MITRE: DESCRIPTION OF THE *ALEMBIC* SYSTEM USED FOR MUC-6

John Aberdeen, John Burger, David Day, Lynette Hirschman, Patricia Robinson, and Marc Vilain

The MITRE Corporation
202 Burlington Rd.
Bedford, MA 01730

{aberdeen, john, day, lynette, parann, mbv}@mitre.org

As with several other veteran MUC participants, MITRE's *Alembic* system has undergone a major transformation in the past two years. The genesis of this transformation occurred during a dinner conversation at the last MUC conference, MUC-5. At that time, several of us reluctantly admitted that our major impediment towards improved performance was reliance on then-standard linguistic models of syntax. We knew we would need an alternative to traditional linguistic grammars, even to the somewhat non-traditional categorial pseudo-parser we had in place at the time. The problem was, which alternative?

The answer came in the form of rule sequences, an approach Eric Brill originally laid out in his work on part-of-speech tagging [5, 7]. Rule sequences now underlie all the major processing steps in *Alembic*: part-of-speech tagging, syntactic analysis, inference, and even some of the set-fill processing in the Template Element task (TE). We have found this approach to provide almost an embarrassment of advantages, speed and accuracy being the most externally visible benefits. In addition, most of our rule sequence processors are trainable, typically from small samples. The rules acquired in this way also have the characteristic that they allow one to readily mix hand-crafted and machine-learned elements. We have exploited this opportunity to apply both machine-learned and hand-crafted rules extensively, choosing in some instances to run sequences that were primarily machine-learned, and in other cases to run sequences that were entirely crafted by hand.

ALEMBIC'S OVERALL ARCHITECTURE

For all the changes that the system has undergone, the coarse architecture of the MUC-6 version of *Alembic* is remarkably close to that of its predecessors. As illustrated in Fig. 1, below, processing is still divided into three main steps: a UNIX- and C-based preprocess, a Lisp-based syntactic analysis, and a Lisp-based inference phase. Beyond these coarse-grain similarities, the system diverges significantly from earlier incarnations. We replaced our categorial grammar pseudo-parser, as suggested above. We also redesigned the preprocess from the ground up. Only the inferential back end of the system is largely unchanged.

The internal module-by-module architecture of the current *Alembic* is illustrated in Fig. 2, below. The central innovation in the system is its approach to syntactic analysis, which is now performed through a sequence of phrase-finding rules that are processed by a simple interpreter. The interpreter has somewhat less recognition power than a finite-state machine, and operates by successively relabeling the input according to the rule actions—more on this below. In support of the syntactic phrase finder, or phraser as we call it, the input text must be tagged for part-of-speech. This part-of-speech tagging is the principal role of the UNIX preprocess, and it is itself supported by a number of pretaggers (*e.g.*, for labeling dates and title words) and zoners (*e.g.*, for word tokenization, sentence boundary determination and headline segmentation).

The phrases that are parsed by the phraser are subsequently mapped to facts in the inferential database, a mapping mediated by a simple semantic interpreter. We then exploit inference to instantiate domain

$$\begin{tikzcd}
\text{UNIX preprocess} \arrow[r, Rightarrow] & \text{Syntactic analysis (Lisp)} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{Tractable inference (Lisp)} \arrow[d, Rightarrow] \arrow[dr, Rightarrow] & \\
& \text{NE markup} & \text{TE templates} & \text{ST templates}
\end{tikzcd}$$

Figure 1: Coarse-grained system architecture.

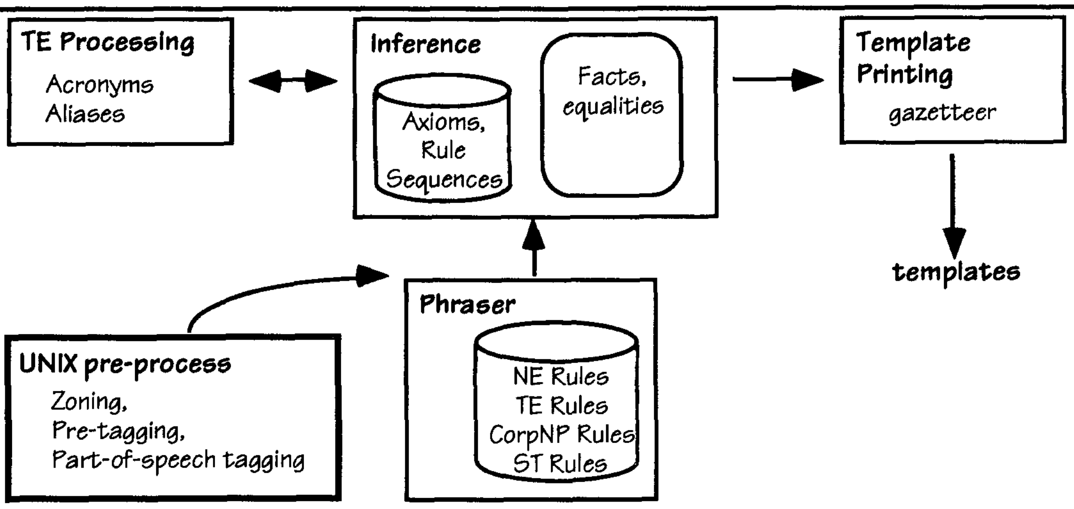

Figure 2: Processing modules in *Alembic.*

constraints and resolve restricted classes of coreference. The inference system also supports equality reasoning by congruence closure, and this equality machinery is in turn exploited to perform TE-specific processing, in particular, acronym and alias merging. Finally, the template generation module forms the final TE and ST output by a roughly one-to-one mapping from facts in the inferential database to templates.

THE PREPROCESSORS

As noted above, the UNIX-based portion of the system is primarily responsible for part-of-speech tagging. Prior to the part-of-speech tagger, however, a text to be processed by Alembic passes through several preprocess stages; each preprocessor "enriches" the text by means of SGML tags. All of these preprocess components are implemented with LEX (the lexical analyzer generator) and are very fast.

An initial preprocessor, the *punctoker*, makes decisions about word boundaries that are not coincident with whitespace. It tokenizes abbreviations (*e.g.*, "Dr."), and decides when sequences of punctuation and alphabetic characters are to be broken up into several lexemes (*e.g.*, "Singapore-based"). The punctoker wraps <LEX> tags around text where necessary to indicate its decisions, as in the following:

```
Singapore<LEX pos=JJ>-based</LEX>
```

As this example suggests, in some cases, the punctoker guides subsequent part-of-speech tagging by adding a part-of-speech attribute to the <LEX> tags that it emits.

The *parasenter* zones text for paragraph and sentence boundaries, the former being unnecessary for MUC-6. The sentence tagging component is both simple and conservative. If any end-of-sentence punctuation has not been "explained" by the punctoker as part of a lexeme, as in abbreviations, it is taken to indicate a sentence boundary. The parasenter is also intended to filter lines in the text body that begin with "@" (but see our error analysis below). A separate *hl-tagger* is invoked to zone sentence-like constructs in the headline field.

The preprocess includes specialized phrase taggers. The *title-tagger* marks personal titles, making distinctions along the lines drawn by the NE and ST tasks. Included are personal honorifics (Dr., Ms.); military and religious titles (Vicar, Sgt.); corporate posts (CEO, chairman); and "profession" words (analyst, spokesperson).

The *date-tagger* identifies TIMEX phrases. It uses a lex-based scanner as a front-end for tokenizing and typing its input; then a pattern-matching engine finds the actual date phrases. The date-tagger is fast, since the pattern matcher itself is highly optimized, and since the lex-based front-end does not actually tokenize the input or fire the pattern-matcher unless it suspects that a date phrase may be occurring in the text.

Both the *date-* and *title-tagger* can tag a phrase as either (1) a single SGML element, or (2) individual lexemes, with special attributes that indicate the beginning and end of the matrix phrase, as in

```
<LEX post=start>chief</LEX > <LEX post=mid>executive</LEX > <LEX post=end>officer</LEX >
```

We adopted this LEX-based phrase encoding so as to simplify (and speed up) the input scanner of the part-of-speech tagger. In addition, a phrase's LEX tags can encode parts-of-speech to help guide the p-o-s tagger.

THE PART-OF-SPEECH TAGGER

Our part-of-speech tagger is closest among the components of our MUC-6 system to Brill's original work on rule sequences [5, 6, 7]. The tagger is in fact a re-implementation of Brill's widely-disseminated system, with various speed and maintainability improvements. Most of the rule sequences that drive the tagger were automatically learned from hand-tagged corpora, rather than hand-crafted by human engineers. However, the rules are in a human-understandable form, and thus hand-crafted rules can easily be combined with automatically learned rules, a property which we exploited in the MUC-6 version of *Alembic*.

The tagger operates on text that has been lexicalized through pre-processing. The following, for example, is how a sample walkthrough sentence is passed to the part-of-speech tagger. Note how punctuation has been tokenized, and "Mr." has been identified as a title and assigned the part-of-speech NNP (proper noun).

```
<S>Even so<lex>,</lex> <LEX pos=NNP ttl=WHOLE>Mr.</LEX> Dooner is on the prowl for more creative talent
and is interested in acquiring a hot agency<lex>.</lex></S>
```

The part-of-speech tagger first assigns initial parts-of-speech by consulting a large lexicon. The lexicon maps words to their most frequently occurring tag in the training corpus. Words that do not appear in the lexicon are assigned a default tag of NN (common noun) or NNP (proper noun), depending on capitalization.

For unknown words, after a default tag is assigned, lexical rules apply to improve the initial guess. These rules operate principally by inspecting the morphology of words. For example, an early rule in the lexical rule sequence retags unknown words ending in "ly" with the RB tag (adverb). In the sentence above, the only unknown word ("Dooner") is not subject to retagging by lexical rules; in fact, the default NNP tag assignment is correct. Lexical rules play a larger role when the default tagging lexicon is less complete than our own, which we generated from the whole Brown Corpus plus 3 million words of Wall Street Journal text. For example, in our experiments tagging Spanish texts (for which we had much smaller lexica), we have found that lexical rules play a larger role (this can also be partially attributed to the more inflected nature of Spanish).

After the initial tagging, contextual rules apply in an attempt to further fix errors in the tagging. These rules reassign a word's tag on the basis of neighboring words and their tags. In this sentence, "more" changes from its initial JJR (comparative adjective) to RBR (comparative adverb). Note that this change is arguably erroneous, depending on how one reads the scope of "more". This tagging is changed by the following rule, which roughly reads: change word W from JJR to RBR if the the word to W's immediate right is tagged JJ

```
JJR RBR nexttag JJ
```

Table 1, below, illustrates the tagging process. The sample sentence is on the first line; its initial lexicon-based tagging is on the second line; the third line shows the final tagging produced by the contextual rules.

In controlled experiments, we measured the tagger's accuracy on Wall Street Journal text at 95.1% based on a training set of 140,000 words. The production version of the tagger, which we used for MUC-6, relies on the

Even	so	,	Mr.	Dooner	is	on	the	prowl	for	more	creative	talent	and	is	interested	in	acquiring	a	hot	agency
rb	rb	,	nnp	NNP	vbz	in	dt	nn	in	JJR	jj	nn	cc	vbz	jj	in	vbg	dt	jj	nn
rb	rb	,	nnp	NNP	vbz	in	dt	nn	in	RBR	jj	nn	cc	vbz	jj	in	vbg	dt	jj	nn

Table 1: Tagging a text with the lexicon (line 2) and contextual rules (line 3). Note the default lexicon assignment of nnp to "Dooner" and the rule-based correction of "more".

learned rules from Brill's release 1.1 (148 lexical rules, 283 contextual rules), for which Brill has measured accuracies that are 2-3 percentage points higher than in our own smaller-scale experiments. For MUC-6, we combined these rules with 19 hand-crafted contextual rules that correct residual tagging errors that were especially detrimental to our NE performance. Tagger throughput is around 3000 words/sec.

THE PHRASER

The *Alembic* phrase finder, or phraser for short, performs the bulk of the system's syntactic analysis. As noted above, it has somewhat less recognition power than a finite-state machine, and as such shares many characteristics of pattern-matching systems, such as CIRCUS [10] or FASTUS [2]. Where it differs from these systems is in being driven by rule sequences. We have experimented with both automatically-learned rule sequences and hand-crafted ones. In the system we fielded for MUC-6, we ended up running entirely with hand-crafted sequences, as they outperformed the automatically-learned rules.

How the phraser works

The phraser process operates in several steps. First, a set of initial phrasing functions is applied to all of the sentences to be analyzed. These functions are responsible for seeding the sentences with likely candidate phrases of various kinds. This seeding process is driven by word lists, part-of-speech information, and pre-taggings provided by the preprocessors. Initial phrasing produces a number of phrase structures, many of which have the initial null labeling (none), while some have been assigned an initial label (*e.g.,* num). The following example shows a sample sentence from the walkthrough message after initial phrasing.

Yesterday, <none>McCann</none> made official what had been widely anticipated: <ttl>Mr.</ttl> <none>James</none>, <num>57</num> years old, is stepping down as <post>chief executive officer</post> on <date>July 1</date> and will retire as <post>chairman</post> at the end of the year.

The post, ttl, and date phrases were identified by the title and date taggers. Mr. James' num-tagged age is identified on the basis of part of speech information, as is the organization name "McCann".

Once the initial phrasing has taken place, the phraser proceeds with phrase identification proper. This is driven by a sequence of phrase-finding rules. Each rule in the sequence is applied in turn against all of the phrases in all the sentences under analysis. If the antecedents of the rule are satisfied by a phrase, then the action indicated by the rule is executed immediately. The action can either change the label of the satisfying phrase, grow its boundaries, or create new phrases. After the *n*th rule is applied in this way against every phrase in all the sentences, the *n+1*th rule is applied in the same way, until all rules have been applied. After all of the rules have been applied, the phraser is done.

It is important to note that the search strategy in the phraser differs significantly from that in standard parsers. In standard parsing, one searches for any and all rules whose antecedents might apply given the state of the parser's chart: all these rules become candidates for application, and indeed they all are applied (modulo higher-order search control). In our phraser, only the *current* rule in a rule sequence is tested: the rule is applied wherever this test succeeds, and the rule is never revisited at any subsequent stage of processing. After the final rule of a sequence is run, no further processing occurs.

The language of phraser rules

The language of the phraser rules is as simple as their control strategy. Rules can test lexemes to the left and right of the phrase, or they can look at the lexemes in the phrase. Tests in turn can be part-of-speech queries, literal lexeme matches, tests for the presence of neighboring phrases, or the application of predicates that are evaluated by invoking a Lisp procedure. There are several reasons for keeping this rule language simple. In the case of hand-crafted rules, it facilitates the process of designing a rule sequence. In the case of machine-learned rules, it restricts the size of the search space on each epoch of the learning regimen, thus making it tractable. In either case, the overall processing power derives as much from the fact that the rules are sequenced, and feed each other in turn, as it does from the expressiveness of the rule language.

To make this clearer, consider a simple named entity rule that is applied to identifying persons.

```
(def-phraser
   label          none
   left-1         phrase ttl
   label-action   person)
```

This rule changes the label of a phrase from none to person if the phrase is bordered on its left by a ttl phrase. On the sample sentence, this rule causes the following relabeling of the phrase around "James".

Yesterday, <none>McCann</none> made official what had been widely anticipated: <ttl>Mr.</ttl> <person>James</person>, <num>57</num> years old, is stepping down as <post>chief executive officer</post> on <date>July 1</date> and will retire as <post>chairman</post> at the end of the year.

Once this rule has run, the labelings it instantiates become available as input to subsequent rules in the sequence, *e.g.*, rules that attach the title to the person in "Mr. James", that attach the age apposition, and so forth. Phraser rules do make mistakes, but as with other sequence-based processors, the phraser applies later rules in a sequence to patch errors made by earlier rules. In the walkthrough message, for example, "Amarati & Puris" is identified as an organization, which ultimately leads to an incorrect org tag for "Martin Puris", since this person's name shares a common substring with the organization name. However, rules that find personal names occur later in our named entities sequence than those which find organizations, thus allowing the phraser to correctly relabel "Martin Puris" as a person on the basis of a test for common first names.

Rule sequences for MUC-6

For MUC-6, *Alembic* relies on three sequences of phraser rules, divided roughly into rules for generating NE-specific phrases, those for finding TE-related phrases, and those for ST phrases. The division is only rough, as the NE sequence yields some number of TE-related phrases as a side-effect of searching for named entities.

To illustrate this process, consider the following walkthrough sentence, as tagged by the NE rule sequence.

But the bragging rights to <org>Coke</org>'s ubiquitous advertising belongs to <org>Creative Artists Agency </org>, the big <location>Hollywood</location> talent agency.

The org label on "Creative Artists Agency" was set by a predicate that tests for org keywords (like "Agency"). "Coke" was found to be an org elsewhere in the document, and the label was then percolated. Finally, the location label on "Hollywood" was set by a predicate that inspects the tried-and-not-so-true TIPSTER gazetteer.

What is important to note about these NE phraser rules is that they do not rely on a large database of known company names. Instead, the rules are designed to recognize organization names in almost complete absence of any information about particular organization names (with the sole exception of a few acronyms such as IBM, GM, *etc.*). This seems to auger well for the ability to apply *Alembic* to different application tasks.

Proceeding beyond named entities, the phraser next applies its TE-specific rule sequence. This sequence performs manipulations that resemble NP parsing, *e.g.*, attaching locational modifiers. In addition, a subsequence of TE rules concentrates on recognizing potential organization descriptors. These rules generate so-called corpnp phrases, that is noun phrases that are headed by an organizational common noun (such as "agency", "maker", and of course "company"). The rules expand these heads leftwards to incorporate lexemes that satisfy a set of part-of-speech constraints. One such phrase, for example, is in the sample sentence above.

But the bragging rights to <org>Coke</org>'s ubiquitous advertising belongs to <org><org>Creative Artists Agency </org>, <corpnp> the big <location>Hollywood</location> talent agency</corpnp></org>

After corpnp phrases have been marked, another collection of TE rules associates these phrases with neighboring org phrases. In this case such a phrase is found two places to the left (on the other side of a comma), so a new org phrase is created which spans both the original org phrase and its corpnp neighbor. See above.

Note that these rule sequences encode a semantic grammar. Organizationally-headed noun phrases are labeled as org, regardless of whether they are simple proper names or more complex constituents such as the

```
 * * * TOTAL SLOT SCORES * * *
-----------------+---------------+------------+-----------------------
SLOT     POS    ACT|  COR   PAR   INC| SPU MIS NON| REC PRE UND OVG ERR SUB
-----------------+---------------+------------+-----------------------
<enamex>938    991|  881    0      0| 110  57   0|  94  89   6  11  16   0
 type    938    991|  775    0    106| 110  57   0|  83  78   6  11  26  12
 text    938    991|  840    0     41| 110  57   0|  90  85   6  11  20   5
 subto  1876   1982| 1615    0    147| 220 114   0|  86  81   6  11  23   8
-----------------+---------------+------------+-----------------------
ALL OB  2286   2406| 1993    0    163| 250 130   0|  87  83   6  10  21   8
MATCHD  2156   2156| 1993    0    163|   0   0   0|  92  92   0   0   8   8
-----------------+---------------+------------+-----------------------
                                                 P&R        2P&R       P&2R
F-MEASURES                                       84.95      83.67      86.28

 * * * TASK SUBCATEGORIZATION SCORES * * *
-----------------+---------------+------------+-----------------------
SLOT     POS    ACT|  COR   PAR   INC| SPU MIS NON| REC PRE UND OVG ERR SUB
-----------------+---------------+------------+-----------------------
Enamex:
 organi  454    493|  392    0     28|  73  34   0|  86  80   7  15  26   7
 person  373    364|  292    0     60|  12  21   0|  78  80   6   3  24  17
 locati  111    134|   91    0     18|  25   2   0|  82  68   2  19  33  16
```

Figure 4: Performance of rules learned for the ENAMEX portion of the NE task (unofficial score)

org-corpnp apposition above. This semantic characteristic of the phraser grammar is clearer still with ST rules. These rules are responsible for finding phrases denoting events relevant to the MUC-6 scenario templates.[1] For the succession scenario, this consists of a few key phrase types, most salient among them: job (a post at an org), job-in and job-out (fully specified successions) and post-in and post-out (partially specified successions). The following example shows the ST phrases parsed out of a key sentence from the walkthrough message.

```
Yesterday, <person>McCann</person> made official what had been widely anticipated:
<post-out>
   <person>
      <person><ttl>Mr.</ttl><person>James</person></person>,
      <age><num>57</num> years old</age>
   </person>,
   is stepping down as
   <post>chief executive officer</post>
</post-out> [...]
```

The post-out phrase encodes the resignation of a person in a post. Note that in the formal evaluation we failed to find a more-correct job-out phrase, which should have included "McCann". This happened because we did not successfully identify "McCann" as an organization, thus precluding the formation of the job-out phrase.

Learning Phrase Rules

We have applied the same general error-reduction learning approach that Brill designed for generating part-of-speech rules to the problem of learning phraser rules in support of the NE task. The official version of Alembic for MUC-6 did not use any of the rule sequences generated by this phrase rule learner, but we have since generated unofficial scores. In these runs we used phrase rules that had been learned for the ENAMEX expressions only—we still used the hand-coded pre-processors and phraser rules for recognizing TIMEX and NUMEX phrases. Our performance on this task is shown in Fig. 4, above. These rules yield six fewer points of P&R than the hand-coded ENAMEX rules—still an impressive result for machine-learned rules. Interestingly, the bulk of the additional error in the machine-learned rules is not with the "hard" organization names, but with person names (ΔR=-15, ΔP=-14) and locations (ΔR=-12, ΔP=-18).

[1]We put about one staff week of work into the ST task, during which we experienced steep hill-climbing on the training set. Nevertheless, we felt that the maturity of our ST processing was sufficiently questionable to preclude participating in the official evaluation. The present discussion should be taken in this light, *i.e.*, with the understanding that it was not officially evaluated at MUC-6.

PHRASE INTERPRETATION AND INFERENCE

The inference component is central to all processing beyond phrase identification. It has three roles.

- As a representational substrate, it records propositions encoding the semantics of parsed phrases;
- As an equational system, it allows initially distinct semantic individuals to be equated to each other, and allows propositions about these individuals to be merged through congruence closure.
- As a limited inference system, it allows domain-specific and general constraints to be instantiated through carefully controlled forward-chaining.

Phrase Interpretation

Facts enter the propositional database as the result of phrase interpretation. The phrase interpreter is controlled by a small set of Lisp interpretation functions, roughly one for each phrase type. Base-level phrases, i.e. phrases with no embedded phrases, are mapped to unary interpretations. The phrase

```
<person>Robert L. James</person>,
```

for example is mapped to the following propositional fact. Note the pers-01 term in this proposition: it designates the semantic individual denoted by the phrase, and is generated in the process of interpretation.

```
person(pers-01)
```

Complex phrases, those with embedded phrases, are typically interpreted as conjuncts of simpler interpretations (the exception being NP coordination, as in "chairman and chief executive"). Consider the phrase "Mr. James, 57 years old" which is parsed by the phraser as follows. Note in particular that the overall person-age apposition is itself parsed as a person phrase.

```
<person><person>Mr. James</person>, <age><num>57</num> years old</age></person>
```

The treatment of age appositions is compositional, as is the case for the interpretation of all but a few complex phrases. Once again, the embedded base-level phrase ends up interpreted as a unary person fact. The semantic account of the overall apposition ends up as a has-age relation modifying pers-02, the semantic individual for the embedded person phrase. This proposition designates the semantic relationship between a person and that person's age. More precisely, the following facts are added to the inferential database.

```
person(pers-02)
has-age((pers-02, age-03) ha-04)
age(age-03)
```

What appears to be a spare argument to the has-age predicate above is the event individual for the predicate. Such arguments denote events themselves (in this case the event of being a particular number of years old), as opposed to the individuals participating in the events (the individual and his or her age). This treatment is similar to the partial Davidsonian analysis of events due to Hobbs [8]. Note that event individuals are by definition only associated with relations, not unary predicates.

As a point of clarification, note that the inference system does not encode facts at the predicate calculus level so much as at the interpretation level made popular in such systems as the SRI core language engine [1, 3]. In other words, the representation is actually a structured attribute-value graph such as the following, which encodes the age apposition above.

```
[[head :person]
 [proxy pers-02]
 [modifiers [[head has-age]
             [proxy ha-04]
             [arguments (pers-02 [[head age]
                                  [proxy age-03]])]]]]
```

The first two fields correspond to the embedded phrase: the head field is a semantic sort, and the proxy field holds the designator for the semantic individual denoted by the phrase. The interpretation encoding the overall apposition ends up in the *modifiers* slot, an approach adopted from the standard linguistic account of phrase modification. Inference in *Alembic* is actually performed directly on interpretation structures, and there is no need for a separate translation from interpretations to more traditional-looking propositions. The propositional notation is more perspicuous to the reader, and we have adopted it here.

Finally, note that the phrase interpretation machinery maintains pointers between semantic individuals and the surface strings from which they originated. One of the fortunate—if unexpected—consequences of the phraser's semantic grammar is that maintaining these cross-references is considerably simpler than was the case in our more linguistically-inspired categorial parser of old. Except for the ORG_DESCRIPTOR slot, the fill rules line up more readily with semantic notions than with syntactic considerations, e.g., maximal projections.

Equality reasoning

Much of the strength of this inferential framework derives from its equality mechanism. This subcomponent allows one to make two semantic individuals co-designating, *i.e.*, to "equate" them. Facts that formerly held of only one individual are then copied to its co-designating siblings. This in turn enables inference that may have been previously inhibited because the necessary antecedents were distributed over (what were then) distinct individuals.

This equality machinery is exploited at many levels in processing semantic and domain constraints. One of the clearest such uses is in enforcing the semantics of coreference, either definite reference or appositional coreference. Take for example the following phrase from the walkthrough message, which we show here as parsed by the phraser.

```
<org>
      <org>Creative Artists Agency</org>,
      <orgnp>the big <location>Hollywood</location> talent agency</orgnp>
</org>
```

In propositional terms, the embedded organization is interpreted as

```
organization(org-05)                    ; "Creative Artists Agency"
```

The appositive noun phrase is interpreted as

```
organization(org-06)                    ; "the...agency"
geo-region(geo-07)                      ; "Hollywood"
has-location((org-06, geo-07) hasloc-08)  ; locational pre-modifier
```

Pressing on, the phraser parses the overall *org-orgnp* apposition as an overarching *org*. To interpret the apposition, the interpreter also adds the following proposition to the database.

```
entity-np-app((org-05, org-06) e-n-a-09)
```

This ultimately causes *org-05* and *org-06* to become co-designating through the equality system, and the following fact appears in the inferential database.

```
has-location((org-05, geo-07) hasloc-10)   ; i.e., Creative Artists Agency is located in Hollywood
```

This propagation of facts from one individual to its co-designating siblings is the heart of our coreference mechanism. Its repercussions are particularly critical to the subsequent stage of template generation. By propagating facts in this way, we can dramatically simplify the process of collating information into templates, since all the information relevant to, say, an individual company will have been attached to that company by equality reasoning. We will touch on this point again below.

Inference

The final role of the *Alembic* inference component is to derive new facts through the application of carefully-controlled forward inference. As was the case with our MUC-5 system, the present *Alembic* allows only limited forward inference. Though the full details of this inference process are of legitimate interest in their own right, we will only note some highlights here. To begin with, the tractability of forward inference in this framework is guaranteed just in case the inference axioms meet a certain syntactic requirement. To date, all the rules we have written for even complex domains, such as the joint-venture task in MUC-5, have met this criterion. Aside from this theoretical bound on computation, we have found in practice that the inference system is remarkably fast, with semantic interpretation, equality reasoning, rule application, and all other aspects of inference together accounting for 6-7% of all processing time in *Alembic*. Details are in [11].

We exploited inference rules in several primary ways for the TE and ST tasks. The first class of inference rules enforce so-called terminological reasoning, local inference that composes the meaning of words. One such rule distributes the meaning of certain adjectives such as "retired" across coordinated titles, as in "retired chairman and CEO". The phrase parses as follows; note the embedded post semantic phrase types.

```
<post>
    <post-qual>retired</post-qual>
    <post>
        <post>chairman</post>
        and
        <post>CEO</post>
    </post>
</post>
```

This particular example propositionalizes as follows, where the group construct denotes a plural individual in Landman's sense (roughly a set [9]).

```
title(ttl-11)                  ; "chairman"
title(ttl-12)                  ; "CEO"
group((ttl-11, ttl-12) grp-13) ; "chairman and CEO"
retired-ttl(grp-13)            ; "retired"
```

To shift the scope of "retired" from the overall coordination to individual titles, the following rule applies.

```
retired-ttl(ttl) <— group((ttl, x) grp) + retired(grp)
```

This rule yields the fact retired-ttl(ttl-11), and a similar rule yields retired-ttl(ttl-12). Other like rules distribute coordinated titles across the title-holder, and so forth. The fact that ***multiple*** rules are needed to distribute adjectives over coordinated noun phrases is one of the drawbacks of semantic grammars. On the other hand, these rules simplify semantic characteristics of distributivity by deferring questions of scope and non-compositionality to a later stage, *i.e.*, inference. Interpretation procedures can thus remain compositional, which makes them substantially simpler to write. Additionally, these kinds of distribution rules further contribute to collating facts relevant to template generation onto the individuals for which these facts hold.

Of greater importance, however, is the fact that inference rules are the mechanism by which we instantiate domain-specific constraints and set up the particulars required for scenario-level templates. Some of this information is again gained by fairly straightforward compositional means. For example, the phrase "Walter Rawleigh Jr., retired chairman of Mobil Corp." yields a succession template through the mediation of one inference rule. The phrase is compositionally interpreted as

```
organization(org-14)
title(ttl-15)
retired-ttl(ttl-15)
job((ttl-15, org-14), job-16)
person(pers-17)
holds-job((pers-17, job-16) h-j-18)
```

The rule that maps these propositions to a succession event is

```
job-out(pers, ttl, org) <— holds-job((pers, job) x) + job((ttl, org), job) + retired-ttl(ttl)
```

When applied to the above propositions this rule yields job-out((pers-17, ttl-15, org-07) j-o-19). This fact is all that is required for the template generator to subsequently issue the appropriate succession event templates.

The most interesting form of domain inference is not compositional of course, but based on discourse considerations. In the present ST task, for example, succession events are not always fully fleshed out, but depend for their complete interpretation on information provided earlier in the discourse. In the walkthrough message, this kind of contextualized interpretation is required early on:

Yesterday, McCann made official what had been widely anticipated: Mr. James, 57 years old, is stepping down as chief executive officer on July 1 [...]. He will be succeeded by Mr. Dooner, 45.

ST-level phrasing and interpretation of this passage produces two relevant facts, a job-out for the first clause, and a successor for the second. Note that although successor is normally a two-place relation, its valence here is one by virtue of the phraser not finding a named person as a subject to the clause.

```
person(pers-20)                        ; "Mr. James"
title(ttl-21)                          ; "chairman"
organization(org-22)                   ; "McCann"
job-out((pers-20, ttl-21, org-22) j-o-23)

person(pers-24)                        ; "Mr. Dooner"
successor((pers-24) succ-25)
```

One approach to contextualizing the succession clause in this text would require first resolving the pronominal subject "He" to "Mr. James" and then exploiting any job change facts that held about this antecedent. An equally effective approach, and simpler, is to ignore the pronoun and reason directly from the successor fact to any contextualizing job-out fact. The rule that accomplishes this is

```
job-in(pers-a, ttl, org) <— successor((pers-a) succ) + job-out-in-context?((succ, job-out) x-1) +
                            job-out((pers-b, ttl, org) x-2)
```

The mysterious-looking job-out-in-context? predicate implements a simple discourse model: it is true just in case its second argument is the most immediate job-out fact in the context of its first argument. Context-encoding facts are not explicitly present in the database, as their numbers would be legion, but are instantiated "on demand" when a rule attempts to match such a fact. Note that what counts as a most immediate contextualized fact is itself determined by a separate search procedure. The simple-minded strategy we adopted here is to proceed backwards from the current sentence, searching for the most recent sentence containing an occurrence of a job-out phrase, and returning the semantic individual it denotes. In this example, the job-out-in-context? predicate succeeds by binding the succ variable to j-o-23, with the rule overall yielding a job-in fact.

```
job-in((pers-24, ttl-21, org-22) j-i-26)
```

As with job-out, this fact is mapped directly by the template generator to an incoming succession template.

Note that this process of combining a job-out and successor fact effectively achieves what is often accomplished in data extraction systems by template merging. However, since the merging takes place in the inferential database, with propagation of relevant facts as a side-effect, the process is greatly simplified and obviates the need for explicit template comparisons.

One final wrinkle must be noted. Inference is generally a non-deterministic search problem, with no firm guarantee as to whether facts will be derived in the same chronological order as the sentences which underlie the facts. Rules that require contextualized facts, however, crucially rely on the chronological order of the sentences underlying these facts. We have thus pulled these rules out of the main fray of inference, and apply them only after all other forward chaining is complete. In fact, these rules are organized as a Brill-style rule sequence, where each rule is allowed to run to quiescence at only one point in the sequence before the next rule becomes active. It is our hypothesis, though, that *all* domain inference rules can be so organized, not just contextualized ones, and that by this organizational scheme, rules can be automatically learned from example.

TASK-SPECIFIC PROCESSING AND TEMPLATE GENERATION

Aside from phrasing and inference, a relatively small—but critical—amount of processing is required to perform the MUC-6 named entities and template generation tasks.

For NE, little is actually required beyond careful document management and printing routines. TIMEX forms, introduced by the preprocessing date-tagger, must be preserved through the rest of the processing pipe. Named entity phrases that encode an intermediate stage of NE processing must be suppressed at printout. Examples such as these abound, but by and large, *Alembic*'s NE output is simply a direct readout of the result of running the named entity phraser rules.

Name coreference in TE

Of all three tasks, TE is actually the one that explicitly requires most idiosyncratic processing beyond phrasing and inference. Specifically, this task is the crucible for name coreference, *i.e.*, the process by which short name forms are reconciled with their originating long forms.

This merging process takes place by iterating over the semantic individuals in the inferential database that are of a namable sort (*e.g.*, person or organization). Every such pair of same-sort individuals is compared to determine whether one is a derivative form of the other. Several tests are possible.

- Identity. If the forms are identical strings, as in the frequently repeated "Dooner", or "McCann" in the walkthrough article, then they are merged.
- Shortening. If one form is a shortening of the other, as in "Mr. James" for "Robert L. James", then the short form is merged as an alias of the longer.
- Acronyms. If one form appears to be an acronym for the other, as in "CAA" and "Creative Artists Agency", then the forms should be merged, with the acronym designated as an alias.

Merging two forms takes place in several steps. First, their respective semantic individuals are equated in the inferential database. This allows facts associated with one form to become propagated to the other. In this way, the nationality information in "Japanese giant NEC" becomes associated with the canonical name "Nippon Electric Corp." As a second step, names that are designated as aliases are recorded as such.

Template generation

We mentioned above that the inferential architecture that we have adopted here is in good part motivated by a desire to simplify template generation. Indeed, template generation consists of nothing more than reading out the relevant propositions from the database.

For the TE task, this means identifying person and organization individuals by matching on person(x) or organization(y). For each so-matching semantic individual, we create a skeletal template. The skeleton is initialized with name and alias strings that were attached to the semantic individuals during name merging. It is further fleshed out by looking up related facts that hold of the matched individual, *e.g.*, has-location(y, z) for organizations or has-title(x, w) for persons. These facts are literally just read out of the database.

Finalization routines are then invoked on the near-final template to fill the ORG_TYPE slot and to normalize the geographical fills of the ORG_LOCALE and ORG_COUNTRY slots.

PERFORMANCE ANALYSIS

We participated in two officially-scored tasks at MUC-6, named entities and template elements. As noted above, we put roughly a staff week into customizing the system to handle the scenario templates task, but chose not to participate in the evaluation because another staff week or so would have been required to achieve performance on a par with other parts of the system.

Overall performance

On the named entity task, we obtained an official P&R score of 91.2, where the separate precision and recall scores were both officially reported to be 91. The overall score is remarkably close to our performance on the

	Formal training		Official test			
	Recall	Precision	Recall	Precision	Recall Δ	Precis. Δ
organization	86	92	84	92	-2	—
name	76	78	77	80	+1	+2
alias	60	79	56	78	-4*	-1
descriptor	27	62	16	49	-11*	-13*
type	83	90	81	89	-2	-1
locale	46	87	43	87	-3	—
country	47	88	45	93	-2	+5
person	94	92	95	87	+1	-5*
name	93	91	93	84	—	-7*
alias	94	95	86	96	-8*	+1
title	95	96	94	93	-1	-3
All Objects	*75*	*86*	*73*	*85*	*-2*	-1
F-Measure, unrevised	*80.21*		*78.52*		*-1.69*	

Table 2: Slot-by-slot performance differences, TE task (unrevised scores).

dry-run test set which served as our principal source of data for NE training and self-evaluation. To be precise, our final dry-run P&R score prior to the MUC-6 evaluation run was 91.8, a scant 0.6 higher than the officially measured evaluation score. The fact that the score dropped so little is encouraging to us.

On the template elements task, our initial TE score was P&R=78.5, and our revised official score was 77.3. Once again, this performance is encouragingly close to *Alembic's* performance on our final self-evaluation using the formal training data set. By the non-revised metric, we achieved a performance of P&R=80.2 on the training data, with an overall drop of 1.7 points of P&R between training and official test. Table 2 summarizes slot-by-slot differences between our training and test performance on the TE task. The major differences we noted between training and testing performance lie in the organization alias and descriptor slots, and in the person name and alias fields; we have marked these discrepancies with asterisks (*) and will address their cause later on in this document.

Walkthrough errors

In order to quantify *Alembic's* performance on the walkthrough message, we compiled an exhaustive analysis of our errors on this text. This was a difficult message for us, and we scored substantially less well on this message than in average, especially on the NE task. To our surprise, the majority of our errors was due not to knowledge gaps in the named entity tagger, so much as to odd bugs and incompletely thought-out design decisions. Table 3 summarizes these errors. Entries marked with daggers (†) correspond to knowledge gaps, *e.g.*, missing or incorrect rules; the other entries are coding or design problems. Fully half the problem instances were due to string matching issues for short name forms. For example, by not treating embedded mid-word hyphens as white space, we failed to process "McCann" as a shortened form of "McCann-Erickson".

Turning now to the template element task, we note that the largest fraction of TE errors are repercussions of errors committed while performing the NE task. In particular, the people-name companies that were treated as persons during NE processing in turn led to spurious person templates. The magnitude of the NE error is mitigated by the fact that identical mentions of incorrectly-tagged named entities are merged for the sake of TE template generation, and thus do not all individually spawn spurious templates. Among the TE errors not arising from NE processing errors, note in particular those that occurred on the most difficult slots, ORG_DESCRIPTOR, ORG_LOCALE, and ORG_COUNTRY. These are all due in this case to missing locational and

Nature of the problem	Problem cases	Resulting errors
Naive string matching	"McCann" *vs.* "McCann-Erickson"	9 inc type
	"John Dooner" *vs.* "John J. Dooner Jr."	1 inc text, 1 spu type/text
Missing phraser patterns†	"Fallon McElligott" — treated as person	1 inc type
	"Taster's Choice" — naive 's processing	1 spu type/text
Poor phraser patterns†	"Coca-Cola Classic" — zealous org rule	1 spu type/text
Missing date patterns†	"the 21st century"	1 mis type/text
Ambiguous name	"New York Times" — not an org	1 spu type/text
Misc. embarrassing bugs	"James" in <HL> — treated as location	1 inc. type
	"J. Walter Thompson" — punctoker lost "J."	1 inc type, 1 inc text

Table 3: NE errors on walkthrough message

Nature of the problem	Problem cases	Resulting errors
Repercussions of NE errors	"Walter Thompson", "Fallon McElligott", "McCann" all treated as person	3 spu pers. 1 mis org. alias
	"John Dooner" treated as two persons	2 spu pers, 1 mis pers. alias
	"Coca-Cola Classic" treated as organization	1 inc org. name††, 1 inc. org alias
Missing org. NP patterns†	"the agency with billings of $400 million"	2 mis org. descriptor
	"one of the largest world-wide agencies"	
Missing location patterns†	"Coke's headquarters in Atlanta"	1 mis org. locale/country
Org. type determination†	"Creative Artists Agency" — treated as gov.	1 inc. org type
Acronym resolution snafu	"CAA" *vs.* "Creative Artists Agency"	1 inc org. name††, 1 mis org. alias
Enthusiastic scorer mapping	"New York Times" (spurious entity) mapped to "Fallon McElligott" (inc. entity type)	1 inc. org. name††

Table 4: TE errors on walkthrough message

organizational NP phraser rules, which is consistent with trends we noted during training. These observations are summarized in Table 4. Once again, single daggers (†) mark errors attributable to knowledge gaps.

Note also that because of lenient template mappings on the part of the scorer, a number of errors that might intuitively have been whole organization template errors turned out only to manifest themselves as organization name errors. These cases are marked with double daggers (††).

Other trends

In addition to this analysis of the single walkthrough message, we opened up some 10% of the test data to inspection, and performed a rough trend estimation. In particular, we wanted to explain the slot-by-slot discrepancies we had noted between our training and test performance (*cf.* Table 2). We found a combination of knowledge gaps, known design problems that had been left unaddressed by the time of the evaluation run, and some truly embarrassing bugs.

To dispense quickly with the latter, we admit to failing to filter lines beginning with "@" in the body of the message. This was due to the fact that earlier training data had these lines marked with <S> tags, whereas the official test data did not. These @-lines were so rare in the formal training data that we had simply not noticed our omission. This primarily affected our NUMEX and TIMEX precision in the named entity task.

In the template element task, our largest performance drop was on the ORG_DESRIPTOR slot, where we lost 11 points of recall and 13 points of precision. This can be largely attributed to knowledge gaps in our phraser rules for organizational noun phrases. In particular, we were missing a large number of head nouns that would have been required to identify relevant descriptor NPs.

On the PERSON_NAME and PERSON_ALIAS slots, we respectively found a 7 point drop in precision and an 8 point drop in recall. These were due to the same problem, a known flaw that had been left unaddressed in the days leading to the evaluation. In particular, we had failed to merge short name forms that appeared in headlines with the longer forms that appeared in the body of the message. For example, "James" in the walkthrough headline field should have been merged with "Robert L. James" in the body of the message. Because these short forms went unmerged, they in turn spawned incorrect person templates, hence the drop in PERSON and PERSON_NAME precision. For the same reason, the templates that were generated for the long forms of these names ended up without their alias slot filled, accounting for the drop in PERSON_ALIAS recall.

A similar problem held true for the ORG_ALIAS slot. In this case, we failed both to extract organization templates from the headline fields, or merge short name forms from headlines with longer forms in the text bodies. We were aware of these "mis-features" in our handling of person and organization name templates, but had left these problems unaddressed since they seemed to have only minimal impact on the formal training data. *Errare humanum est.*

POST-HOC EXPERIMENTS

With this error analysis behind us, we pursued a number of post-hoc experiments. Most interesting among them was a simple attempt at improving recall on organization names. Indeed, *Alembic* has only a short list of known organizations—less than a half-dozen in total. Virtually all of the organizations found by *Alembic* are recognized from first principles. We decided to compare this strategy with one that uses a large lexicon of organization names.

All of the MUC-6 NE training set was used to generate a list of 1,808 distinct organization name strings. This could certainly be larger, but seemed a reasonable size. Nonetheless, this lexicon by itself got less than half of the organizations in the official named-entity test corpus: organization recall was 45 and precision 91. Another interesting question is how much an organization lexicon might have helped had it been *added* to our rule-based phrasing algorithm, not simply used by itself. This configuration actually decreased our performance slightly (F-score down by 0.5 points of P&R), trading a slight increase in organization recall for a larger decrease in precision. The biggest problem here is due to overgeneration (up from 4% to 6%), and partial matches such as the following,

```
Kraft <ENAMEX>General Foods</ENAMEX>
First <ENAMEX>Fidelity</ENAMEX>
```

where "General Foods" and "Fidelity" were in the training corpus for the organization lexicon, but the longer names above were not.

Admittedly, the way we integrated the organization lexicon into Alembic was relatively naïve, thereby leading to some of these silly precision errors. We believe that if we more intelligently took advantage of this knowledge source, we could reduce the additional precision errors almost entirely. In addition, we were disappointed by the fact that our exhaustive compilation only produced somewhat less than 2,000 organization names, and only led to a piffling improvement in recall. Perhaps had we made use of larger name lists, we might have obtained better recall improvements—a case in point is the gargantuan Dunn & Bradstreet listing exploited by Knight-Ridder for their named entities tagger [4]. Note, however, that all but a few of the organizations that were found in both the training name list and the test data were found by *Alembic* from first principles anyway. We may thus tentatively conclude that short of being as encyclopedic as the D&B listing, a larger, better-integrated organization lexicon may have provided no more than a limited improvement in F-score. To further improve our organization tagging, it appears that we will simply have to expend more energy writing named entity phraser rules.

CONCLUDING THOUGHTS

All in all, we are quite pleased with the performance of *Alembic* in MUC-6. While we regret not participating in the ST task, we do believe that the framework was up to it, especially in light of our TE scores. There were many lessons learned, and there will continue to be, as we further analyze our results and make improvements to the system. Several points stand out. We had hoped to avoid full NP parsing, but the definition of the ORG_DESCRIPTOR slot clearly requires this, and we will need to return to larger-scale parsing strategies in the future. We had hoped to include more machine-learned phraser rules, and as the rule learner matures, we almost certainly will.

One thing is clear to us, however, and that is that rule sequences are an extremely powerful tool. They were easy to hand-craft and adapt to the MUC-6 requirements. They run fast. And they work well.

References

[1] Alshawi, H. & Van Eijck, J. "Logical forms in the core language engine". In *Proceedings of the 27th Meeting of the Assoc.iation for Computational Linguistics* (ACL-89). Vancouver, B.C., 1985.

[2] Appelt, D., Hobbs, J., Bear, J., Israel, D., & Tyson, M. "FASTUS: A finite-state processor for information extraction from real-world text". In *Proceedings of the 13th Intl. Joint Conference on Artificial Intelligence* (IJCAI93). Chambéry, 1993. 1172-1178.

[3] Bayer, S. & Vilain, M. "The relation-based knowledge representation of King Kong". SIGART *Bulletin*, 2(3), 15-21.

[4] Borkovsky, A. "Knight-Ridder's value adding name finder: a variation on the theme of FASTUS". In *Proceedings of the 6th Message Understanding Conference* (MUC-6). Columbia, Md., 1995.

[5] Brill, E. "Some advances in rule-based part of speech tagging". In *Proceedings of the 12th National Conference on Artificial Intelligence* (AAAI-94). Seattle, 1994.

[6] Brill, E. *A corpus-based approach to language learning.* Doctoral Dissertation, Univ. Pennsylvania. 1993.

[7] Brill, E. "A simple rule-based part of speech tagger". In *Proceedings of the 3rd Conference on Applied Natural Language Processing* (Applied ACL 92). Trento, 1992.

[8] Hobbs, J. "Ontological promiscuity". In *Proceedings of the 23rd Meeting of the Association for Computational Linguistics* (ACL-85). Chicago, Ill., 1985.

[9] Landman, F. "Groups". *Linguistics and Philosophy,* 12(3), 359-605 and 12(4), 723-744.

[10] Lehnert, W., McCarthy, J., Soderland, S., Riloff, E., Cardie, C., Peterson, J., Feng, F., Dolan, C., & Goldman, S. "University of Massachusetts/Hughes: Description of the CIRCUS system as used for MUC-5". In *Proceedings of the 5th Message Understanding Conference* (MUC-5). Baltimore, Md., 1993.

[11] Vilain, M. Semantic inference in natural language: validating a tractable approach. In *Proceedings of the 14th Intl. Joint Conference on Artificial Intelligence* (IJCAI95). Montreal, 1995. 1346-1351.