SRI INTERNATIONAL: DESCRIPTION OF THE TACITUS SYSTEM AS USED FOR MUC-3

Jerry R. Hobbs
SRI International
Menlo Park, California 94025
hobbs@ai.sri.com
(415) 859-2229

BACKGROUND

TACITUS is a system for interpreting natural language texts that has been under development since 1985. It has a preprocessor and postprocessor currently tailored to the MUC-3 application. It performs a syntactic analysis of the sentences in the text, using a fairly complete grammar of English, producing a logical form in first-order predicate calculus. Pragmatics problems are solved by abductive inference in a pragmatics, or interpretation, component.

The original purpose of TACITUS was to aid us in investigating the problems of inferencing in natural language. For that reason, the system employed a straight-line modularization, with syntactic analysis being done by the already-developed DIALOGIC parser and grammar; only the correct parse was chosen and passed on the the inferencing component.

With the discovery of the abduction framework in 1987 [1], we realized that the proper way to deal with syntax-pragmatics interactions was in a unified abductive framework. However, the overhead in implementing such an approach at the level of coverage that the DIALOGIC system already provided would have been enormous, so that effort was not pursued, and we continued to focus on pragmatics problems.

When we began to participate in the MUC-2 and MUC-3 evaluations, we could no longer chose manually which syntactic analysis to process, so we began to invest more effort in the implementation of heuristics for choosing the right parse. We do not view this as the ideal way of handling syntax-pragmatics interactions, but, on the other hand, it has forced us into the development of these heuristics to a point of remarkable success, as an analysis of our results in the latest evaluation demonstrate.

We developed a preprocessor for MUC-2 and modified it for MUC-3. Our relevance filter was developed for MUC-3, as was our current template-generation component.

Those involved in the MUC-3 effort were Douglas Appelt, John Bear, Jerry Hobbs, David Magerman, Ann Podlozny, Mark Stickel, and Mabry Tyson. Others who have been involved the development of TACITUS over the years include Bonnie Lynn Boyd, William Croft, Todd Davies, Douglas Edwards, Kenneth Laws, Paul Martin, and Barney Pell.

THE MODULES OF THE SYSTEM

The system has six modules. As we describe them, their performance on Message 99 of TST1 will be described in detail, especially their performance on the first two sentences. Then their performance on the first 20 messages of TST2 will be summarized.

Preprocessor

This component regularizes the expression of certain phenomena, such as dates, times, and punctuation. In addition, it decides what to do with unknown words. There are three choices, and these are applied sequentially.

1. Spelling Correction. This is applied only to words longer than five letters.

2. Hispanic Name Recognition. A statistical trigram model for distinguishing between Hispanic surnames and English words was developed and is used to assign the category `Last-Name` to some of the words that are not spell-corrected.

3. Morphological Category Assignment. Words that are not spell-corrected or classified as last names, are assigned a category on the basis of morphology. Words ending in "-ing" or "-ed" are classified as verbs. Words ending in "-ly" are classified as adverbs. All other unknown words are taken to be nouns. This misses adjectives entirely, but this is generally harmless, because the adjectives incorrectly classified as nouns will still parse as prenominal nouns in compound nominals. The grammar will recognize an unknown noun as a name in the proper environment.

There were no unknown words in Message 99, since all the words used in the TST1 set had been entered into the lexicon.

In the first 20 messages of TST2, there were 92 unknown words. Each of the heuristics either did or did not apply to the word. If it did, the results could have been correct, harmless, or wrong. An example of a harmless spelling correction is "twin-engined" to the adjective "twin-engine". A wrong spelling correction is the verb "nears" to the preposition "near". An example of a harmless assignment of Hispanic surname to a word is the Japanese name "Akihito". A wrong assignment is the word "panorama". A harmless morphological assignment of a category to a word is the assignment of `Verb` to "undispute" and "originat". A wrong assignment is the assignment of `Noun` to "upriver".

The results were as follows:

	Unknown	Applied	Correct	Harmless	Wrong
Spelling Correction	92	25	8	12	5
Hispanic Surname	67	20	8	10	2
Morphological Assignment	47	47	29	11	7

If we look only at the Correct column, only the morphological assignment heuristic is at all effective, giving us 62%, as opposed to 32% for spelling correction and 40% for Hispanic surname assignment. However, Harmless assignments are often much better than merely harmless; they often allow a sentence to parse that otherwise would not. If we count both the Correct and Harmless columns, then spelling correction is effective 80% of the time, Hispanic surname assignment 90% of the time, and morphological assignment 86%.

Using the three heuristics in tandem meant that 85% of the unknown words were handled either correctly or harmlessly.

Relevance Filter

This component works on a sentence-by-sentence basis and decides whether the sentence should be submitted to further processing. It consists of two subcomponents.

1. Statistical Relevance Filter. We went through the 1300-text development set and identified the relevant sentences. We then developed a unigram, bigram, and trigram statistical model for relevance on the basis of this data. We chose our cutoffs so that we would identify 85% of the relevant sentences and overgenerate by no more than 300%. The component is now apparently much better than this.

2. Keyword Antifilter. In an effort to capture those sentences that slip through the statistical relevance filter, we developed an antifilter based on certain keywords. If a sentence in the text proves to contain relevant information, the next few sentences will be declared relevant as well if they contain certain keywords.

In Message 99, the statistical filter determined 9 sentences to be relevant. All of them were relevant except for one, Sentence 13. No relevant sentences were missed. The keyword antifilter decided incorrectly that two other sentences were relevant, Sentences 8 and 9. This behavior is typical.

In the first 20 messages of the TST2 set, the results were as follows: There were 370 sentences. The statistical relevance filter produced the following results:

	Actually Relevant	Actually Irrelevant
Judged Relevant	42	33
Judged Irrelevant	9	286

Thus, recall was 82%. Precision was 56%. These results are excellent. They mean that, using this filter alone, we would have processed only 20% of the sentences in the corpus, processing less than twice as many as were actually relevant, and only missing 18% of the relevant sentences.

The results of the keyword antifilter were as follows:

	Actually Relevant	Actually Irrelevant
Judged Relevant	5	57
Judged Irrelevant	4	227

Clearly, the results here are not nearly as good. Recall was 55% and precision was 8%. This means that to capture half the remaining relevant sentences, we had to nearly triple the number of irrelevant sentences we processed. Using the filter and antifilter in tandem, we had to process 37% of the sentences. The conclusion is that if the keyword antifilter is to be retained, it must be refined considerably.

Incidentally, of the four relevant sentences that escaped both the filter and the antifilter, two contained only redundant information that could have been picked up elsewhere in the text. The other two contained information essential to 10 slots in templates.

Syntactic Analysis

The sentences that are declared relevant are parsed and translated into logical form. This is done using the DIALOGIC system, developed in 1980-1 essentially by constructing the union of the Linguistic String Project Grammar and the DIAGRAM grammar which grew out of SRI's Speech Understanding System research in the 1970s. Since that time it has been considerably enhanced. It consists of about 160 phrase structure rules. Associated with each rule is a "constructor" expressing the constraints on the applicability of that rule, and a "translator" for producing the logical form.

The parser used by the system is a recently developed agenda-based scheduling chart-parser. As nodes and edges are built, they are rated and only a certain number of them are retained for further parsing. This number is a parameter the user can set. The nodes and edges are rated on the basis of their scores from the preference heuristics. Prior to November 1990, we used a simple, exhaustive, bottom-up parser, with the result that sentences of more 15 or 20 words could not be parsed. The use of the scheduling parser has made it feasible to parse sentences of up to 60 words.

For sentences of longer than 60 words and for faster, though less accurate, parsing of shorter sentences, we developed a technique we are calling "terminal substring parsing". The sentence is segmented into substrings, by breaking it at commas, conjunctions, relative pronouns, and certain instances of the word "that". The substrings are then parsed, starting with the last one and working back. For each substring, we try either to parse the substring itself as one of several categories or to parse the entire set of substrings parsed so far as one of those categories. The best such structure is selected, and for subsequent processing, that is the only analysis of that portion of the sentence allowed. The categories that we look for include main, subordinate, and relative clauses, infinitives, verb phrases, prepositional phrases, and noun phrases. The effect of this technique is to give only short "sentences" to the parser, without losing the possibility of getting a single parse for the entire long sentence. Suppose a sixty-word sentence is broken into six ten-word substrings. Then the parsing, instead of taking on the order of 60^3 in time, will only take on the order of $6 * 15^3$. (When parsing the initial 10-word substring, we are in effect parsing at most a 15-"word" string covering the entire sentence, consisting of the 10 words plus the nonterminal symbols covering the best analyses of the other five substrings.)

When sentences do not parse, we attempt to span it with the longest, best sequence of interpretable fragments. The fragments we look for are main clauses, verb phrases, adverbial phrases, and noun phrases. They are chosen on the basis of length and their preference scores. We do not attempt to find fragments for strings of less than five words. The effect of this heuristic is that even for sentences that do not parse, we are able to extract 88% of the propositional content.

The parse tree is translated into a logical form that regularizes to some extent the role assignments in the predicate-argument structure. For example, for a word like "break", if the usage contains only a subject, it is taken to be the Patient, while if it contains a subject and object, they are taken to be the Agent and Patient respectively. Arguments inherited from control verbs are handled here as well.

Our lexicon includes about 12,000 entries, including about 2000 personal names and about 2000 location, organization, or other names. This does not include morphological variants, which are dealt with in a separate morphological analyzer.

In Message 99, of the 11 sentences determined to be relevant, only Sentence 14 did not parse. This was due to a mistake in the sentence itself, the use of "least" instead of "at least". Hence, the best fragment sequence was sought. This consisted of the two fragments "The attacks today come after Shining Path attacks" and "10 buses were burned throughout Lima on 24 Oct." The parses for both these fragments were completely correct. Thus, the only information lost was from the three words "during which least".

Of the 10 sentences that parsed, 5 were completely correct, including the longest, Sentence 7 (27 words in 77 seconds). There were three mistakes (Sentences 3, 4, and 9) in which the preferred multiword senses of the phrases "in front of" and "Shining Path" lost out to their decompositions. There were two attachment mistakes. In Sentence 3 the relative clause was incorrectly attached to "front" instead of "embassy", and in Sentence 8, "in Peru" was attached to "attacked" instead of "interests". All of these errors were harmless. In addition, in Sentence 5, "and destroyed the two vehicles" was grouped with "Police said ..." instead of "the bomb broke windows"; this error is not harmless. In every case the grammar prefers the correct reading. We believe the mistakes were due to a problem in the scheduling parser that we discovered the week of the evaluation but felt was too deep and far-reaching to attempt to fix at that point.

In the first 20 messages of TST2, 131 sentences were given to the normal parser (as opposed to the terminal substring parser). A parse was produced for 81 of the 131, or 62%. Of these, 43 (or 33%) were completely correct. 30 more had three or fewer errors. Thus, 56% of the sentences were parsed correctly or nearly correctly.

These results naturally vary depending on the length of the sentences. There were 64 sentences of under 30 morphemes. Of these, 37 (58%) had completely correct parses and 48 (75%) had three or fewer errors. The normal parser attempted only 8 sentences of more than 50 morphemes, and only two of these parsed, neither of them even nearly correctly.

Of the 44 sentences that would not parse, 9 were due to problems in lexical entries. 18 were due to shortcomings in the grammar. 6 were due to garbled text. The causes of 11 failures to parse have not been determined. These errors are spread out evenly across sentence lengths. In addition, 7 sentences of over 30 morphemes hit the time limit we had set, and terminal substring parsing was invoked.

The shortcomings in the grammar were the following constructions, which are not currently covered:

which Adverbial VP
Subordinate-Conjunction Adverbial S
as VP
the next few days
more Noun to X than to Y
NP and, Adverb, NP (this is handled without the commas)
of how S
Adverb or Adverb
(NP, NP)
Verb – Adverbial – NP
Infinitive and Infinitive
S (containing the word "following") : NPConjunction
PP is NP
be as S/NP

no longer
cut short NP

Our results in syntactic analysis are quite encouraging since they show that a high proportion of a corpus of long and very complex sentences can be parsed nearly correctly. However, the situation is even better when one considers the results for the best-fragment-sequence heuristic and for terminal substring parsing. A best sequence of fragments was sought for the 44 sentences that did not parse for reasons other than timing. A sequence was found for 41 of these. The average number of fragments in a sequence was two. This means that an average of only one structural relationship was lost. Moreover, the fragments covered 88% of the morphemes. That is, even in the case of failed parses, 88% of the propositional content of the sentences was made available to pragmatics.

For 37% of these sentences, correct syntactic analyses of the fragments were produced. For 74%, the analyses contained three or fewer errors. Correctness did not correlate with length of sentence.

Terminal substring parsing was applied to 14 sentences, ranging from 34 to 81 morphemes in length. Only one of these parsed, and that parse was not good. This is not surprising, given that this technique is called only when all else has already failed. Sequences of fragments were found for all the other 13 sentences. The average number of fragments was 2.6, and the sequences covered 80% of the morphemes. None of the fragment sequences was without errors. However, eight of the 13 had three or fewer mistakes.

We have found all of this extremely encouraging. Even more encouraging is the fact that a majority of the errors in parsing can be attributed to five or six causes. Two prominent ones are the tendency of the scheduling parser to lose favored close attachments of conjuncts and adjuncts near the end of sentences, and the tendency to misanalyze the string

$$[[\text{Noun Noun}]_{NP}\ \text{Verb}_{trans}\ \text{NP}]_{S}$$

as

$$[\text{Noun}]_{NP}\ [\text{Noun Verb}_{ditrans}\ ()\ \text{NP}]_{S/NP}$$

We believe that many such problems could be solved with a few days work.

Pragmatics, or Interpretation

The literals in the logical form are assigned assumability costs, based on their syntactic role, the predicates involved, and other factors. They are then passed to the abductive theorem-prover PTTP, which attempts to find a proof from a knowledge base of axioms about the terrorist domain. The fundamental idea behind this component is that the interpretation of a text is the best explanation for what would make it true. Generally, in this domain, the explanation is one that involves seeing the text as an instance of an "Interesting Act" schema, a schema which includes the principal roles in bombings, kidnappings, and so forth. The explanation of a sentence is identified with an abductive proof of its logical form. This proof may include assumptions of unprovable literals, and each assumption incurs a cost. Different proofs are compared according to the cost of the assumptions they introduce, and the lowest cost proof is taken to be the best explanation, *provided that all the assumptions are consistent.*

The agents and objects of "Interesting Acts" are required to be "bad guys" and "good guys" respectively. "Bad guys" are terrorists, guerrillas, and their organizations, and good guys are civilians, judges, government officials, etc. Members of the armed forces can be "bad guys" on certain occasions, but they are never "good guys."

The knowledge base includes a taxonomy of people and objects in the domain. The primary information that is derived from this taxonomy is information about the disjointness of classes of entities. For example, the classes of "good guys" and "bad guys" are disjoint, and any abductive proof that assumes "good guy" and "bad guy" of the same entity is inconsistent. To view an attack by guerrillas on regular army troops as an interesting act would require assuming the victims, i.e. the troops, were "good guys" and since the "good guys" are inconsistent with the military, no consistent explanation of the event in question in terms of "Interesting Act" is possible, and hence no template would be generated for such an incident.

The abductive reasoner attempts to minimize the extensions of most predicates by factoring goals with previous assumptions. That means that whenever it is consistent to assume that two individuals that share

a property represented by one of the predicates to be minimized are the same, it does so. This factoring mechanism is the primary mechanism by which anaphora is resolved. Two entities with similar properties are generally assumed to be identical. Pronominal anaphora works differently, in that the structure of the text is taken into account in creating an ordered list of possible antecedents. The abductive reasoner will resolve the pronoun with the first object on the antecedent list that leads to a consistent proof.

Using the factoring mechanism for anaphora resolution requires one to have a rich enough domain theory so that incorrect resolutions can be eliminated from consideration. Otherwise, the system is strongly biased toward collapsing everything into a single individual or event. On the other hand, consistency checking can be computationally hard, and whatever theory is adopted for consistency checking must be fast. Our experience has been that the taxonomic consistency check described above is mostly adequate for rejecting incorrect resolutions, but we have found it necessary to augment the taxonomic check with some other strategies for determining inconsistency. For example, we reject as inconsistent any proof that assumes that a single individual has two distinct proper surnames.[1] We also assume it is inconsistent to resolve an individual with a set, and to resolve two sets that are known to be of different cardinality.

The domain knowledge base is divided into a set of axioms, which are used for abductively proving the sentences from the text, and a class hierarchy, which is used for checking the consistency of the proofs. The axioms are divided into a core set of axioms describing the events in the domain that correspond to the incident types, and lexical axioms, which are meaning postulates that relate the predicate introduced by a lexical item to the core concepts of the domain.

The knowledge base includes approximately 550 axioms at the current stage of development. This breaks down into about 60 axioms expressing the core facts about the schemas of interest, 430 axioms relating lexical entries to these core schemas, and approximately 60 axioms for resolving compound nominals, of-relations, and possessives. The knowledge base also includes approximately 1100 locations, for which relevant axioms are introduced automatically at run-time.

Template Generation

The task of the template generation component is to take the results of the abductive proofs in pragmatics, and put them into a template form according to the specifications of the task. This generates one template for every interesting act that is assumed by pragmatics, with several exceptions. An interesting act can be both an ATTACK and a MURDER, and only the MURDER template would be produced. An interesting act of type MURDER might be divided into two templates, if it was found that some of the victims survived the attack. For example "Terrorists shot John and Mary. John was wounded and Mary was found dead at the scene," would generate one MURDER template and one ATTEMPTED MURDER template.

For each interesting act, a cluster of contemporaneous and causally related events from the text is formulated. Any temporal or locative information that is associated with any of these events, or the agents and objects participating in the events, is used to fill the DATE and LOCATION slots of the respective templates. Each slot is then filled by looking at the arguments of the relevant predicates, and if any of these arguments represent sets, the sets are expanded into their constituents for the slot fills.

For string fills, proper names are preferred, if any are known, and if not, the longest description from all the coreferential variables denoting that entity is used, excluding certain uninformative descriptors like "casualties."

In a final pass, analysis eliminates from consideration templates that do not pass certain coherence or relevance filters. For example, any template that has a "bad guy" as the object of an attack is rejected, since this is probably a result of an error in solving some pragmatics problem. Templates for events that take place in the distant past are rejected, as well as events that take place repeatedly or over vague time spans (e.g. "in the last three weeks"). Finally, templates for events that take place in irrelevant countries are eliminated. This final filter, unfortunately, can eliminate entirely otherwise correct templates for which the location of the incident is incorrectly identified. This was responsible for several costly mistakes in the evaluation.

[1] This, of course, does not account for the situation in which a criminal has an alias, but in practice this occurs seldom enough, and the effect of this mistake on the ability to produce correct template fills seems small enough that it is clearly a benefit to do so.

CAUSES OF FAILURES

It is difficult to evaluate the interpretation and template generation components individually. However, we have examined the first twenty messages of TST2 in detail and attempted to pinpoint the reason for each missing or incorrect entry in a template.

There were 269 such mistakes, due to problems in 41 sentences. We have classified them into a number of categories, and the results for the principal causes are as follows:

Reason	Mistakes	Sentences
Simple Axiom Missing	49	9
Unknown Words	38	3
Combinatorics	28	3
Parsing Problems	26	5
Unconstrained Factoring	25	3
Lexicon Error	24	2
Syntax-Pragmatics Mismatch in Logical Form	22	5
Complex Axioms or Theory Missing	14	5
Relevance Filter Missed Sentence	11	2
Underconstrained Axiom	8	3

An example of a missing simple axiom is that "bishop" is a profession. An example of a missing complex axiom or theory is whatever it is that one must know to infer the perpetrator from the fact that a flag of a terrorist organization was left at the site of a bombing. An underconstrained axiom is one that allows, for example, "damage to the economy" to be taken as a terrorist incident. Unconstrained factoring is described above. An example of a lexicon error would be a possibly intransitive verb that was not correctly specified as intransitive. The syntax-pragmatics mismatches in logical form were representation decisions (generally recent) that did not get reflected in either the syntax or pragmatics components. "Combinatorics" simply means that the theorem-prover timed out; that this number was so low was a pleasant surprise for us.

Note in these results that two incorrect lexical entries and problems in handling three unknown words were responsible for 23% of the mistakes. This illustrates the discontinuous nature of the mapping from processing to evaluation. A difference of δ in how a text is processed can result in a difference of considerably more than ϵ in score. The lesson is that the scores cannot be used by themselves to evaluate a system. One must analyze its performance at a deeper, more detailed level, as we have tried to do here.

ACKNOWLEDGEMENTS

This research has been funded by the Defense Advanced Research Projects Agency under Office of Naval Research contracts N00014-85-C-0013 and N00014-90-C-0220.

REFERENCES

[1] Hobbs, Jerry R., Stickel, Mark, Appelt, Douglas, and Martin, Paul, "Interpretation as Abduction", SRI International Artificial Intelligence Center Technical Note 499, December 1990.