MITRE-Bedford: Description of the *Alembic* System as Used for MUC-5

John Aberdeen, John Burger, Dennis Connolly, Susan Roberts, & Marc Vilain

The MITRE Corporation
202 Burlington Road
Bedford, MA 01730

BACKGROUND

The *Alembic* text understanding system fielded at MUC-5 by MITRE is an extensive rewrite of the system we presented at MUC-4. *Alembic*[1] is a research prototype that is still young, and is thus in the process of ongoing development. Our work is part of an internally-funded research initiative aimed at processing open source texts, *i.e.*, free natural language texts drawn from broadcast transcripts, news wires, *etc.* This initiative explores several major research areas:

- Error recovery, primarily involving issues of semi-parsing and recovery of plausible attachments.
- Robustness, primarily involving issues of uncertain reasoning and tractable inference.
- Self-extensibility, focusing primarily on machine learning of natural language and user-configurable semantics.
- System integration, through SGML (the Standard Generalized Markup Language), both at the level of meaning analysis and at the overall application level.

ARCHITECTURE

The system's underlying architecture, shown in Figure 1 overleaf, follows a by-now familiar task breakdown. Processing occurs in two main phases: preprocessing (in the elementizer module), which covers processes ranging from document destructuring to simple parsing, and natural language analysis per se, including application-specific output generation. As with several other MUC systems, our use of an explicit pre-processing phase is directly borrowed from work by Jacobs *& al* (1991). One way *Alembic* differs from other MUC systems, however, is in exploiting SGML (Goldfarb 1990) as the interchange lingua franca between processing phases. The intention is to allow system modules whose invocation occurs early in the analysis of a document to record processing results directly in the document through SGML markup. This information then becomes available to subsequent modules as meta-data. Further, the elementizer itself is composed of a sequence of processing phases that also exchange partial results through SGML.

As a result of this SGML-based architecture, the system's overall flow of control is governed from an object-oriented document manager built on top of Goldfarb's public domain SGML parser, ARC-SGML. For MUC-5, the pre-processing elementizer thus takes a source message file and normalizes it in various ways, yielding an intermediate SGML document. The document manager then builds an internal document object by parsing the resulting SGML. The actual content analysis of the document is performed by invoking the natural language analysis modules on the internal document object, and the results of these analyses are stored as attributes of the document.

Another way in which *Alembic* differs from a prototypical MUC-class system is in its emphasis on text enrichment. That is, the output of the system is actually a new version of the source document enriched by semantic markup. The markup consists of a template-based index and simple semantic annotations identifying people, organizations, places, *etc.* For the MUC task, *Alembic* also provides selective output that consists solely of filled templates.

[1]**alembic** **1** : an alchemical apparatus used for distillation **2** : something that refines or transmutes as if by distillation.

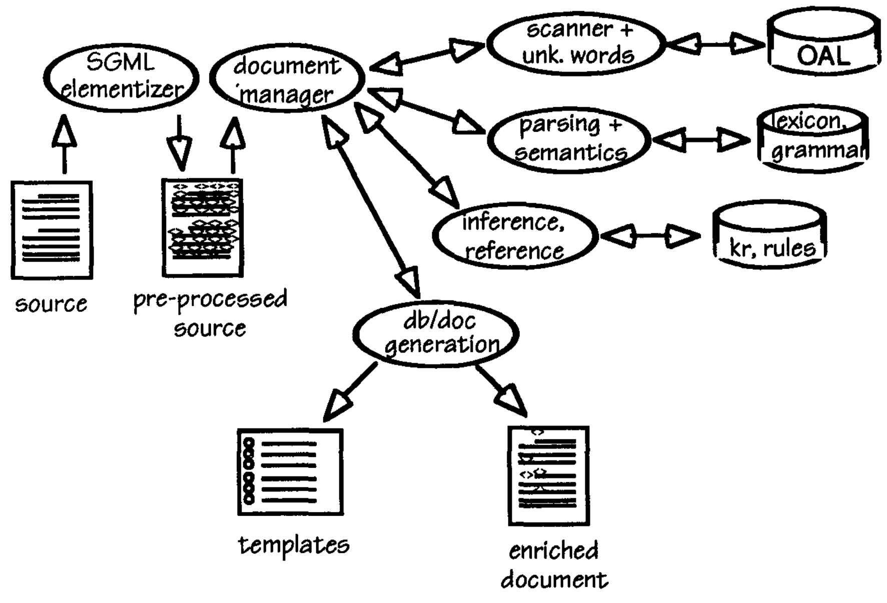

Figure 1: System Architecture

	Zoner	Prepro.	Prepar.	Parse	Defrag.	Interp.	Ambig.	Disc.	Templ.
Elementizer	✓	✓	✓						
Scanner		✓							
Parser				✓		✓			
Inference					✓		✓	✓	
Generator									✓

Table 1: Correspondence between *Alembic* and Hobbs' proto-system

INDIVIDUAL PROCESSING MODULES

As shown in Figure 1, *Alembic* comprises six main processing modules. These modules perform all but one of the functions described by Hobbs (1993) for his generic information extraction system—we omit the text filtering phase. The correspondence from Hobbs' inventory of processing modules to *Alembic* is summarized in Table 1.

Elementizer

The elementizer (so called because it inserts SGML elements into the text) is built as a cascade of finite-state LEX-style tokenizers. The elementizer performs Hobbs' zoning, pre-processing, and pre-parsing functions. Our zoning is more sophisticated than that in most like systems because we interpret the markup of MUC-5 messages as *bona fide* SGML. Of Hobbs' several pre-processing tasks, the elementizer performs two: detecting sentence and paragraph boundaries, and tokenizing certain parts of the lexicon. The parts in question include some gargantuan sub-lexica that we do not want to have to load into our main processor, e.g., lists of personal names and the inventory of geography terms from the TIPSTER gazetteer. Also tokenized are contractions and other punctuation-laden lexemes, and assorted closed classes such as corporate designators and currency names. Finally, the elementizer performs some rudimentary parsing of dates and other constructs whose idiosyncrasies are much easier to handle in a finite-state framework than in the strictly binary-branching framework of our main parser.

The following elementized sample reproduces parts of the second paragraph of the walkthrough message. Note, for example, normalization tags such as <NUM> (for numerals) and tokenization tags, including <PT> (for punctuation), <GEO> (Gazetteer entries), and <LEX> (misc. closed classes).

```
<P> THE JOINT VENTURE <PT>,</PT> BRIDGESTONE SPORTS <GEO co>TAIWAN</GEO> <LEX> CO.</LEX>
<PT>,</PT> ... A MONTH<PT>.</PT> <S>THE MONTHLY OUTPUT <NAME>WILL</NAME> BE LATER RAISED
TO <NUM V="50000">50,000</NUM> UNITS<PT>,</PT> BRIDGESTON SPORTS OFFICIALS SAID<PT>.</PT>
```

The elementizer expects its output to be normalized with respect to an SGML document type definition at some later point, in our case as the elementized source file is read into the document manager. This allows us to exploit a number of SGML shortcuts, such as implicit tags, thus simplifying the elementization task significantly. In the preceding sample, for example, note how unambiguous </P>, <S> and </S> tags are not introduced. They are reconstructed from context by the document manager when the elementized message is read into *Alembic*. More esoterically, perhaps, we also exploit short forms of attribute names so as to minimize the size of elementized messages. The <GEO co> tag for example is actually short for (roughly)

```
<GEO continent=NIL country=T province=NIL city=NIL airport=NIL port=NIL island=NIL island_group=NIL>
```

Document Manager

The document manager provides an object-oriented framework for working with SGML documents. The manager is entirely CLOS-based, and SGML elements are thus made available as instances of CLOS objects. A sentence element (corresponding to the string bracketed by matching <S> and </S> tags) is mapped into an instance of the S class in CLOS, and any S -specific code (e.g., the linguistic parser) is thus made applicable to the element.

As mentioned, the document manager is built around a public domain SGML parser/tokenizer written by Goldfarb, the originator of SGML. The parser takes care of canonicalizing a document as it is read in by, e.g., inserting any tags left implicit by the elementizer, or by filling in the default attribute values of attributes.

The parser consists of C language routines made available through the Consortium for Lexical Research. On the Lisp side, the document manager makes available a simple text-processing protocol that is easily specialized for contentful elements, such as <S>, <SO>, and so forth. We exploit this protocol to structure the entire flow of control of the main body of *Alembic*, from invoking the scanner all the way through producing template files.

Scanner

The scanner performs the remainder of Hobbs' pre-processing phase. That is, it builds a chart over which the parser will operate, and populates the chart with categories corresponding to either (1) ordinary lexical items in the input stream, or (2) SGML elements identified by the elementizer (geographic terms, dates, and so forth). The latter are handled by specializing the document manager's text processing protocol. <GEO> elements, for example, instantiate a prototypical geographical NP category into the chart (our categories are structured attribute value vectors with variables, i.e., standard unification-style DAGs (Shieber 1986; Pareschi & Steedman 1987)):

```
[[syn :N]
 [bar-level 2]
 [sem [[head geo-region]]]]
```

Non-SGML items are handled through a three-level lexicon. Our primary lexicon contains lexical items to which is associated fine-grained information. This may include complex syntactic relationships, as with sentential complement verbs, or they may be associated with meaningful semantics, as with domain-specific lexicon entries. The primary lexicon was populated in part by adapting knowledge bases provided to us by Richard Tong of ADS.

If a lexical item is not found in the primary lexicon, we obtain its part of speech from a secondary lexicon based on the OAL (Oxford Advanced Learner) word list. This occurs most often for open-class tokens out of the immediate realm of the domain. Finally, if an item does not even appear in the OAL, we assume it to be an instance of a special unknown noun category, UNK-N. This category is allowed to appear anywhere that is permitted for common nouns. In addition, UNK-Ns can participate in many syntactic positions that would otherwise be reserved to proper nouns, e.g., as part of personal names and as pre-nominal modifiers to the left of the adjective position.

Linguistic Parser

The original parser in *Alembic* was strictly based on combinatory categorial grammars (Steedman 1985, 1987). We originally chose CCGs because their processing is inherently bottom-up and because they are strongly lexically governed. One lesson learned in our MUC-4 experience, however, is that strongly categorial frameworks are inadequate for processing free text, chiefly because they only provide inconvenient treatments of modifier attachment.

As a result, one change we brought to our parser in the last year has been to recast it as a hybrid of the categorial and phrase structure frameworks. Specifically, we rely on structured CCG-style categories to handle argument structure, and on binary-branching phrase structure rules to handle modifier attachment and other general phrase structure phenomena. A transitive verb, for example is treated as the V/N category, i.e., as a function that takes a noun phrase argument on the right (the N term) and produces a verb phrase (the V term). The noun phrase is attached to the verb through the combinatorial rule of function application, i.e.,

$$\frac{\alpha/\beta \quad \beta}{\alpha}$$

Strictly speaking, the above is a combinatorial schema, in which α and β in this case match V and N respectively. As in standard CCGs, a single combinatorial function application schema is used for attaching all syntactic arguments to a functor category. Complex argument attachment phenomena are thus encoded through complex syntactic categories, e.g., V/N/N (for ditransitive verbs such as "give") or V/$\bar{S}$/N (for "tell" and similar sentential complement verbs).

In contrast, modifier attachment is handled by providing rules specific to the particular modification phenomenon, as in standard phrase structure grammar. The temporal modifiers of a verb, for example, are handled through the following rule (not a schema).

$$\frac{\mathrm{V} \quad \mathrm{DATE}}{\mathrm{V}}$$

There are a number of advantages to this approach with respect to linguistic modeling. Beyond handling modifiers, Wittenburg (1987) used related strategies to project gapped constituents. From a computational perspective, however, the most important is that it makes the grammar entirely unary- or binary-branching. This is so because the syntactic phenomena that must traditionally be treated through greater-than-binary rules are typically complex argument structures—in our framework, these phenomena are handled through "stacked" functor categories, such as V/N/N.[2] By relying on the binary-branching property of the grammar, we can thus simplify many aspects of our parser; we don't need to construct dotted rules, for example.

Although there are many further interesting and unconventional aspects of *Alembic*'s parser, they are only of secondary interest to an information extraction audience. For the present purpose, we will thus only allude to some of the parser's other relevant characteristics.

- The parser organizes rules according to specificity, so that only the most specific variant of a phrase structure rule is ever used in combining two categories.
- We rely on a number of simple attachment disambiguation heuristics to reduce the number of parses generated for a given sentence. These heuristics are simple and have so far proved themselves workable in practice. In particular, we prefer argument attachments to modifier attachments, and among multiple modifier attachments, we prefer those that attach deepest in the parse tree.
- Although the parser often produces spanning parses of even long sentences, we only require it to produce semi-parses. We use a greedy algorithm to segment the chart into maximal chunks and use these when a spanning parse is unavailable.

[2]There are admittedly non-binary phenomena besides complex argument structure, for example the grammar of dates. We handle these cases through the finite-state pre-parsing that is performed by the *Alembic* elementizer.

Semantic interpretations

The parser produces semantic interpretations in concert with syntactic parses. The output of the parser is effectively a semantic interpretation of the predicate argument structure of the sentence. We use semantic structures derived from those in the King Kong NL interface (Bayer & Vilain 1991); these are in turn related to the "interpretation level" of Alshawi & Van Eijck (1989). These are semantic objects that have four primary components:

- *Head:* that is, a semantic predicate corresponding to the lexical head of a phrase.
- *Args:* a list of semantic interpretations corresponding to the syntactic arguments of the lexical head.
- *Proxy:* a semantic individual that instantiates the lexical head.
- *Mods:* a set of interpretations corresponding to the syntactic modifiers of the phrase.

For example, the following is the top level interpretation of "The company last March set up a joint venture."

```
[[head create-entity]                              "set up"
 [args ( [[head company]...]                       "the company"
         [[head business-venture]...] )]           "a joint venture"
 [proxy c-e-1]
 [mods { [[head time-of]                           "last March"
          [args ( c-e-1 [[head date]...] ) ] ...] } ]]
```

In this case, the head slot of the interpretation corresponds to the main verb of the sentence, and the args slot holds the arguments of the verb, e.g., the subject and direct object. The proxy slot denotes a semantic individual which serves the role of an event instance in a partially Davidsonian scheme, as in (Hobbs 1985) or (Bayer & Vilain 1991). That is, modifiers of the lexical head (from the mods slot) are made to predicate the proxy individual., as in the example with the date modifier "last March," which predicates a time-of relation of the proxy individual c-e-1.

As with other frameworks with similar interpretation levels, it is worth observing that these representations are ambiguous with respect to quantifier scope. Although quantifier information is saved during parsing (as a fifth quant slot on interpretations), it is ignored except in as much as is needed for determining definiteness of referential expressions. As noted below, this representational ambiguity is of interest because we perform inference and data extraction directly upon propositionalized versions of these interpretations, with no need to scope quantifiers.

Inference

The inference module is a completely new addition to *Alembic*. It is built on top of a propositional database that stores propositionalized versions of the semantics of sentences. The database itself is coupled to a tractable inference mechanism (Vilain 1991, 1993) and to an equality reasoner based on congruence closure (McAllester 1989).

With respect to data extraction, the inference module performs two chief functions. First it provides a uniform framework that seamlessly integrates general-purpose semantic processing and the bulk of data extraction. Second, it serves as a common substrate upon which *Alembic* performs various kinds of discourse processing, ranging from general definite reference resolution to domain-specific event or entity merging. As we note below, the inferential nature of the process allows these two functions to be performed through simple rules, thus avoiding the monolithic and unwieldy pattern-matching expressions that are typically used for semantic processing and event merging.

This material is best illustrated through examples—see below in our discussion of the walkthrough message.

Template Generation

As mentioned, we rely on the inference mechanism to perform the bulk of the work of data extraction. The template generator, to a large extent, simply reads the template structure out of the contents of the propositional database. The primary complexity lies in maintaining backpointers from facts in the propositional database to the text strings required to produce string fills. Since these fills are typically NPs, this is simply achieved by mapping semantic individuals to their originating lexical heads, and mapping these in turn to their maximal projections.

In addition, the template generator must handle the innumerable details concerning fill rules for geography, names, and aliases. These minutiæ are handled through a combination of inference rules and ad hoc code.

SEMANTIC PROCESSING IN THE WALKTHROUGH MESSAGE

From a data extraction perspective, most of the early stages of processing *Alembic* performs on this message are only of partial interest. Although *Alembic* has its own distinctive ways of pre-processing, scanning, and parsing the document, the heart of the actual extraction process takes place during inference. As this is also one of the distinguishing characteristics of *Alembic*, our description of the walkthrough message thus focuses largely on the inferential process.

Mapping parses to propositions

Inference begins when the semantic interpretations produced by the parser are added to the propositional database. For example, for the first sentence in the message, the parser produces the following schematic syntactic attachments (two fully spanning parses are produced for this sentence).

```
BRIDGESTONE SPORTS CO.
    SAID FRIDAY
        IT WILL 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
```

The actual semantic interpretations are nested structures that generally mirror this attachment scheme: the top level of the interpretation, for example, is provided below. Note that in the particular case of verbs with sentential complements (e.g., "Bridgestone Sports Co. said"), the propositional complement (e.g., "it will set up") is actually pulled out to the top level of the interpretation, and the matrix sentence is demoted to a modifier role.

```
[[head create-entity]                              "has set up"
 [args ( [[head pronominal-thing]]                 "it"
         [[head business-venture]] )]              "a joint venture"
 [proxy p1]
 [mods { [[head declare]                           "said Friday"
          [args ( [[head company]] p1 ) ]] } ]]    "Bridgestone Sports Co."
```

A nested interpretation such as this is then added to the database by flattening and propositionalizing it. For the first sentence of the walkthrough message, the following propositions are added (among others).

```
(company c1)                       "Bridgestone Sports Co."
(declare (c1 c-e-1))               "said Friday"
(pronominal-thing t1)              "it"
(create-entity (t1 b-v-1) c-e-1)   "has set up"
(business-venture b-v-1)           "a joint venture"
```

Note again that this process of propositionalization does not require making any choice of quantifier scoping.

General semantic inference

Once a sentence has been propositionalized and added to the database, the first set of inferences performed are typically of a general semantic nature. In the case of the first walkthrough sentence, the first set of rules to derive new facts are those that identify the participants of the events. A very general such rule pulls participants out of the conjoined co-agentive phrase "with a local concern and a Japanese trading house." To begin with, this co-agentive phrase causes the propositions below to be added in addition to those shown above.

(co-agent (bv1 g1))	"with"
(group g1)	*coordination*
(group-member (g1 lc1))	"a local concern"
(group-member (g1 jth1))	"a Japanese trading house"

The extraction of participants is performed by the following rule:

(event x) & (co-agent (x y)) & (group y) & (group-member (y z)) → (participant (x z))

This rule yields:

(participant (bv1 lc1))
(participant (bv1 jth1))

An additional rule identifies these participants as venture entities:

(business-venture x) & (participant (x y)) → (jv-entity (x y)),

yielding:

(jv-entity (bv1 lc1))
(jv-entity (bv1 jth1))

Together, these two rules identify the existence of two of the joint venture participants (but not their names). The third participant is identified (modulo reference resolution) by a lexical rule about creating business ventures:

(create-entity (x y)) & (business-venture y) → (participant (x y)).

This rule, along with the immediately preceding one yield:

(participant (bv1 pt1))	i.e., "it" is a participant in the business venture
(jv-entity (bv1 pt1))	i.e., "it" is a venture entity of the venture

These rules typify the kind of inference that goes on in *Alembic*, wherein domain-specific lexical inferences build directly on top of general domain-independent inferences. This provides an appealing degree of declarative modularity, and allows significant portions of the semantic infrastructure to be reused across multiple domains.

Discourse processing

Perhaps the most appealing aspect of *Alembic*'s propositional database is that it provides a very convenient substrate for discourse processing. Indeed, *Alembic* exploits the equality mechanism built into the database to cause facts to become propagated and shared as a result of coreference determinations. In the version of *Alembic* used in MUC-5, these co-reference determinations can happen in three different ways.

Definite reference

The definite reference mechanism incorporated into the MUC-5 version of the system is a drastic simplification of the probabilistic mechanism we used in MUC-4 (Aberdeen *& al* 1992). We restricted the mechanism to consider only designated classes in our sort hierarchy, namely joint ventures and companies. Once a definite reference to a designated class has been identified, it is simply resolved to the most recent instance of that class. This strategy is error-prone, and we intend to replace it shortly with a probabilistic reference mechanism more in line with the one we used in MUC-4 (but trained on more reliable data!). What is most interesting about our approach, however, is not the particular reference algorithm we ended up implementing, but the way it exploits the equality mechanism.

To see this, consider sentence 2 of the walkthrough message, where we encounter the definite reference "The joint venture, Bridgestone Sports Taiwan Co." This phrase engenders the following propositions in the database:

(business-venture bv2)	"The joint venture"
(has-name (bv2 c2))	*apposition*
(company c2)	"Bridgestone Sports Taiwan Co."

The definite reference is resolved to the phrase "a joint venture" from sentence 1, which corresponds to the individual bv1. The individuals bv1 and bv2 are then equated, and the following is automatically inferred:

(has-name (bv1 c2))

Other facts known about bv2 are automatically propagated to bv1 as well.

Event and entity merging

Event and entity merging approaches to discourse processing have gained significant visibility recently in light of the success of systems with very fragmentary parsing, e.g., CIRCUS (Lehnert *& al* 1992) or FASTUS (Appelt *& al* 1993). Although we have only had limited experience with this class of techniques, we believe that entity and event merging can be elegantly handled through *Alembic*'s propositional database. For instance, consider sentence 4 of the walkthrough message, which refers to "The new company, based in Kaohsiung." The MUC-5 version of our system contained the following entity-merging rule, which can be glossed as saying that any new company mentioned in the context of a joint venture is in fact that joint venture (or perhaps its joint venture company):

(company x) & (new-thing x) & (jv-in-context (x y)) → (= (x y)),

As a result of applying this rule, *Alembic* propagates facts known about the company in sentence four to the joint venture entity from sentence 1, e.g., the fact that it is based in Kaohsiung, and so forth. What is most appealing about merging events in this way is that it avoids much of the monolithic pattern-matching that is typically required for event merging, especially for copying information from the source events into their combined merger.

Stereotypical pronominal reference

Finally, *Alembic* exploits the inference mechanism to encode some patterns of pronominal reference that are so stereotypical as to be effectively frozen. One such pattern appears in the first sentence: "Bridgestone Sports Co. said Friday *it* will set up a joint venture." This pattern is handled by the following simple rule:

(declare (x y)) & (event (z) y) & (pronominal-thing z) → (= (x z))

This rule matches sentences headed by propositional statement verbs (through the declare term). When the embedded clause (matched through the event term[3]) has a pronominal subject (the pronominal-thing term), the rule equates the subject arguments of the two clauses. As with definite reference, the equality mechanism then propagates facts known about the matrix subject to the embedded subject—in this case the fact that it is a company, *etc.*

SYNOPSIS OF EVALUATION MEASURES

We believe the development effort we undertook in this past year has paid off to a large extent, though not as much as we had originally hoped. We achieved an overall ERR score of 87, corresponding to a P&R score of 22. Although this is on the low end of scores reported for MUC-5, it represents a significant improvement over our MUC-4 performance (P&R of 9 on TST3), especially given the increased difficulty of the MUC-5 task.

More important perhaps than these single performance measures is our overall performance profile. *Alembic* is a clean system: we have among the highest precision scores to be presented here (P=58). In contrast, our performance was hampered by our relatively lower recall (R=14). This high-precision profile is also reflected in our comparatively better system-by-system ranking in the richness-normalized-error measures (minerr=.89 maxerr=.92) than in the primary error per response fill measure. This is due to the fact that *Alembic* avoids generating spurious fills, a characteristic that is favored by the richness-normalized error measures.

Turning to our recall score, our major problem in obtaining a high recall level was simply an inadequate domain model, especially an insufficient complement of extraction rules for locating joint venture entities. For example, we failed to exploit such reliable indicators of these entities as ownership or partnership events. Since entities

[3]In our domain model, event is a generalization that matches any relation, including the create-entity event in the example.

CONDITION	F-SCORE (P&R)	RECALL	PRECISION	ERROR
Full System	23.73	14	60	86
No OAL	24.27	15	61	85
No Reference	23.44	14	60	86

Table 2: Ablation experiments

contribute overwhelmingly to the overall complement of fills, we were correspondingly penalized. In addition, we experienced in the final run a number of hitherto unobserved bugs in the entity fill normalization code; these errors contributed to lowering both our recall and precision scores.

Beyond the official final runs, we conducted several ablation experiments in an effort to gain additional insight into the nature of our performance at MUC-5. In particular, we created versions of *Alembic* in which we selectively disabled (1) the secondary lexicon (OAL), or (2) definite reference resolution. We then compared the performance of these derivative systems to the baseline performance of the full system on the 86-message section of the dry run test set. These results are summarized in Table 2, and as can be readily seen, there are no substantial differences between the experimental conditions. We were somewhat surprised to note a slight performance improvement when we disabled the secondary OAL lexicon, which attests to a certain degree to the robustness of our fallback strategy of mapping unknown lexical items to the UNK-N category.

One should refrain, however, from drawing any sweeping conclusions from these experiments. Indeed, the baseline recall level of *Alembic* on this task is still fairly low, so the effects of these experimental manipulations are probably masked by the lack of sufficient baseline recall. For example, if *Alembic* fails to identify a company as a joint venture participant, it does not matter whether the system later succeeds through reference resolution at determining the company's nationality or location.

Finally, we performed an additional experiment in which we hand-tagged all geographical entities, corporations, and personal names, rather than leaving this task to the elementizer and parser. The intention here was to provide an oracle for these phrase types, so as to help us determine how much of our recall error might have been due to inadequacies in the parser or elementizer, especially with respect to parsing the names of joint venture participants. Once again, we found no significant difference between the baseline system and the oracle-driven system. This substantiates our anecdotal impressions that an inadequate extraction model is the main cause of our recall errors.

FUTURE WORK

Alembic is still on the early end of its maturation curve. Our grammar, for example, was undergoing constant development up until the final runs—unlike many other MUC sites, we did not enter this endeavor with anything remotely close to a general grammar of English, or even of the so-called noun and verb groups. Throughout this time period, basic system development was thus responsible for stealing precious time away from domain engineering.

One benefit of all our MUC-5 development, is that *Alembic* has reached a reasonably stable state. We now have a flexible framework within which to explore data extraction tasks such as EJV in MUC-5. We are especially encouraged by our recall/precision profile over the last few weeks of development. Our daily tests revealed a consistently high precision level that remained stable even with daily improvements in recall.

We are also pleased by our preliminary experiences with the new inference component. Among MUC-5 participants, *Alembic* is distinguished by its ability to accommodate strategies that place a high premium on full parsing (the lexical inference rules) alongside strategies more geared towards fragment parsing (the event merging rules). We look forward to exploiting this new flexibility as we continue informal self-evaluation on the MUC-5 task, and as we explore future data extraction problems.

Acknowledgments

Principal funding for this work was provided by the MITRE Corporation. We would like to thank Lynette Hirschman for her ideas and feedback, and welcome her as the newest contributor to this project. Finally, we would also like to express our gratitude to Richard Tong and his colleagues at ADS for providing us with their MUC-4 lexicon and taxonomies, parts of which are still in place in our current system.

REFERENCES

Aberdeen, J., Burger, J., Connolly, D., Roberts, S., & Vilain, M. (1992). "MITRE-Bedford: Description of the Alembic system as used for MUC-4". In Sundheim, B., ed., *Prcdgs. of the Fourth Message Understanding Conf. (MUC-4)*, McLean, VA.

Alshawi, H.& Van Eijck, J. (1989). "Logical forms in the core language engine". In *Prcdgs. of the 27th Annual Mtg. of the Assoc. for Computational Linguistics (ACL89)*, Vancouver, BC.

Appelt, D. E., Hobbs, J. R., Bear, J., Israel, D., & Tyson, M. (1993). "FASTUS: A finite-state processor for information extraction from real-world text." In *Prcdgs. of the 13th Intl. Joint Conf. on Aritificial Intelligence (IJCAI93)*, Chambéry, France.

Bayer, S. L. & Vilain, M. B. (1991). "The relation-based knowledge representation of King Kong". *Sigart Bulletin* 2(3), 15-21.

Goldfarb, C. F. (1990) *The SGML Handbook*. Oxford: Clarendon Press.

Hobbs, J. R. (1985). "Ontological promiscuity". In *Prcdgs. of the 23rd Annual Mtg. of the Assoc. for Computational Linguistics (ACL85)*, Chicago, IL.

Hobbs, J. R. (1993). "The generic information extraction system." In Sundheim, B., ed., *Prcdgs. of the Fifth Message Understanding Conf. (MUC-5)*, Baltimore, MD.

Jacobs, P. S., Krupka, G., & Rau, L. (1991). "Lexico-semantic pattern-matching as a companion to parsing". In *Prcdgs. of the Fourth DARPA Speech and Natural Language Wkshp.* San Mateo, CA: Morgan Kaufman.

Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E., & Soderland, S. (1992). "University of Massachusetts: Description of the CIRCUS system as used for MUC-4". In Sundheim, B., ed. *Prcdgs. of the Fourth Message Understanding Conf..(MUC-4)*, McLean, VA.

McAllester, D. A. (1989). *Ontic: a knowledge representation system for mathematics*. Cambridge, MA: MIT Press.

Pareschi, R. & Steedman, M. (1987). "A lazy way to chart-parse with categorial grammars." In *Prcdgs. of the 25th Annual Mtg. of the Assoc. for Computational Linguistics (ACL87)*, Stanford, CA.

Shieber, S. (1986) *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes N° 4. Available through the University of Chicago Press.

Steedman, M. (1985) "Dependency and coördination in the grammar of Dutch and English," *Language 61.3*.

Steedman, M. (1987). "Combinatory grammars and parasitic gaps," *Natural Language and Linguistic Theory 5.3*.

Vilain, M. (1991). "Deduction as parsing: Tractable classification in the KL-ONE framework". In *Prcdgs. of the Ninth Natl. Conf. on Artificial Intelligence (AAAI91)*. Anaheim, CA.

Vilain, M. (1993). Validation of Terminological Inference in an information extraction task. In *Prcdgs. of the 1993 ARPA Human Language Wkshp.* San Mateo, CA: Morgan Kaufman.

Wittenburg, K. (1987). "Predictive combinators: A method for efficient processing of combinatory categorial grammars," *Prcdgs. of the 25th Annual Mtg. of the Assoc. for Computational Linguistics (ACL87)*, Stanford, CA.