METAL: The LRC Machine Translation System

Jonathan Slocum

with Winfield S. Bennett,
John Bear, Martha Morgan, and Rebecca Root

Linguistics Research Center
University of Texas

Working Paper LRC-84-2
April, 1984

This paper was presented at the ISSCO Tutorial on Machine Translation held in Lugano, Switzerland, from 2-6 April, 1984.

METAL: The LRC Machine Translation System

Jonathan Slocum

with Winfield S. Bennett,
John Bear, Martha Morgan, and Rebecca Root

Linguistics Research Center
The University of Texas

Introduction

The Linguistics Research Center of the University of Texas has been working on Machine Translation (MT) since its founding in 1961. Unlike some centers of MT research, its early focus was largely theoretical. Most U.S. groups which developed first- and second-generation systems for MT were eliminated through the impact of the National Academy of Sciences report, "Languages and Machines," released in 1966. Limited funding continued, with some lapses, at the LRC. In 1978, Rome Air Development Center provided the means to apply the Center's theoretical findings; this was augmented in 1979 by support from Siemens AG, Munich. In 1980, Siemens became the sole project sponsor.

We will describe here the prototype MT system developed by the LRC. With the exception of the text-processing programs, which are written in SNOBOL, the LRC MT system is written in LISP and runs on a Symbolics Lisp Machine. The system includes a translation program, METAL, as part of a suite of programs designed to automate the complete process of translating technical texts. Sections of this paper will deal with our domain of application, a general description of the MT system, an overview of some of our key linguistic and computational techniques, a more detailed description of METAL, notes on the METAL parser, the results of some extensive experiments in translating German telecommunication and data processing texts into English, and conclusions regarding our contribution to the science of Computational Linguistics.

The Domain of Application

Natural language texts range in "complexity" from edited abstracts through technical documentation and scientific reports to newspaper articles and literary materials. As far as MT is concerned, the former portion of this spectrum is less complex than the latter because it is characterized by relatively less syntactic and semantic variety. Paradoxically, the order of complexity for human translators is essentially reversed due to a dramatic increase in the size of the vocabulary: no qualified human translator has much difficulty with straightforward syntax or normal idiomatic usage, but the prevalence and volatility of technical terms and jargon poses a considerable problem. Time pressures impose another set of problems, for which there appear to be no solutions in the realm of unaided human translation, and few if any good ones in the realm of machine-aided human translation as commonly interpreted (e.g., a text editor coupled with automated dictionary look-up).

In terms of the demand for MT, then, the market is in technical translation. There is no significant demand for machine translation of folklore, literary materials, and the like, but there is a substantial and growing demand for machine translation of technical texts.

Advantages of Translating Technical Texts

There are several advantages in translating technical texts as opposed to more general texts. One of these concerns vocabulary: technical texts tend to concentrate on one subject area at a time, wherein the terminology (lexical semantics) is relatively consistent, and where the vocabulary is relatively unambiguous, even though it may be quite large. (This is not to say that lexical problems disappear!) Another advantage is that there is typically little problematic anaphora, and little or no "discourse structure" as usually defined. Third, in accordance with current practice for high-quality human translation, revision is to be expected. That is, there is no a priori reason why machine translations must be "perfect" when human translations are not expected to be so: it is sufficient that they be acceptable to the humans who revise them, and that they prove cost-effective overall (including revision).

Problems in Translating Technical Texts

Notwithstanding the advantages of translating technical texts, there are definite problems to be confronted. First of all, the volume of such material is staggering: potentially tens or hundreds of millions of pages per year. Even ignoring all cost-effectiveness considerations, the existence of this much candidate material demands a serious concern for efficiency in the implementation. Second, the emphasis in Machine Translation is changing from information acquisition to information dissemination. The demand is not so much for loosely approximate translations from which someone knowledgeable about the subject can infer the import of the text (perhaps with a view toward determining whether a human translation is desired); rather, the real demand is for high-quality translations of, e.g., operating and/or maintenance manuals -- for instructing someone not necessarily knowledgeable about the vendor's equipment in precisely what must (and must not) be done, in any given situation. Fidelity, therefore, is essential.

In addition to the problems of size and fidelity, there are problems regarding the text itself: the format and writing style. For example, it is not unusual to be confronted with a text which has been "typeset" by a computer, but for which the typesetting commands are no longer available. This can be true even when the text was originally produced, or later transcribed, in machine-readable form. The format may include charts, diagrams, multi-column tables, section headings, paragraphs, etc. Misspellings, typographical errors, and grammatical errors can and do appear. Technical texts are notable for their frequency of "unusual" syntax such as phrase- and sentence-fragments, a high incidence of acronyms and formulas, plus a plethora of parenthetical expressions. The "discourse structure," if it can be argued to exist, may be decidedly unusual -- as exemplified by a flowchart. Unknown words will appear in the text. Sentences can be long and complicated, notwithstanding the earlier statement about reduced complexity. Technically-oriented individuals are renowned for abusing natural language. The sucessful MT system will address these problems as well as those more commonly anticipated.

General System Description

In this section we discuss the facilities of the LRC MT system which substantially automate the overall translation process, including the production and maintenance of the lexical databases along with several text-processing programs and METAL's place among them.

Lexical Database Management

In any large software system, the problem of producing and maintaining the data sets on which the programs operate becomes important, if not critical. First of all, when there is a large volume of such material the data entry process itself can consume a significant amount of time; second, the task of insuring data integrity becomes an even larger time sink. We will briefly expand on these two problems, and indicate how the LRC MT system provides software tools to help cope with them.

There are two problems associated with data entry: creating the data in the first place, and getting it entered in machine-readable form. In most applications of database management systems, the creation of data is relatively straightforward: such data items as personal name, identification number(s), age, job title, salary, etc., serve as examples to illustrate the point that the data items usually pre-exist, thus data entry becomes relatively more important. In an application such as MT represents, however, creating the original data is the major bottleneck: one must decide, for each of thousands of words, many details of behavior in a complicated linguistic environment. Certainly these details may be said to "pre-exist," but a real problem arises when humans attempt to identify them. In general, the more sophisticated the MT system, the more of these details there are. Data entry, relatively speaking, becomes a small or insignificant issue -- although it remains a significant issue in absolute terms.

As part of the LRC MT system, we have developed a sophisticated "lexical default" program that accepts minimal information (the root form of the word, and its category) and automatically encodes almost all of the features and values that, for METAL, specify the details of linguistic behavior. This is accomplished by a combination of morphological analysis of the root form of the input word, and search of the existing lexical database for "similar" entries. Defaulted lexical entries are created in machine-readable form to begin with, and are available for human review/revision using standard on-line editing facilities. This greatly reduces both coding time and coding errors.

A potentially harder problem, however, is the maintenance of data integrity. Humans will make mental errors in creating lexical entries, and will aggravate these by making typographical errors during data entry. Even assuming a lexical default program (which, of course, does not make such mistakes), the process of human revision of the defaulted entries may introduce errors. Therefore, the LRC MT system includes a validation program that, working from a formal specification of what is legal in lexical entries, identifies any errors of format and/or syntax within each submitted entry. The formal specification is organized by language, by lexical category within language, and by feature within category. The incidence of semantic errors -- which our validation program could not detect -- has not been found to be significantly high. As a result, the use of this program has virtually eliminated incorrect coding of monolingual lexical entries, which used to be a major source of error in METAL translations. (Related programs employ similar techniques to identify errors in the phrase-structure grammar rules and transformations that constitute the remainder of our linguistic rule base.)

There is also the problem of maintaining an existing lexical database. In any MT system, there will be a need for changing existing entries in the light of experience; in a system like ours, which serves as a vehicle for research in MT, this problem is magnified by the occcasional need for large-scale changes in lexical entries to accommodate new system features, or even theories of translation. As part of the LRC MT system, then, we have incorporated a general relational Data Base Management System (DBMS) along with a group of interface routines that transform, upon entry, both monolingual and transfer lexical entries from a format optimized for human use into a format more suitable for storage by the DBMS, and which reverse the transformation when retrieving the entries. Not only do the interface routines facilitate the entry, retrieval, and revision of lexical entries, but they also are integrated with the validation program so that a lexicographer may not, while using the DBMS, introduce errors in the format and/or syntax of a lexical entry. This same on-line DBMS is accessed directly by METAL during translation; therefore, changes made in the database are instantly reflected in translations, resulting in rapid turnaround that both encourages and enhances research and development.

Finally, there is the problem of tying all these modules together with a powerful, high-level user interface which optimizes the task of entry acquisition and maintenance. The LRC has developed an "Interactive Coder" that uses a menu-selection scheme as the interface mechanism for acquiring and maintaining lexical entries. Starting with words typed in, or drawn from a concordance of unknown words resulting from the dictionary analysis of a new text, or appearing in an abbreviated transfer entry input file created by a text editor or perhaps a formatted dump from an existing on-line database, the Interactive Coder will use the database interface, the default program, the formal specifications governing lexical entries, and the validator, to facilitate the process of lexical entry acquisition and maintenance. The METAL system runs on a Lisp Machine with a high-resolution bit-mapped display and a "mouse" pointing device, and the Interactive Coder takes great advantage of this sophisticated, powerful menu interface medium to optimize the overall dictionary maintenance task. Figure 1 outlines the process of semi-automatic terminology acquisition and subsequent "linguistic coding" supported by the lexical database management system.

Text-processing Programs

As a glance at any technical manual will show, it is not always the case that all material in a document must or can be translated. Large portions of a text (up to 50% of the characters, in our experience) may not be translatable material; the bulk of this may fall outside sentence boundaries, but some will fall within them. Thus it is necessary for a text to be marked, or annotated, to distinguish that which is to be translated (e.g., tables of contents, instructions, prose paragraphs) from that which is not (e.g., text formatting information, flowchart box boundaries, acronyms, and various attention-focusing devices). In the LRC MT system, a program determines which members of the machine's character set are unused in the text at hand, and a few of these are used to mark the text.

```
                         |
                         |
  +----------------------------------------------+
  | Acquire terminology in machine-readable form |
  +----------------------------------------------+
                         |
+----------------------->|
|                        |
|            +-----------------------+
|            | Code transfer entries |
|            +-----------------------+
|                        |
|                        |
|          +---------------------------+
|          | Validate transfer entries |
|          +---------------------------+
| (error)              | | |
+----------<-----------+ | +----------->-----------+
                         |                         |
      +--------------------------------------+     |
      | Generate default monolingual entries |     |
      +--------------------------------------+     |
                         |                         |
+----------------------->|                         |
|                        |                         |
|      +-----------------------------------+       |
|      | Review/revise monolingual entries |       |
|      +-----------------------------------+       |
|                        |                         |
|                        |                         |
|         +------------------------------+         |
|         | Validate monolingual entries |         |
|         +------------------------------+         |
| (error)              | |                         |
+---------<------------+ |<------------------------+
                         |
            +------------------------+
            | Add entries to database |
            +------------------------+
                         |
                         |
            +--------------------------+
            | Release database for use |
            +--------------------------+
                         |
                         |
```

Figure 1

Coding Lexical Entries
using the Interactive Coder

Within multi-column tables, different sentences would intermingle if the MT system were to read the text in the usual computer fashion (i.e., horizontally across the line). Our text-processing component currently detects most such situations, and makes annotations that, after further automatic processing, will allow the translation component to "see" the sentences as humans would: reading vertically down the page. Another annotation program attempts to identify translatable units (isolated words, phrases, and sentences) and brackets them. Some untranslatable material (e.g., flowchart box labels) may appear to fall within sentence boundaries, but plays no grammatical role in a sentence: therefore we introduced a "toggle" convention for excluding such material so that it is invisible to the analyzer. On the other hand, some untranslatable material within a sentence (e.g., equations and formulas) may indeed play a grammatical role: the annotation program marks such strings to be analyzed, but carried through to the target language without translation. Human verification and emendation of the annotation component's output is necessary, as might be expected, but this task does not require significant training, and in particular it requires no knowledge of the Target Language(s) or the MT system; it does require knowledge of what is to be translated.

Once the text has been annotated, another program extracts the sentences to be translated and prepares them for input to the MT system. Essentially, this entails copying any bracketed units into a new file, excluding toggled-out material. Anything outside of brackets is ignored completely. Complicating this process is the fact that, in order to minimize human intervention in reformatting the text after translation, the extraction program must record the location of the original sentences copied into the MT input file, so that the results of translation may be formatted just like the original document.

The next (optional) step is dictionary pre-analysis of the text. The MT input file is scanned, and the unique word occurrences are identified; each is then submitted to lexical analysis. This step offers the potential advantage of detecting dictionary shortages (i.e., unknown words) before they have a chance to affect the translation process; also, METAL's proposed spelling corrections can be noted for human review, and suspected acronyms can be verified. A concordance of the unknown and misspelled words is constructed; this may be accessed, for example, by the menu-driven Interactive Coder, if the user decides to create dictionary entries for unknown words before proceeding with the translation. After the user has checked (and, if desired, corrected) any textual errors noted during dictionary pre-analysis, translation may proceed.

Translation, briefly stated, involves reading in the Source Language (SL) file of sentences and printing out the corresponding Target Language (TL) file. This step -- performed by METAL itself -- is detailed below. The process of text reformatting is an important component of the overall process of translation. If an MT system provides only a sequence of translated sentences, then a human revisor must manually reformat the entire document, producing charts, figures, etc. This can be a source of substantial and unnecessary frustration, and can constitute a significant cost factor in translation. The alternative of providing unformatted translations to the end-user is distinctly unattractive. In the LRC MT system, a special reconstitution program automatically formats translated text as much like the original text as is possible: margins, indentation, and blank lines are all observed, and special forms (e.g., multi-column tables) are all reconstructed.

The last step involves human revision of the machine´s translations. (This step may actually take place before reconstitution, using the MT input and MT output files separately, or an interlinear version of these two that is also available; but some small text format problems may remain that require human attention after reconstitution.) The linguistic and computational research behind METAL has always had as its goal a comprehensive analysis and translation of whole sentences in their context; the work of the revisor, then, should be mostly a matter of review. Where METAL fails to achieve a unified analysis of a sentence, a translation by phrases is provided which the revisor must render into a well-formed translation. Figure 2 depicts the steps in "production" machine translation using the LRC MT system.

```
            +----------------------------------------+
            | Acquire text in machine-readable form |
            +----------------------------------------+
                              |
                              |
            +------------------------------------------+
            |   Identify characters used in text;     |
            | choose other characters for annotation |
            +------------------------------------------+
                              |
                              |
                   +---------------------------+
                   | Mark multi-column tables |
                   |=      Human review      =|
                   +---------------------------+
                              |
                              |
                  +-----------------------------+
                  | Mark translatable material |
                  |=       Human review       =|
                  +-----------------------------+
                              |
                              |
                +---------------------------------+
                | Extract translatable material; |
                |  produce format template file  |
                +---------------------------------+
                              |
                              +------------>---------------+
                              |                 optionally |
                    +--------------------------+           |
                    | Identify unknown words  |           |
                    |=  Update lexicon(s)    =|           |
                    +--------------------------+           |
                              |                            |
                              |<---------------------------+
                              |
                   +--------------------------+
                   | Translate text with METAL |
                   +--------------------------+
                              |
                    +-----<-----+----->-----+
                    |        (choice)        |
            +----------------+         +------------------+
            | Reconstitution |         |= Human revision =|
            +----------------+         +------------------+
                    |                          |
                    |                          |
          +--------------------+       +--------------------+
          |= Human revision  =|       |  Reconstitution   |
          |= Touch-up format =|       |= Touch-up format =|
          +--------------------+       +--------------------+
```

Figure 2
Steps in Translation

Linguistic Techniques Employed

Our distinction between "linguistic techniques" and "computational techniques" (discussed in the next major section) is somewhat artificial, but it has some validity in a broad sense, as should become clear from an overview of the points considered. In this section we present the reasons for our use of the following linguistic techniques: (a) a phrase-structure grammar; (b) syntactic features; (c) semantic features; (d) scored interpretations; (e) transformations indexed to specific rules; (f) a transfer component; and (g) attached procedures to effect translation.

Phrase-Structure Grammar

In the LRC MT system we employ a phrase-structure grammar, augmented by sufficient lexical controls to make it resemble lexical-functional grammar [Bresnan, 1977]. Of all our linguistic decisions, this is surely the most controversial, and consequently will receive the most attention. Generally speaking, there are two competing claims: first, that syntax rules per se are inadequate and wasteful (e.g., [Cullingford, 1978]); and second, that other forms of grammar (ATNs [Woods, 1970], transformational [Petrick, 1973], procedural [Winograd, 1972], word-experts [Small, 1980], etc.) are superior. We will deal with these in turn.

There are schools of thought that claim that syntax rules per se are inappropriate models of language. Language should, according to this notion, be treated [almost] entirely on the basis of semantics, guided by a strong underlying model of the current situational context, and the expectations that may be derived therefrom. We cannot argue against the claim that semantics is of critical concern in Natural Language Processing. However, as yet no strong case has been advanced for the abandonment of syntax. Moreover, no system has been deleloped by any of the adherents of the "semantics only" school of thought that has more-or-less successfully dealt with ALL of a wide range -- or at least large volume -- of material. A more damaging argument against this school is that every NLP system to date that HAS been applied to large volumes of text (in the attempt to process ALL of it some significant sense) has been based on a strong syntactic model of language (see, e.g., [Boitet et al., 1980b], [Damerau, 1981], [Hendrix et al., 1978], [Lehmann et al., 1981], [Martin et al., 1981], [Robinson, 1982], and [Sager, 1981]).

There are other schools of thought that hold context-free phrase-structure (PS) rules in disrespect, while admitting the utility (necessity) of syntax. It is claimed that the phrase-structure formalism is inadequate, and that other forms of grammar are necessary. (This has been a long-standing position in the linguistics community; but this view is now being challenged by some, who are once again supporting PS rules as a model of natural language use [Gazdar, 1981, 1983].) The anti-PS positions in the Natural Language Processing community are all, of necessity, based on practical considerations, since the models advanced to replace PS rules are formally equivalent in generative power (assuming the PS rules to be augmented, which is always the case in modern NLP systems employing them). But cascaded ATNs [Woods, 1980], for example, are only marginally different from PS rule systems. It is curious to note that only one of the remaining contenders (a transformational grammar [Damerau, 1981]) has been demonstrated in large-scale application -- and even this system employs PS rules in the initial stages of parsing. Other formal systems (e.g., procedural grammars [Winograd, 1972]) have been applied to semantically deep (but linguistically impoverished) domains -- or to excessively limited domains (e.g., Small's [1980] "word expert" parser seems to have encompassed a vocabulary of well under 20 items).

For practical application, it is necessary that a system be able to accumulate grammar rules, and especially lexical items, at a prodigious rate by current NLP standards. The formalisms competing with PS rules and dictionary entries of modest size seem to be universally characterizable as requiring enormous human resources for their implementation in even a moderately large environment. This should not be surprising: it is precisely the claim of these competing methodologies (those that are other than slight variations on PS rules) that language is an exceedingly complex phenomenon, requiring correspondingly complex techniques to model. For "deep understanding" applications, we do not contest this claim. But we do maintain that there are some applications that do not seem to require this level of effort for adequate results in a practical setting. Our particular application -- automated translation of technical texts -- seems to fall in this category, as does, e.g., the EPISTLE text-critiquing system [Heidorn et al., 1982].

The LRC MT system is currently equipped with approximately 550 PS rules describing the best-developed Source Language (German), and around 10,000 lexical entries in each of the two main languages (German, and the best-developed Target Language: English). The current state of our coverage of the SL is that the system is able to parse and acceptably translate the majority of sentences in previously-unseen texts, within the subject areas bounded by our dictionaries. We have recently begun the process of adding to the system an analysis grammar of the current TL (English), so that the direction of translation may be reversed; we anticipate bringing the English grammar up to the level of the German grammar in about two years' time. Our expectations for eventual coverage are that, for each SL, around 1,000 PS rules will be adequate to account for almost all sentence forms actually encountered in technical texts. We do not feel constrained to account for every possible sentence form in such texts -- nor for sentence forms not found in such texts (as in the case of poetry) -- since the required effort would not be cost-effective whether measured in financial or human terms, even if it were possible using current techniques (which we doubt).

Syntactic Features

Our use of syntactic features is relatively noncontroversial, given our choice of the PS rule formalism. We employ syntactic features for two purposes. One is the usual practice of using such features to restrict the application of PS rules (e.g., by enforcing subject-verb number agreement). The other use is perhaps peculiar to our type of application: once an analysis is achieved, certain syntactic features are employed to control the course (and outcome) of translation -- i.e., generation of the TL sentence. The "augmentations" to our PS rules include procedures written in a formal language (so that our linguists do not have to learn LISP) that manipulate features by restricting their presence, their values if present, etc., and by moving them from node to node in the "parse tree" during the course of the analysis. As is the case with other researchers employing such techniques, we have found this to be an extremely powerful (and of course necessary) means of restricting the activities of the parser.

Semantic Features

We employ simple semantic features, as opposed to complex models of the domain. Our reasons are primarily practical. First, features seem sufficient for at least the initial stage of our application. Second, the thought of writing complex models of even one complete technical domain is staggering: the operation and maintenance manuals we have worked with (describing a digital telephone switching system) are part of a document collection that is expected to comprise some 100,000 pages of text when complete. A research group the size of ours would not even be able to read that volume of material, much less write the "necessary" semantic models subsumed by it, in any reasonable amount of time. (The group members would also have to become electronics engineers, in all likelihood.) If such models are indeed required for our application, we will never succeed.

As it turns out, we are doing surprisingly well without such models. In fact, our semantic feature system is not yet being employed to restrict the analysis effort at all; instead, it is used at "transfer time" (described later) to improve the quality of the translations, primarily of prepositions. We look forward to extending the use of semantic features to other parts of speech, and to substantive utilization during analysis; but even we were pleased at the results we achieved using only syntactic features.

Scored Interpretations

It is a well-known fact that NLP systems tend to produce many readings of their input sentences (unless, of course, constrained to produce the first reading only -- which can result in the "right" interpretation being overlooked). The LRC MT system may produce multiple interpretations of the input "sentence," assigning each of them a score, or plausibility factor [Robinson, 1982]. This technique can be used, in theory, to select a "best" interpretation from the available readings of an ambiguous sentence. We base our scores on both lexical and grammatical phenomena -- plus the types of any spelling/typographical errors, which can sometimes be "corrected" in more than one way.

Our experiences relating to the reliability and stability of heuristics based on this technique are decidedly positive: we employ only the (or a) highest-scoring reading for translation (the others being discarded), and our informal experiments indicate that it is rarely true that a better translation results from a lower-scoring analysis. (Surprisingly often, a number of the higher-scoring interpretations will be translated identically. But poorer translations are frequently seen from the lower-scoring interpretations, demonstrating that the technique is indeed effective.)

Indexed Transformations

We employ a transformational component, during both the analysis phase and the translation phase. The transformations, however, are indexed to specific syntax rules rather than loosely keyed to syntactic constructs. (Actually, both styles are available, but our linguists have never seen the need or practicality of employing the open-ended variety). It is clearly more efficient to index transformations to specific rules when possible; the import of our findings is that it seems to be unnecessary to have open-ended transformations -- even during analysis, when one might intuitively expect them to be useful.

Transfer Component

It is frequently argued that translation should be a process of analyzing the Source Language (SL) into a "deep representation" of some sort, then directly synthesizing the Target Language (TL) (e.g., [Carbonnel, 1978]). We and others [King, 1981] contest this claim -- especially with regard to "similar languages" (e.g., those in the Indo-European family). One objection is based on large-scale, long-term trials of the "deep representation" (in MT, called the "pivot language") technique by the MT group at Grenoble [Boitet, 1980a]. After an enormous investment in time and energy, including experiments with massive amounts of text, it was decided that the development of a suitable pivot language (for use in Russian-French translation) was probably impossible. Another objection is based on practical considerations: since it is not likely that any NLP system will in the foreseeable future become capable of handling unrestricted input -- even in the technical area(s) for which it might be designed — it is clear that a "fail-soft" technique is necessary. It is not obvious that such is possible in a system based solely on a pivot language; a hybrid system capable of dealing with shallower levels of understanding is necessary in a practical setting. This being the case, it seems better in near-term applications to start off with a system employing a "shallow" but usable level of analysis, and deepen the level of analysis as experience dictates, and theory plus project resources permit.

Our alternative is to have a `transfer´ component which maps "shallow analyses of sentences" in the SL into "shallow analyses of equivalent sentences" in the TL, from which synthesis then takes place. While we and the rest of the NLP community continue to explore the nature of an adequate pivot language (i.e., the nature of deep semantic models and the processing they entail), we can hopefully proceed to construct a usable system capable of progressive enhancement as linguistic theory becomes able to support deeper models.

Attached Translation Procedures

Our Transfer procedures (which effect the actual translation of SL into TL) are tightly bound to nodes in the analysis (parse tree) structure [Paxton, 1977]. They are, in effect, suspended procedures -- parts of the same procedures that constructed the corresponding parse tree nodes to begin with. We prefer this over a more general, loose association based on, e.g., syntactic structure because, aside from its advantage in sheer computational efficiency (search for structural transfer rules is eliminated), it eliminates the possibility that the "wrong" procedure can be applied to a construct. The only real argument against this technique, as we see it, is based on space considerations: to the extent that different constructs share the same transfer operations, wasteful replication of the procedures that implement said operations (and editing effort to modify them) is possible. We have not noticed this to be a problem. For a while, our system load-up procedure searched for duplicates of this nature and automatically eliminated them; however, the gains turned out to be minimal: different structures typically do require different operations.

Computational Techniques Employed

Again, our separation of "linguistic" from "computational" techniques is somewhat artificial, but nevertheless useful. In this section we present the reasons for our use of the following computational techniques: (a) a "some-paths," parallel, bottom-up parser; (b) associated rule-body procedures; (c) spelling correction; (d) another fail-soft analysis technique; and (e) recursive parsing of parenthetical expressions.

Some-paths, Parallel, Bottom-up Parser

Among all our choices of computational techniques, the use of a "some-paths," parallel, bottom-up parser is probably the most controversial. Our current parser operates on the sentence in a well-understood parallel, bottom-up fashion; however, the notion of "some-paths" will require some explanation. In the METAL system, the grammar rules are grouped into "levels" indexed numerically (0, 1, 2...), and the parser always applies rules at a lower level (e.g., 0) before applying any rules at a higher level (e.g., 1). Thus, the application of rules is partially ordered. Furthermore, once the parser has applied all rules at a given level it halts if there exist one or more "sentence" interpretations of the input; only if there are none does it apply more rules -- and then, it always starts back at level 0 (in case any rules at that level have been activated through the application of rules at a higher level, as can happen with a recursive grammar). Thus, the rule-application algorithm is Markov-like, and the system will not necessarily produce all interpretations of an input possible with the given rule base. Generally speaking, the lower-level rules are those most likely to lead to readings of an input sentence, and the higher-level rules are those least likely to be relevant (though they may be necessary for particular input sentences, in which case they will eventually be applied). As a result, the readings derived by our parser are the "most likely" readings (as judged by the linguists, who assign the rules to levels). This works very well in practice.

Our evolving choices of parsing methodologies have received our greatest experimental scrutiny. We have collected a substantial body of empirical evidence relating to parsing techniques and strategy variations. Since our evidence and conclusions would require lengthy discussion, and have received some attention elsewhere [Slocum, 1981], we will only state for the record that our use of a some-paths, parallel, bottom-up parser is justified based on our findings. First of all, all-paths parsers have certain desirable advantages over first-path parsers (discussed below); second, our some-paths parser (which is a variation on an all-paths technique) has displayed clear performance advantages over its predecessor technique: doubling the throughput rate while increasing the accuracy of the resulting translations. We justify our choice of technique as follows: first, the dreaded "exponential explosion" of processing time has not appeared, on the average (and our grammar and test texts are among the largest in the world), but instead processing time appears to be linear with sentence length -- even though our system may produce all possible readings; second, top-down parsing methods suffer inherent disadvantages in efficiency, and bottom-up parsers can be and have been augmented with "top-down filtering" to restrict the syntax rules applied to those that an all-paths top-down parser would apply; third, it is difficult to persuade a top-down parser to continue the analysis effort to the end of the sentence, when it blocks somewhere in the middle -- which makes the implementation of "fail-soft" techniques having production utility that much more difficult; and lastly, the lack of any strong notion of how to construct a "best-path" parser, coupled with the raw speed of well-implemented parsers, implies that a some-paths parser which scores interpretations and can continue the analysis to the end of the sentence, come what may, may be best in a contemporary application such as ours.

Associated Rule-body Procedures

We associate a procedure directly with each individual syntax rule, and evaluate it as soon as the parser determines the rule to be (seemingly) applicable [Pratt, 1973; Hendrix, 1978] -- hence the term "rule-body procedure". This practice is equivalent to what is done in ATN systems. From the linguist's point of view, the contents of our rule-body procedures appear to constitute a formal language dealing with syntactic and semantic features/values of nodes in the tree -- i.e., no knowledge of LISP is necessary to code effective procedures. Since these procedures are compiled into LISP, all the power of LISP is available as necessary. The chief linguist on our project, who has a vague knowledge of LISP, has employed OR and AND operators to a significant extent (we didn't bother to include them in the specifications of the formal language, though we obviously could have), and on rare occasions has resorted to using COND. No other calls to true LISP functions (as opposed to our formal operators, which are few and typically quite primitive) have seemed necessary, nor has this capability been requested, to date. The power of our rule-body procedures seems to lie in the choice of features/values that decorate the nodes, rather than the processing capabilities of the procedures themselves.

Spelling Correction

There are limitations and dangers to spelling correction in general, but we have found it to be an indispensable component of an applied system. People do make spelling and typographical errors, as is well known; even in "polished" documents they appear with surprising frequency (about every page or two, in our experience). Arguments by LISP programmers [re: INTERLISP's DWIM] aside, users of applied NLP systems distinctly dislike being confronted with requests for clarification -- or, worse, unnecessary failure -- in lieu of automated spelling correction. Spelling correction, therefore, is necessary.

Luckily, almost all such errors are treatable with simple techniques: single-letter additions, omissions, and mistakes, plus two- or three-letter transpositions account for almost all mistakes. Unfortunately, it is not infrequently the case that there is more than one way to "correct" a mistake (i.e., resulting in different corrected versions). Even a human cannot always determine the correct form in isolation, and for NLP systems it is even more difficult. There is yet another problem with automatic spelling correction: how much to correct. Given unlimited rein, any word can be "corrected" to any other. Clearly there must be limits, but what are they?

Our informal findings concerning how much one may safely "correct" in an application such as ours are these: the few errors that simple techniques have not handled are almost always bizarre (e.g., repeated syllables or larger portions of words) or highly unusual (e.g., blanks inserted within words); correction of more than a single error in a word is dangerous (it is better to treat the word as unknown, hence a noun); and "correction" of errors which have converted one word into another (valid in isolation) should not be tried.

Fail-soft Grammatical Analysis

In the event of failure to achieve a comprehensive analysis of the sentence, a system such as ours -- which is to be applied to hundreds of thousands of pages of text -- cannot indulge in the luxury of simply replying with an error message stating that the sentence cannot be interpreted. Such behavior is a significant problem, one which the NLP community has failed to come to grips with in any coherent fashion. There have, at least, been some forays. Weishedel and Black [1980] discuss techniques for interacting with the linguist/developer to identify insufficiencies in the grammar. This is fine for system development purposes. But, of course, in an applied system the user will be neither the developer nor a linguist, so this approach has no value in the field. Hayes and Mouradian [1981] discuss ways of allowing the parser to cope with ungrammatical utterances; such work is in its infancy, but it is stimulating nonetheless. We look forward to experimenting with similar techniques in our system.

What we require now, however, is a means of dealing with "ungrammatical" input (whether through the human's error or the shortcomings of our own rules) that is highly efficient, sufficiently general to account for a large, unknown range of such errors on its first and subsequent outings, and which can be implemented in a short period of time. We found just such a technique several years ago: a special procedure (invoked when the analysis effort has been carried through to the end of the sentence) searches through the parser's chart to find the shortest path from one end to the other; this path represents the fewest, longest-spanning phrases which were constructed during the analysis. Ties are broken by use of the standard scoring mechanism that provides each phrase in the analysis with a score, or plausibility measure (discussed earlier). We call this procedure `phrasal analysis'.

Our phrasal analysis technique has proven to be useful for both the developers and the end-users, in our application: the system translates each phrase individually, when a comprehensive sentence analysis is not available. The linguists use the results to pin-point missing (or faulty) rules. The users (who are professional translators, editing the MT system's output) have available the best translation possible under the circumstances, rather than no usable output of any kind. Phrasal analysis -- which is simple and independent of both language and grammar -- should prove useful in other applications of NLP technology; indeed, IBM's EPISTLE system [Miller et al., 1980] employs an almost identical technique [Jensen and Heidorn, 1982].

Recursive Parsing of Parenthetical Expressions

Few NLP systems have ever dealt with parenthetical expressions; but MT researchers know well that these constructs appear in abundance in technical texts. We deal with this phenomenon in the following way: rather than treating parentheses as lexical items, we make use of LISP's natural treatment of them as list delimiters, and treat the resulting sublists as individual "words" in the sentence; these "words" are "lexically analyzed" via recursive calls to the parser. Aside from the elegance of the treatment, this has the advantage that "ungrammatical" parenthetical expressions may undergo phrasal analysis and thus become single-phrase entities as far as the analysis of the encompassing sentence is concerned; thus, ungrammatical parenthetical expressions need not result in ungrammatical (hence poorly handled) sentences.

The Translation Component: METAL

This section presents a more detailed description of METAL -- the actual translation component of the LRC MT system. The top-level control structure is quite simple: the function TRANSLATE is invoked with a sentence in the SL (currently, German) and returns as its value an equivalent sentence in the TL (currently, English). TRANSLATE invokes three functions in succession: PARSE (for sentence analysis), TRANSFER (for structural translation), and GENERATE (for sentence synthesis). After sketching the format and content of dictionary entries, we will briefly discuss how the linguistic rules (lexicons and grammars) govern analysis, transfer, and synthesis, illustrating this three-step process using example sentences.

Dictionary Entries

METAL lexicons are divided into two types: monolingual, and bilingual (called "transfer"). A monolingual lexicon must be created for each of the languages involved in the translation process; transfer lexicons link the source- and target-language monolingual lexicons. Monolingual lexicons consist of entries for each lexical item. Each entry begins with a left parenthesis followed immediately by the canonical or "dictionary" form of the entry, then a series of feature labels, each with a sequence of zero or more values enclosed within parentheses. The entry is terminated by a right parenthesis. The entries for the German noun stem (NST) `Ausgabe´ and the corresponding English NST `output´ will serve as examples (see Figure 3).

Space contraints do not allow a full analysis of the entries; simply stated, each monolingual entry provides METAL with the information necessary for analysis and synthesis of the lexical items. In addition to entries for distinct word stems, the METAL monolingual lexicons contain separate lexical entries for such morphemes as prefixes, infixes, suffixes, and punctuation.

Transfer lexicons consist essentially of canonical word pairs which indicate the many-many correspondence between the SL and TL word stems. Each pair may be augmented by an arbitrary collection of context restrictions that must be met in order for the indicated translation to take place. A sample transfer entry for the pair "Ausgabe - output" is included in Figure 3; there are no restrictions (conditions) placed on this transfer [indicating the translation of `Ausgabe´ into `output´, or vice versa], other than the Subject Area tag [DP = Data Processing].

As an example of transfer restriction, it is possible to specify that a given German preposition corresponds to any of several English prepositions depending on the semantic type of its object noun. Four entries for the German preposition `vor´, shown in Figure 4, will illustrate. In these entries the appropriate English translation is defined by a restriction on semantic type (TY) and sometimes Grammatical Case (GC). These transfer entries are valid for ALL subject areas, but must be tried in a particular order (as evidenced by numeric "preference factors" in the entries). Thus the presence in context of an object noun of semantic TYpe other than ABStract, DURative, or PuNcTual results in the English translation `in front of´; else the presence of a Dative object noun of type ABStract or PuNcTual will result in the English translation `before´; else the presence of a Dative object noun of type DURative will result in the English translation `ago´ [which will later be postposed]; otherwise, the translation `in front of´ is chosen.

GERMAN monolingual entry:

```
(Ausgabe              CAT (NST)
   ALO   (Ausgabe)
   PLC   (WI)
   SNS   (1)
   TAG   (DP)
   CL    (P-N S-0)
   DR    (NP RD)
   FC    (PP)
   GD    (F)
   SX    (N)
   TY    (ABS DUR)
)
```

English monolingual entry:

```
(output               CAT (NST)
   ALO   (output)
   PLC   (WI)
   SNS   (1)
   TAG   (DP)
   CL    (P-S S-01)
   DR    (NP RD)
   FC    (PP)
   ON    (VO)
   SX    (N)
)
```

German-English Transfer entry:

```
(Ausgabe (NST DP) 0                      ! output (NST DP) 0                    )
(                                        +                                      )
```

Figure 3

German monolingual,
English monolingual,
and Transfer entries
for Ausgabe = output

```
(vor (PREP ALL) 30                    ! in_front_of (PREP ALL) 0         )
(    OPT TY * ABS DUR PNT             !                                  )
(                                     +                                  )

(vor (PREP ALL) 20                    ! before (PREP ALL) 0              )
(    GC D                             !                                  )
(    TY ABS PNT                       !                                  )
(                                     +                                  )

(vor (PREP ALL) 10                    ! ago (PREP ALL) 0                 )
(    GC D                             !                                  )
(    TY DUR                           !                                  )
(                                     +                                  )

(vor (PREP ALL) 0                     ! in_front_of (PREP ALL) 0         )
(                                     +                                  )
```

Figure 4

German-English Transfer entries
for vor = in front of, before, or ago

```
NN         NST          N-FLEX
0          1            2
(LVL 0)    (REQ WI)     (REQ WF)

TEST        (INT 1 CL 2 CL)

CONSTR      (CPX 1 ALO CL)
            (CPY 2 NU CA)
            (CPY 1 WI)

ENGLISH     (XFR 1)
            (ADF 1 ON)
            (CPY 1 MC DR)

SPANISH     ...
```

Figure 5

A German Context-free PS rule
for building a Noun STem + an
inflectional ending into a NouN

Analysis

For human-engineering reasons, one of the most convenient forms for expressing a grammar is via context-free phrase-structure rules. Context-free rules alone may or may not fully describe human language (see [Gazdar, 1983] for arguments that CF grammars are indeed sufficient), but, in any case, more general phrase-structure rules preclude efficient computational treatment, and CF rule-based systems seem to function as well as or better than any other technique, in practice. It has become traditional to augment the context-free rules by associating with them procedures in some programming language in order to provide more generative power, while maintaining computational tractability. In METAL, these "rule-body procedures" are invoked as soon as the parser finds a phrase matching their constituent phrase structure.

The traditional purpose of such procedures is to restrict the application of a rule by tests on syntax (e.g., number agreement between noun and verb) and/or semantics (e.g., whether the proposed syntactic subject can be interpreted as an agent); if such tests fail, the syntactic phrase is not built. In METAL, these procedures not only accept or reject rule application, but they also construct an interpretation of the phrase. Traditional parsers automatically build a "parse tree" and may add the output of such procedures as semantic information; in METAL, the parser (i.e., the LISP program) makes no commitment to a syntactic structure, but instead, linguistic procedures construct the interpretation (phrase) and compute its weight, or plausibility measure. The weight of a phrase is used when comparing it with any others that span the same sequence of words in order to identify the most likely reading.

A rule-body procedure in our system has several components: a constituent test part that checks the sons to ensure their utility in the current rule; an agreement TEST part to enforce syntactic and/or semantic correspondence among constituents; a phrase CONSTRuctor, which formulates the interpretation (phrase) defined by the current rule; and one or more Target-Language-specific transfer parts which operate during the second stage of translation (following complete sentence analysis). The inter-constituent test, the phrase constructor, and the transfer procedures may include calls to case frame procedures and/or transformations, as well as simpler routines to test and set syntactic and semantic features/values.

Case frames may apply semantic and syntactic agreement restrictions to the predicate (verb structure) and its arguments (noun and prepositional phrases) when constructing a clause. Each predicate′s lexical entry specifies its possible "central arguments." For German, the case frame will identify the case role-players according to voice (e.g., active) and mood (e.g., indicative) of the clause, and information about each potential argument such as its semantic type, form (noun phrase or prepositional phrase), and grammatical case (e.g., accusative) or prepositional marker. The restrictions can be general, or specific to the individual verb, preposition, and/or noun. The frame will fail, causing application of the clause rule to be rejected, if any of the restrictions are not met. Otherwise, case roles are assigned to the central arguments and the "peripheral arguments" are then identified.

The geometry of interpretations typically (though not always) parallels their original phrase structure. In other words, they are usually topologically equivalent to what the parser would produce if it were automatically constructing a tree. Some rules, however, incorporate transformations which may arbitrarily alter the phrase being constructed. The transformation module allows a linguist to specify a structural descriptor to any depth, to perform syntactic and/or semantic tests as in rule body procedures, and to specify a new structure into which the old is transformed. The transformation program attempts to match the "old" pattern descriptor with the currently instantiated phrase. If the match is successful, and the specified conditions are met, a new phrase is constructed using the "new" pattern descriptor, with the (old) matched phrase usually providing (most of) the structural contents, and constructor operations may further annotate the phrase with new features and/or values. The transformation module can have no effect on the parsing algorithm, whatever the outcome of its application, unless the rule is written so that failure to complete a transformation causes the interpretation to be rejected; in such a case, only the fact of the rejection has an effect on the parser: it abandons that search path, just as it would if any other condition in the rule-body procedure were unsatisfied.

A grammar in METAL consists of a number of partially-ordered (LeVeLled), augmented phrase-structure syntax rules, plus a collection of indexed transformations. A relatively simple PS rule for building nouns will be used to illustrate the parts and format of METAL grammar rules (see Figure 5).

The first line consists of a left-hand element, the "father" node (here, NN), and one or more right-hand elements -- the "sons" (here, NST and N-FLEX). In the example rule, the left-hand element is the noun (NN) node and the right-hand elements are the noun stem (NST) and the nominal ending (N-FLEX) nodes. The second line enumerates the elements (from 0 to n) for reference in the rule-body procedure. Each constituent may have individual conditions, called `column tests´, to restrict exactly what elements fit the rule. If any column test fails, the grammar rule will fail -- i.e., the parser will abandon its attempt to apply this rule. In this example, the column test for the first element (NST) requires it to be word-initial (WI) -- i.e., preceded by a blank space in the matrix sentence; the column test for the second element (N-FLEX) requires it to be word-final (WF) -- i.e., followed by a blank space.

In addition to the column tests, which apply only to single elements, each rule has a TEST part that states agreement restrictions between the right-hand elements. Failure of any agreement test will also result in failure of the entire rule. In the example rule, the single agreement test states that there must be an intersection (INT) of the inflectional class (CL) values for the two constituents; i.e., the values for the feature CL coded on the NST and the N-FLEX are compared to insure that they have at least one value in common.

Only after all conditions have been satisfied is it possible for METAL to build the appropriate syntax tree. This is done in the CONSTR part of the rule, which can also add or copy information in the form of features and values from the sons to the father. In the example rule, the CONSTRuctor (by not applying a transformation) would produce the tree represented below:

```
          NN
          |
    +-----+-----+
    |           |
   NST        N-FLEX
```

In the example rule-body procedure, the CONSTRuctor will copy all features with their associated values from the first element (i.e., the NST), except for the allomorph (ALO) and inflectional class (CL) features, using the operation CPX. CONSTR in this rule will also copy (CPY) the grammatical number (NU) and case (CA) features from the second constituent (the N-FLEX), and the word initial (WI) feature from the first constituent (the NST).

Transformations may be applied in the TEST, CONSTR, and/or Transfer portions of grammar rules. These range from simple movement and deletion operations to highly complex transformations which add structure, perform tests, etc. The following exemplifies a simple movement transformation:

```
(XFM (&:1 (&:2 &:3))
     (&:1 (&:3 &:2)))
```

This transformation simply exchanges the two sons (#2 and #3) of the current node (#1): each ampersand represents one and only one constituent, or node.

Determining whether a sequence of words constitutes a clause is handled by a case frame, which is invoked in the TEST portion of clause-level rules. Simply stated, the case frame uses the argument information coded on the verb stem's lexical entry to identify its arguments, perform agreement tests, and label those arguments. In METAL, an argument may be a noun phrase, prepositional phrase, or adverbial phrase, depending on the verb. For a more detailed discussion of the grammar or lexicon, see [Bennett, 1982].

Transfer

The purpose of the TRANSFER module is to restructure the most plausible interpretation of the SL sentence into an interpretation of an equivalent sentence in the TL(s). Every non-terminal node (phrase) in every sentence interpretation has attached to it the "suspended" rule-body procedure that originally created it; this eliminates the need to search through a monolithic "transfer grammar" for a matching pattern or routine -- and also eliminates the danger of inadvertently applying an inappropriate pattern or routine that happened to match (part of) the same structure. The suspended procedure associated with the root phrase in the most plausible interpretation is (re)invoked by TRANSFER. The appropriate Target-Language-specific Transfer part of a rule-body procedure can recursively transfer all or some of the node's sons (i.e., its non-terminal constituents) in any order, apply transformations, and/or lexically transfer a terminal son. Lexical transfer replaces a SL canonical form with a TL canonical form using the appropriate transfer lexicon; this process may be sensitive to sentential context. The TL stem is created and appropriate suffixes are added to create the proper TL word. Features in TL lexical entries may be used to help select the proper sense (i.e., word).

The final parts of a grammar rule are the Transfer sections [in Figure 5, ENGLISH and SPANISH]. In the multi-lingual METAL system, there is a separate transfer section for each Target Language. (In our English analysis grammar, there are, e.g., GERMAN and CHINESE sections, which means that METAL can translate bi-directionally as well as into multiple languages.) The appropriate Transfer section(s) are individually invoked only after a sentence [S] has been analyzed, at which point the system will perform the Transfer operations specified, generally moving down the tree to the terminal nodes where lexical substitution takes place. In our example rule (Figure 5), the first operation is

(XFR 1)

which causes the system to recursively invoke TRANSFER on the first son (i.e., the NST). Because the NST happens to be a terminal (lexical) node, it will be translated using the appropriate Transfer entry. The remaining two operations (ADF and CPY) are performed as the system ascends the tree. Thus, while analysis generally proceeds bottom-up, transfer proceeds top-down. At each node in the tree, all nodes below are accessable for reading (to determine context) and writing (to pass down information necessary for proper transfer).

Transfer in METAL is not a particularly simple process. Consider the following sentence pair:

German:	die auszugebenden Resultate
gloss:	the to-be-output results
English:	the results to be output

Here, the German participial verb form must be post-posed in English; a transformation (conditioned on the form of the participial phrase) must be employed in cases like these. Prepositions present notorious problems; they must be translated and positioned with respect to their object NPs at least:

German:	vor diesem Haus
English:	in front of this house

German:	vor dieser Woche
English:	before this week

German:	vor einer Woche
gloss:	ago one week
English:	one week ago

Clearly the relationship is complex: both the German noun (i.e., its semantic type) and its determiner (if any) influence the selection of a suitable English translation, as well as its position in the phrase.

A TL verb case frame, when applied during the transfer phase, will order the case role-fillers as required by the verb based on voice, mood, etc. The syntactic form of the central arguments is chosen and, if necessary, prepositions are introduced as specified in the Transfer verb entry. Consider the following examples:

German:	aus Gold besteht die Tuer
gloss:	of gold consists the door
English:	the door consists of gold

German:	auf Gold besteht der Mann
gloss:	on gold insists the man
English:	the man insists on gold

Here, it is not only true that the complements must be re-ordered in English, but it is also necessary to translate the verb-preposition combination as a unit. This, in turn, may reflect on [help disambiguate] the semantic type of the matrix-subject, as the following examples illustrate:

German:	aus Gold besteht er
gloss:	of gold consists [it]
English:	it consists of gold

German:	auf Gold besteht er
gloss:	on gold insists [he]
English:	he insists on gold

Various of these factors can and do interact, as illustrated by the following example:

German:	die aus Gold bestehende Tuer
gloss:	the of gold consisting door
English:	the door consisting of Gold

In the METAL system, the Transfer procedures attached to analysis rules interact with complex Transfer lexical entries to determine the proper form and wording of the Target-Language structures. Generally speaking, each node appearing in an analysis tree is responsible for producing its appropriate translation, in context. (This is not always true, since a higher-level node can usurp the function of one or more of its sons -- either performing transfer directly, or assigning a new transfer procedure to be executed in place of the original.) We have found this combination of techniques (lexical transfer interacting with grammatical structural transfer procedures) to be a flexible and powerful tool that facilitates high-quality translation.

The top-level node (phrase) in the newly constructed TL tree is eventually returned by TRANSFER as its functional value, and this in turn is used for synthesis.

Synthesis

The GENERATE function synthesizes the translation by simply taking the TL tree produced by TRANSFER, and inflecting and appending together all of the lexical allomorphs (words and their inflections) located in its terminal nodes. The value of the function GENERATE is a sentence; it is returned to the function TRANSLATE, which returns that sentence as its functional value. For synthesis into multiple Target Languages, transfer and synthesis (but not analysis) may be invoked multiple times.

The METAL Parser

The parser -- the LISP program that interprets a sentence according to the linguistic rules -- is the heart of METAL. If the parser is inefficient, the analysis effort will consume far too much space and time to be of practical benefit. The current METAL parser is a variation on the Left-Corner parallel, bottom-up parsing algorithm [Chester, 1980]. During an extensive series of experiments comparing a dozen parsers on the basis of their practical performance characteristics [Slocum, 1981], a Left-Corner parser augmented by top-down filtering [Pratt, 1973], which closely resembles an Earley parser, was determined to be the most efficient, and soon replaced the previous METAL parser based on the Cocke-Kasami-Younger algorithm [Aho and Ullman, 1972]. However, the newest, "some-paths" implementation of the Left-Corner parser has since proven to be even more efficient than the fast Left-Corner parser.

The LRC parsers work with a special data structure called a "chart" that records the complete state of an analysis at every point in that analysis. Roughly speaking, the parser starts by adding every word in the sentence to the chart; it then draws grammatical inferences from those additions. The grammatical inferences are of two varieties: (1) interpretations of the syntax rules found to apply to the current portion of the input, referred to as instantiated phrases; and (2) predictions about what types of phrases (what grammatical categories) may appear next in the input. After the parser has drawn all possible grammatical inferences on a given "level,", the chart is examined for phrases which span the sentence, and whose syntactic categories appear in the user-definable list of acceptable ROOTCATEGORIES: usually `(S)´. These phrases constitute the interpretations of the input sentence; if there are none, the parser reverts to the lowest LeVeL for which there are pending rules, and continues.

In order for the parser to add a word to the chart, the word must be lexically analyzed. There are three ways to do this: (1) the word may appear in the dictionary as an entry; (2) the word may be decomposed into a sequence of morphemes, each of which appears in the dictionary; or (3) the word may have a lexical entry generated on-the-fly. In METAL, any combination of the three is possible. Words, or sequences of letters that appear to be words, are looked up in the dictionary; independently, an attempt is made to decompose each word into an acceptable sequence of morphemes, each of which appears in the dictionary. (In METAL, the dictionary is composed of lexical entries in the usual sense, plus any literals appearing in phrase-structure grammar rules.) Lexical entries for numbers are automatically generated. Definitions for unknown words and non-words are also generated. Parenthetical expressions are "lexically analyzed" via a recursive call to the parser: each is parsed as if it were a complete sentence, then its interpretations are automatically transformed into "lexical entries" for incorporation into the analysis of the encompassing sentence. Unknown words may be decomposed into sequences of known words and other lexical items (numbers, acronyms, etc.), especially if these are flagged by punctuation marks (e.g., hyphens, slashes) in the input. For example, the German "word" `10mal´ may be decomposed into `10´ and `mal´ [in English, `times´].

Morphological analysis is a relatively simple process, relying on a letter tree [discrimination net] to indicate the legal transitions from character to character in known morphemes (defined via lexical entries or as literals in syntax rules). For a highly synthetic language like German, this tree is searched recursively to discover successive morphemes in a word. As a bonus, METAL includes a program capable of correcting the most common spelling and typographic errors (deletion, substitution, addition, wrong case, and transposition); thus typical transcription errors pose no problem. The results of morphological analysis can be ambiguous in many ways: morph sense, morph category, and even morph boundaries may be indeterminate. The parser (using the phrase structure rules) sorts out the ambiguities according to word and sentential context, as a natural part of its operation -- i.e., lexical disambiguation (including homograph resolution) is not a process distinct from other forms of disambiguation (e.g., syntactic).

As PARSE finds phrase-structure rules that are applicable to a current sequence of morphs, words, and/or phrases in the ongoing analysis, it does not automatically build a syntactic structure expressing this fact; instead, it invokes a special routine which is responsible for determining (through the invocation of the rule-body procedure) the applicability of the rule, and for constructing and scoring the interpretation. This special routine constructs a preliminary parse tree, invokes the rule-body procedure to determine if the rule is applicable (and possibly to annotate the tree and/or transform it), and if the resulting interpretation is acceptable it scores the phrase based on the scores of its constituents and any preference assigned by the rule-body procedure; it rejects the interpretation if its score falls below cutoff, else attaches the "suspended" rule-body procedure to the phrase (important in the transfer phase, as explained elsewhere). The scores of the root nodes in the sentence analyses will be used later to determine the "most plausible analysis" for transfer and synthesis.

If the METAL parser fails to achieve a unified interpretation of a sentence (or of a parenthetical expression, which is recursively parsed as if it were a sentence), it attempts to "fake" an analysis of the sentence. A phrasal analysis is constructed from the fewest, largest, highest-scoring phrases that together span the input sentence. An S phrase is built just as if there were a grammar rule with the discovered phrases listed in its right-hand-side. A default rule-body procedure is attached, to be invoked during the transfer phase; this procedure will simply use the (XFR) operator to invoke TRANSFER on the constituent phrases of the dummy S phrase just built.

Recent Experimental Results

In the last four years, METAL has been applied to the translation into English of over 1,000 pages of German telecommunication and data processing texts. To date, no definitive comparisons of METAL translations with human translations have been attempted; this situation will soon be remedied. However, some stimulating quantitative and qualitative statistics have been gathered.

Quantitative Results

On our Symbolics LM-2 Lisp Machine, with 256K words of physical memory, preliminary measurements indicate an average performance of 5-6 seconds (real time) per input word; this is already 6 times the speed of a human translator, for like material. The paging rate indicates that, with added memory, we could expect a significant boost in this performance ratio. With a faster, second-generation Lisp Machine, we would expect another substantial reduction of real-time processing requirements: preliminary measurements (before system tuning) show a doubling of the throughput rate; further speed increases are anticipated.

Qualitative Results

Measuring translation quality is a vexing problem -- a problem not exclusive to machine translation or technical texts, to be sure. In evaluating claims of "high-quality" MT, one must carefully consider how `quality´ is defined; "percentage of words [or sentences] correct [or acceptable]", for example, requires definition of the operative word, `correct´. A closely related question is that of who determines correctness. Acceptability is ultimately defined by the user, according to his particular needs: what is acceptable to one user in one situation may be quite unacceptable in another situation, or to another user in the same situation. For example, some professional post-editors have candidly informed us that they actually look forward to editing MT output because they "can have more control over the result." For sociological reasons, there seems to be only so much that they dare change in human translations; but as everyone knows (and our informants pointed out), "the machine doesn´t care." The clear implication here is that "correctness" has traditionally suffered where human translation is concerned; or, alternately, that "acceptability" depends in part on the relationship between the translator and the revisor. Either way, judgements of "correctness" or "acceptability" by translators and editors is likely to be more harsh when directed toward MT than when directed toward human translation (HT). It is not yet clear what the full implications of this situation are, but the general import should be of some concern to the MT community.

For different (and obvious) reasons, qualitative assessments by MT system vendors are subject to bias -- generally unintentional -- and must be treated with caution. But one must also consider other circumstances under which the measurement experiment is conducted: whether (and for how long, and in what form) the text being translated, and/or its vocabulary, was made available to the vendor before the experiment; whether the MT system was previously exercised on that text, or similar texts; etc. At the LRC, we conduct two kinds of measurement experiments: "blind", and "follow-up". When a new text is acquired from the project sponsor, its vocabulary is extracted by various lexical analysis procedures and given to the lexicographers. This may include a partial or full concordance of the text, in which each word is displayed in a context that includes the full matrix sentence. The lexicographers then write ("code") entries for any novel words discovered in the text. The linguistic staff never sees the text prior to a blind experiment. Once the results of the blind translation are in, the project staff are free to update the grammar rules and lexical entries according to what is learned from the test, and may try out their revisions on sample sentences from the text. Some time later the same text is translated again, so that some idea of the amount of improvement can be obtained.

In addition to collecting some machine performance statistics, we count the number of "correct" sentence translations and divide by the total number of sentence units in the text, in order to arrive at a "correctness" figure. (For our purposes, "correct" is defined as "noted to be unchanged for morphological, syntactic, or semantic reasons, with respect to the original machine translation, after revision [by professional post-editors] is complete." (Non-essential stylistic changes are not considered to be errors.) In the course of experimenting with over 1,000 pages of text in the last four years, our "correctness" figures have varied from 45% to 85% (of full-sentence units) depending on the individual text and whether the experiment was of the "blind" or "follow-up" variety.

Cost-effectiveness

The single numerical assessment of greatest interest in the MT community is almost certainly cost-effectiveness. Until METAL is evaluated by unbiased third parties, taking into account the full costs of translation and revision using METAL vs. conventional (human) techniques, the question of METAL's cost-effectiveness cannot adequately be addressed. However, we have identified some performance parameters that are interesting. Our sponsor has calculated that METAL should prove cost-effective if it can be implemented on a second-Generation Lisp Machine supporting 4-6 post-editors who can sustain an average total output of about 60 revised pages/day. At 275 words/page, and 8 hours/day, this works out to 1.7 seconds/word, minimum real-time machine performance. If the new second-generation Lisp Machines are, as generally claimed, three times as fast as the current generation (represented by our LM-2), then our immediate target is 5.2 seconds/word, minimum real-time performance -- about what we now experience on our LM-2. If this level of performance can be sustained while maintaining a high enough standard of quality that an individual revisor can handle 10-15 pages/day, METAL will have achieved cost-effectiveness.

We have also measured revision performance: the amount of time required to edit texts translated by METAL. In the first such experiment, conducted late in 1982, two Siemens post-editors revised METAL's translations at the rate of 15-17 pages/day (depending on the particular editor). In a second experiment, conducted in mid-1983, the rates were only slightly higher (15-20 pages/day), but the revisors nevertheless reported a significant improvement in their subjective impression of the quality of the output. In a third experiment, conducted in early 1984, the revisors reported further improvement in their subjective impression of the quality of the output, and their revision rates were much higher: around 29 pages/day. Thus, our experimental performance figures indicate that we may already have reached the goal of cost-effectiveness.

Conclusions

MT Research at the Linguistics Research Center involves the selection and testing of Natural Language Processing techniques in a real-world environment. With an efficient computational component such as we now have, it becomes possible to empirically validate new linguistic theories as they are proposed. Our research can therefore answer questions about their extensibility, and the limits of their application.

Some "old" Natural Language Processing techniques are producing surprisingly good results, and new ones are being developed. Others have not proven to be effective, and have been abandoned. METAL is capable of producing useful English translations for a wide variety of German sentences; translation from English into German has recently begun. However, further development is currently underway to resolve a number of remaining problems. Two areas in which the LRC is making improvements in the METAL linguistic component are the treatment and placement of adverbials, and more extensive use of semantics.

In the future, we look forward to the development, phased introduction, and empirical assessment of more advanced NLP techniques -- especially w.r.t. anaphora resolution and the use of semantic models. We see no evidence that today's advanced but experimental NLP techniques will soon [in this decade, or even century] be able to supplant the more primitive techniques that are currently effective in a large-scale application such as ours. But we nevertheless hope that such techniques can, even if in elementary form, be effectively utilized within practical applications of current techniques to further improve the overall quality and cost-effectiveness of translation. We have tried to anticipate this eventuality in the design of the LRC MT system by allowing for future evolution, possibly even revolution. In the process of applying advanced NLP techniques to practical problems in real-world settings, such as translation, we fully expect the feedback of experience to substantially influence both the form and content of linguistic theories.

References

Aho, A. V., and J. Ullman. The Theory of Parsing, Translation and Compiling, Vol. 1. Prentice-Hall, New Jersey, 1972.

Bennett, W. S., "The Linguistic Component of METAL," Working Paper LRC-82-2, Linguistics Research Center, University of Texas, July 1982.

Boitet, Ch., P. Chatelin, and P. Daun Fraga, "Present and Future Paradigms in the Automatized Translation of Languages," Proceedings of the 8th International Conference on Computational Linguistics, Tokyo, Sept. 30 - Oct. 4, 1980[a].

Boitet, Ch., and N. Nedobejkine, "Russian-French at GETA: Outline of the Method and Detailed Example," Proceedings of the 8th International Conference on Computational Linguistics, Tokyo, Sept. 30 - Oct. 4, 1980[b].

Bresnan, J. W., "A Realistic Transformations Grammar," in Halle, Bresnan, and Miller (eds.), Linguistic Theory and Psychological Reality. MIT Press, Cambridge, Mass., 1977.

Carbonnel, J., R. E. Cullingford, and A. V. Gershman, "Knowledge-Based Machine Translation," Research Report #146, Dept. of Computer Science, Yale University, Dec. 1978.

Cullingford, R. E., "Script Application: Computer Understanding of Newspaper Stories," Research Report #116, CS Dept., Yale University, 1978.

Damerau, F. J., "Operating Statistics for the Transformational Question Answering System," AJCL 7 (1), January-March 1981, pp. 30-42.

Gazdar, G., "Unbounded Dependencies and Coordinate Structure," in Linguistic Inquiry, 12 (2), Spring 1981, pp. 155-184.

Gazdar, G., "Phrase Structure Grammars and Natural Languages," Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, 8-12 August 1983, vol. 1, pp. 556-565.

Hayes, P. J., and G. V. Mouradian, "Flexible Parsing," American Journal of Computational Linguistics, Vol. 7, No. 4, (Oct-Dec) 1981, pp. 232-242.

Heidorn, G.E., K. Jensen, L. A. Miller, R. J. Byrd, and M. S. Chodorow, "The EPISTLE text-critiquing system," IBM System s Journal 21 (3), 1982.

Hendrix, G. G., E. D. Sacerdoti, D. Sagalowicz, and J. Slocum, "Developing a Natural Language Interface to Complex Data," ACM Transactions on Database Systems, Vol. 3, No. 2, (June) 1978, pp. 105-147.

Jensen, K., and G. E. Heidorn, "The Fitted Parse: 100% Parsing Capability in a Syntactic Grammar of English," Research Report RC-9729 (#42958), Computer Sciences Department, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, December 1982.

Kay, M., "The Proper Place of Men and Machines in Language and Translation," Technical Report, Xerox PARC, Palo Alto, California, 1980.

King, M., "Design Characteristics of a Machine Translation System," Proc. 7th IJCAI, Vancouver, B.C., Canada, Aug. 1981, v. 1, pp. 43-46.

Lehmann, W. P., W. S. Bennett, J. Slocum, et al., "The METAL System," Final Technical Report RADC-TR-80-374, Rome Air Development Center, Griffiss AFB, New York, January 1981. Available as Report AO-97896, National Technical Information Service, U.S. Department of Commerce, Springfield, Va.

Martin, W. A., K. W. Church, and R. S. Patil, "Preliminary Analysis of a Breadth-First Parsing Algorithm: Theoretical and experimental results," paper presented at the University of Texas Symposium on Modelling Human Parsing Strategies, 24-26 March 1981.

Miller, L. A., G. E. Heidorn, and K. Jensen, "Text-Critiquing with the EPISTLE System: An Author's Aid to Better Syntax," Research Report RC 8601 (#37554), Behavioral Sciences and Linguistics Group, Computer Sciences Dept., IBM Thomas J. Watson Research Center, Yorktown Heights, New York, December 1980.

Paxton, W. H., "A Framework for Speech Understanding," Tech. Note 142, AI Center, SRI International, Menlo Park, California, June 1977.

Petrick, S. R., "Transformational Analysis," in R. Rustin (ed.), Natural Language Processing. Algorithmics Press, New York, 1973, pp. 27-41.

Pratt, V. R., "A Linguistics Oriented Programming Language," Proc. 3rd IJCAI, Stanford University, California, August 1973, pp. 372-381.

Pratt, V. R., "LINGOL - A Progress Report," Advance Papers of the Fourth Internationsal Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR, 3-8 Sept. 1975, pp. 422-28.

Robinson, J. J., "DIAGRAM: A Grammar for Dialogues," CACM 25 (1), Jan. 1982.

Sager, N. Natural Language Information Processing. Addison-Wesley, Reading, Massachusetts, 1981.

Slocum, J., "An Experiment in Machine Translation," Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics, Philadelphia, 19-22 June 1980, pp. 163-167.

Slocum, J., "A Practical Comparison of Parsing Strategies for Machine Translation and Other Natural Language Processing Purposes," University Microfilms International, Ann Arbor, Mich., 1981.

Slocum, J., and W. S. Bennett, "The LRC Machine Translation System: An Application of State-of-the-Art Text and Natural Language Processing Techniques to the Translation of Technical Manuals," Working Paper LRC-82-1, Linguistics Research Center, University of Texas, July 1982.

Small, S., "Word Expert Parsing: A Theory of Distributed Word-Based Natural Language Understanding," Tech. Rep. 954, CS Dept., Univ. of Maryland, 1980.

Weischedel, R. M., and J. E. Black, "If the Parser Fails," Proceedings of the 18th Annual Meeting of the ACL, Univ. of Pennsylvania, June 19-22, 1980.

Winograd, T. Understanding Natural Language. Academic Press, New York, 1972.

Woods, W. A., "Transition Network Grammars for Natural Language Analysis," CACM, 13 (10), October 1970, pp. 591-606.

Woods, W. A., "Syntax, Semantics, and Speech," BBN Report 3067, Bolt, Beranek, and Newman, Inc., Cambridge, Massachusetts, April 1975.

Woods, W. A., "Cascaded ATN Grammars," AJCL, 6 (1), January-March 1980, pp. 1-12.

Further Reading

Hutchins, W. J., "Progress in Documentation: Machine Translation and Machine-aided Translation," Journal of Documentation, 34 (2), June 1978, pp. 119-159.

Vauquois, B. La Traduction Automatique a Grenoble. Dunod, Paris, 1975.

Languages and Machines: Computers in Translation and Linguistics. A report by the Automatic Language Processing Advisory Committee [ALPAC], Division of Behavioral Sciences, National Academy of Sciences, National Research Council, Publication 1416, Washington, D.C., 1966.