our Tipster systems can be run on a Mac (a minimum of 8 MB RAM is recommended). The AutoSlog training interface, OTB training interface, and appositive training interface were all implemented and run on MACs using the Macintosh Common Lisp Interface Toolkit, as was the system demo presented at the 24-month meeting.

ACKNOWLEDGMENT

This work was supported by the Department of Defense under Contract No. MDA904-92-C-2390. The views expressed in this paper are those of the authors and should not be interpreted to represent opinions or policies of the United States Government.

BIBLIOGRAPHY

Cardie, C. (1993). A Case-Based Approach to Knowledge Acquisition for Domain-Specific Sentence Analysis. *Eleventh National Conference on Artificial Intelligence* (AAAI-93). Washington, D.C. pp. 798-803.

Dolan, C. P., Goldman, S. R., Cuda, T. V., & Nakamura, A. M. (1991). Hughes Trainable Text Skimmer: Description of the TTS System as Used for MUC-3. In B. Sundheim (Ed.), *Third Message Understanding Conference* (MUC-3). Naval Ocean Systems Center, San Diego California: Morgan Kaufmann. pp. 155-162.

Lehnert, W. (1991). Symbolic/Subsymbolic Sentence Analysis: Exploiting the Best of Two Worlds. *Advances in Connectionist and Neural Computation Theory. Vol. I.* (ed: J. Pollack and J. Barnden) Ablex Publishing, Norwood, New Jersey. pp. 135-164.

Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E., Soderland, S. (1992a). University of Massachusetts: MUC-4 Text Results and Analysis *Proceedings of the Fourth Message Understanding Conference* (MUC-4). Morgan Kaufmann. San Mateo, CA. pp. 151-158.

Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E., Soderland, S. (1992b). Description of the CIRCUS system as Used for MUC-4. *Proceedings of the Fourth Message Understanding Conference* (MUC-4). Morgan Kaufmann. San Mateo, CA. pp. 282-288.

Quinlan, J. R. (1983). Learning Efficient Classification Procedures and Their Application to Chess End Games. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* . Morgan Kaufmann. pp. 463-482.

Riloff, E. (1993). Automatically Constructing a Dictionary for Information Extraction Tasks. *Eleventh National Conference on Artificial Intelligence* (AAAI-93). Washington, D.C. pp. 811-816.

Riloff E., and Lehnert, W. (1993). Automated Dictionary Construction for Information Extraction from Text. *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications.* IEEE Computer Society Press. pp. 93-99.