The LRC Machine Translation System: An Application of State-of-the-Art Text and Natural Language Processing Techniques to the Translation of Technical Manuals

Jonathan Slocum
Siemens Corporation

and Winfield S. Bennett

Working Paper LRC-82-1
July, 1982

ABSTRACT

This paper describes the prototype Machine Translation system developed at the Linguistics Research Center of the University of Texas. The LRC MT 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, the principles guiding our MT system design, a general description of the MT system, a more detailed description of METAL, notes on the METAL parser, a discussion of the system's technical merits, the results of some extensive experiments in translating German telecommunications texts into English, and conclusions regarding our contribution to the science of Computational Linguistics.

This paper is a composite of two papers presented separately by the authors at the 9th International Conference on Computational Linguistics (COLING 82), held in Prague, during July 5-10, 1982. Due to mail delays, it did not appear in the Proceedings.

The LRC Machine Translation System: An Application of State-of-the-Art Text and Natural Language Processing Techniques to the Translation of Technical Manuals

Jonathan Slocum
Siemens Corporation

and Winfield S. Bennett

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 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.

This paper describes the prototype MT system developed by the LRC; it represents a rare large-scale application of state-of-the-art Natural Language Processing techniques. The LRC MT system includes a translation program, METAL, as part of a suite of programs designed to automate the complete process of translating technical texts. Since 1980, METAL has been implemented in INTERLISP and run on a DEC 2060. As the COLING 82 conference convenes, the LRC MT system is being implemented on a much less expensive, personal minicomputer: a Symbolics LM-2 (Lisp Machine).

Sections of this paper will deal with our domain of application, the principles guiding our MT system design, a general description of the MT system, a more detailed description of METAL, notes on the METAL parser, a discussion of the system's technical merits, the results of some extensive experiments in translating German telecommunications 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 newspaper stories, 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.

Principles of Advanced MT

As the LRC MT system has been developed, our goal has been to observe certain design principles. Here we briefly discuss the design goals and how the LRC MT system adheres to them.

Separation of Program from Linguistic Theory

The LRC MT system clearly separates the program component from the linguistic component. Linguistic theory is not static. The linguistic theory on which an MT system is based at any given time must be able to undergo further improvement with little or no impact on the computer programs implementing, or interpreting, that theory. So, for example, the grammar rules which describe the languages covered by our system are modifiable without concern for the programs that utilize the rules for translation. Similarly, computational procedures may be found amenable to improvement, and the LRC MT system programmers are free to change their component with no undue restrictions imposed by the linguistic component.

Modularity within Components

The requirement of modularity is the sine qua non of flexibility. In the LRC parser, for example, one routine is responsible for morphological analysis of words; another, for idiomatic analysis; another, for application of syntax rules; yet another, for application of transformations; and so forth. The observance of modularity is not confined to the programs alone, but applied to the linguistic component as well: for example, the grammar rules are segmented into two types of acceptance test procedures (one for individual constituents, one for collections of constituents), and into two types of operations (one for analysis [independent of target language], one for transfer). Because of careful attention to separation of responsibility in this manner, our system is easy to modify and extend in accordance with the dictates of experience. In the LRC MT system, evolution is provided for so that the system is not rendered obsolete by its own design.

Linguistic Rule Base

The LRC MT system grammars and lexicons are maintained in a format optimized for use by linguists, rather than by METAL or its programmers. The issue of overall efficiency in research, development, and application precludes interest in machine efficiency alone. Machine Translation is an exceedingly difficult problem whose optimal solution is not yet well understood. Empirical results can and will dictate that linguistic procedures be changed. For this to be effected by linguists who are not sophisticated in the computer arts, our rule base is expressed in a formalism with which they are familiar.

Minimal Constraints on Representation

The program component imposes no significant constraint on how the linguistic component represents interpretations of sentences. The most common representation formalism in modern linguistics, for example, is syntax trees; in related disciplines, other formalisms are preferred. In order to allow freedom of choice, a few specialized routines are written for each desired representation; the LRC parser interfaces with these modules in a well-defined manner. Thus the linguists may change their representation formalism at will.

"Fail-soft" Mechanisms

One great drawback of almost all natural language processing systems has been their fragility. When confronted by a sentence beyond the descriptions provided by the rule base, analysis usually terminates; most systems that have any sort of "intelligent error recovery" at all provide only spelling correction. Some recent research has focused on how parsers might interact with the user when blocks occur. This is certainly interesting, but hardly relevant to an applied MT system, where the users are neither linguists nor programmers, and where any interaction with the user during translation runs the risk of degrading performance to an unacceptable level. Since coverage deficiencies will arise in any collection of fixed rules, an MT system must incorporate provisions for overcoming them. In the LRC MT system, a sentence which cannot be analyzed as a unit is analyzed into the lowest possible number of phrases or, that seeming unreasonable, into a sequence of technical terms; these are translated individually. In addition, the interpretation-rejection mechanism is implemented to allow "relaxation" of conditions and restrictions in case the rule base fails to account for a "grammatical" sentence, or in case of an error in the input sentence (such as subject-verb number disagreement).

Extension to New Languages

The LRC MT system admits extension through the addition of new languages; the work involves little if any more than writing grammars and lexicons for the new language. The framework in which our linguistic theory is formulated and expressed is able to account for languages other than the ones to which it is now being applied (specifically, for most, if not all, languages in the Indo-European family). Historically, attempts to take a system designed for translation from/into one language and modify it for translation from/into another have not been notably successful. The reason for this is in part due to the typical lack of extensibility built into the fabric of MT systems.

Multi-lingual Translation

For some applications it is desirable to translate a text into not just one but several languages. Typically, the amount of time spent analyzing a text greatly exceeds that spent synthesizing its Target Language (TL) equivalent, so that a system like ours that is able to translate into several languages following a single analysis has a decided practical advantage. This may even have a theoretical advantage insofar as such practice counteracts a tendency to produce a grammar that analyzes a Source Language (SL) only to the extent required to translate into a particular TL. Some systems have been constructed ab initio using a single-target strategy; the usual result is a complete inability to translate into any other language without a total revision of the system.

A Framework for Application

If a new theory of MT is proposed, and is claimed to advance the state of the art, it cannot indulge in the luxury of confining its attention to isolated problems in small texts: (1) it is usually the case that attempts at large-scale application reveal striking deficiencies in methods that work well in small-scale experiments; (2) some proposals for MT, while perhaps workable in theory, clearly require access to encyclopedic knowledge which may not be

available in appropriate form for another century -- and certainly not in this century. To some extent this can be regarded as indicative of the ultimate difficulty of translation. Nevertheless, any proposed advance in MT today must address the problems encountered in production translation; in doing so, the theory will benefit considerably through refinement in a real-world environment. Among other things, this implies a serious concern for efficiency in the underlying programs. It also implies a means for resolving the text-processing problems confronting any MT system. The LRC MT system has been used to translate moderately large amounts of text -- extremely large, by Artificial Intelligence (AI) standards -- and shows every sign of being efficient enough to serve as a framework for direct application.

A Framework for Research

No system today or in the near future will constitute an optimum solution to the MT problem. Instead, it will at best constitute an implementation of the most highly developed practical linguistic and computational theories of translation. Both kinds of theory will continue to evolve, and both would benefit from large-scale testing. Since an efficient, applied MT system would be a prime vehicle for such testing, it seems only reasonable to require it to support a research function. The LRC MT system has always served as a framework for front-line research in machine translation.

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. Generally speaking, 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 such errors. Therefore, the LRC MT system includes a validation program that, working from a formal specification of what is and is not legal in lexical entries, identifies any errors of format and/or syntax within each submitted entry. The formal specification is organized by language, by grammatical 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.

Finally, there is 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 accomodate 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; therefore, changes made in the database are instantly reflected in translations, resulting in a rapid turnaround that both encourages and enhances research and development. See Figure 1.

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

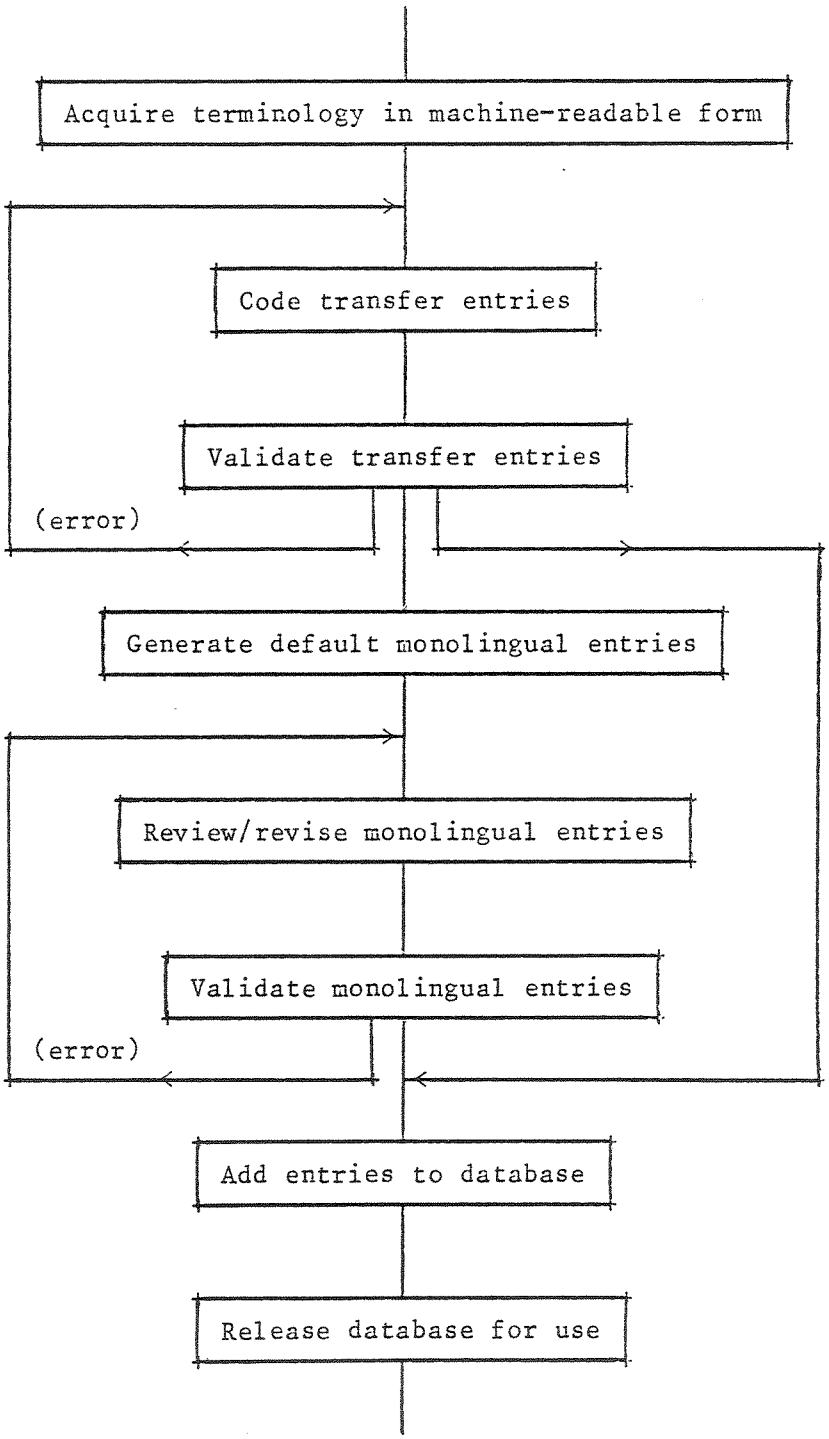


Figure 1

Coding Lexical Entries

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.

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 though 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.

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 lexical 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, proposed spelling corrections can be noted for human review, and suspected acronyms can be verified. As another potential advantage, a "lemmatized concordance" can be constructed, in which each word is indexed according to its root, rather than inflected, form. As a final advantage, a dictionary restricted to the text at hand could be constructed if one wished to reduce dictionary storage requirements in the MT system. After a human has checked any textual errors noted during lexical pre-analysis, translation may proceed.

Translation, briefly stated, involves reading in the Source Language (SL) file 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 unnecessary frustration and expense, and can constitute a significant cost factor in translation. The alternative of providing unformatted translations to the user is distinctly unattractive. In

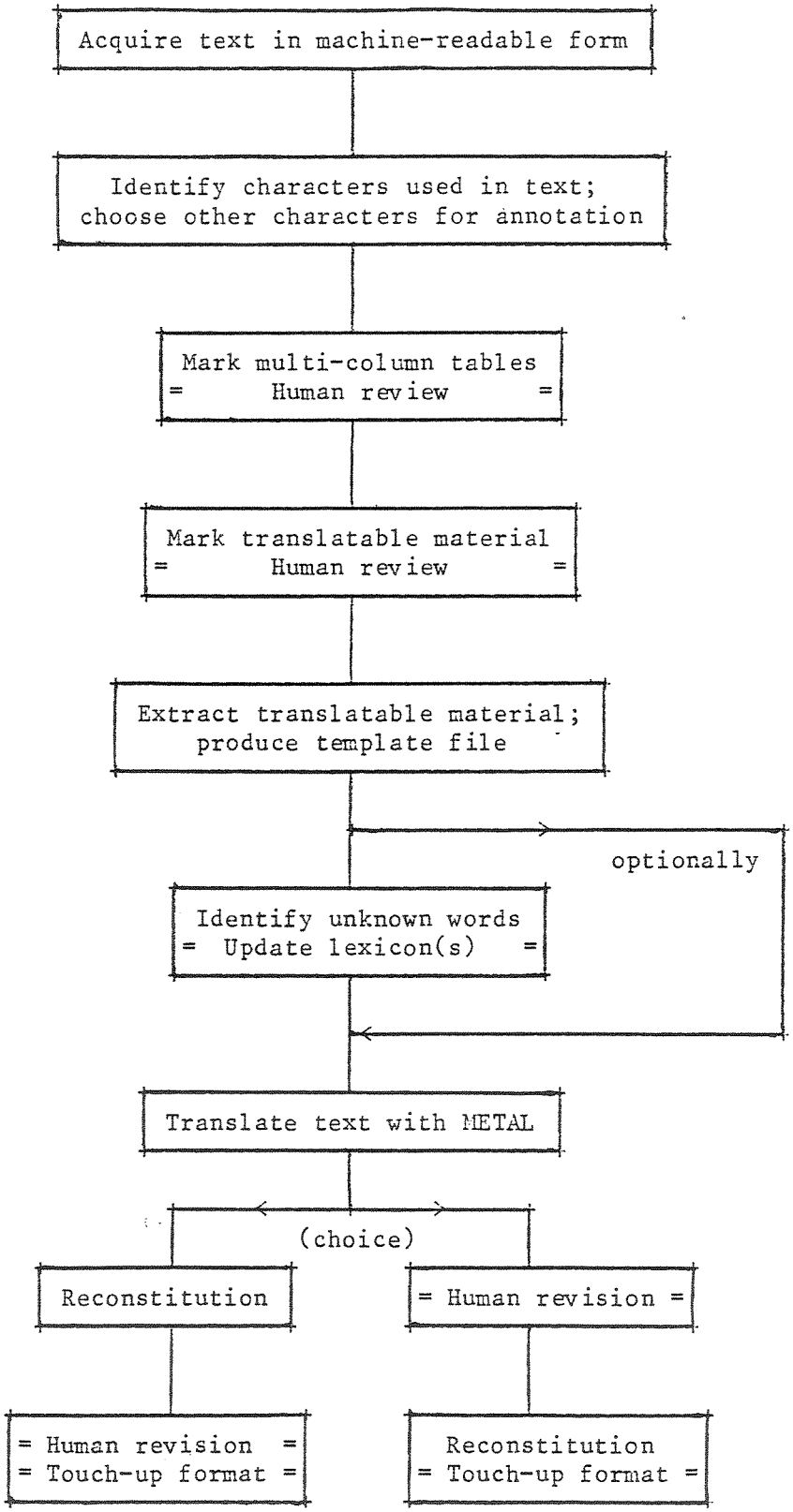


Figure 2
Steps in Translation

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 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; where METAL fails to achieve a concise phrasal analysis, a translation of the technical terms is provided, and the revisor must complete the translation. Figure 2 depicts the steps in "production" machine translation using the LRC MT system.

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, TRANSFER, and GENERATE. We briefly discuss the linguistic rules that describe how analysis, transfer, and synthesis are to take place, then proceed to illustrate the three-step process using an example sentence.

Lexical 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 an analysis of either entry; simply stated, each monolingual entry provides the system with the information necessary for parsing 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 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.

```
(Ausgabe              CAT (NST)
        ALO   (Ausgabe)
        PLC   (WI)
        TAG   (ALL)
        RC    (AGT)          (LOC)
        MC    (durch ueber) (an auf) $
        FC    (PP)
        CL    (P-6 S-A)
        GD    (F)
        CP    (LC)
        DR    (NP RD)
        SX    (N)
)
```

```
(output               CAT (NST)
        ALO   (output)
        PLC   (WI)
        TAG   (ALL)
        RC    (AGT)      (LOC)
        MC    (through) (on) $
        FC    (PP)
        CL    (P-02 S-01)
        CP    (LC)
        ON    (VO)
        DR    (NP RD)
        SX (N)
)
```

```
(output          (Ausgabe) NST (CAT NC))
```

Figure 3

Monolingual and Transfer Entries
for the German noun stem `Ausgabe´
and the English noun stem `output´

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. Three transfer entries for the German preposition `vor´, shown below, will illustrate:

```
(ago          (vor) PREP (CAT PREP) (TY DUR))
(before       (vor) PREP (CAT PREP) (TY PUN))
(in-front-of  (vor) PREP (CAT PREP) (TY * DUR PUN))
```

In these entries the English translation is defined by a restriction on semantic type (TY); i.e., the presence in context of an object noun of TYpe DURative results in the English translation `ago´ (which will later be postposed); if PUNctual, then `before´; otherwise, `in front of´ is chosen.

Grammar Rules

For human-engineering reasons, the most convenient form for expressing a grammar is via context-free phrase-structure rules. Context-free rules alone cannot fully describe a human language, but more general phrase-structure rules preclude efficient computational treatment. The traditional solution to this dilemma is to augment the context-free rules by associating with them procedures in some programming language in order to provide the necessary generative power. 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 syntax rule is not applied. 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 separate syntax 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 semantic correspondence among constituents; a phrase CONSTRuctor, which formulates the interpretation (phrase) defined by the current rule; and a TRANSFer part, which operates 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 frames 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 necessarily) 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.

The grammar for METAL consists of a number of unordered, augmented phrase-structure syntax rules plus transformations. The following relatively simple rule for building nouns will be used to illustrate the parts and format of METAL grammar rules:

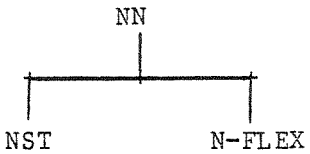
```
NN        NST          N-FLEX
0         1            2
--        (REQ WI)     (REQ WF)

TEST      (INT 1 CL 2 CL)

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

TRANSF    (XFR 1)
          (ADF 1 ON)
          (CPY 1 MC DR)
----------
```

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 the system 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:



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).

The final part of the grammar rule is the TRANSF section. (In a multi-lingual implementation, there would be a separate TRANSF section for each Target Language.) TRANSFer is invoked only after a sentence (S) has been built, at which point the system will perform the operations specified, generally moving down the tree to the terminal nodes where lexical substitution takes place. In our example rule, the first operation is

```
(XFR 1)
```

which causes the system to recursively invoke transfer on the first node (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.

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

Space does not allow a more detailed discussion of the grammar or lexicon (see Bennett, 1982). However, the short German sentence "Die Ausgaben werden auf Magnetband geschrieben" will be used to demonstrate the translation process in the following three sections on analysis, transfer, and synthesis. For the sake of brevity, only the successful rules for one interpretation will be outlined. (Using an all-paths parser, METAL actually tries all possible rules and may find a number of possible interpretations for a given sentence.)

Analysis

The PARSE module constructs a unified syntactic and semantic analysis of the "sentence" (actually, unit of translation), and its output serves as input to TRANSFER. The parsing program interprets the sentence according to the available linguistic analysis rules. These come in several varieties: dictionary rules, called lexical entries; phrase-structure syntax rules; case frames, which determine the relationships between the main verb and the nouns in its clause; and transformations. More detailed discussion of the parser (i.e., the LISP program that interprets the linguistic rules) appears in the next major section.

The trace below indicates the appropriate grammar rules for parsing the German noun phrase "die Ausgaben". The specific morphemes are given in square brackets; the numbers are solely for reference.

```
NN:1 -> NST[AUSGABE] N-FLEX[N]
NO:2 -> NN:1
NP:3 -> DET[DIE] NO:2
```

The auxiliary `werden´ is then lexically analyzed as such, and the prepositional phrase "auf Magnetband" is built-up using the sequence of rules indicated in the following trace:

```
NN:4 -> NST[MAGNETBAND]
NO:5 -> NN:4
NP:6 -> NO:5
PP:7 -> PREP[AUF] NP:6
```

Then the past participle `geschrieben´ is built-up to be a verb (VB); this VB, in turn, becomes a non-finite predicate (NFPRED) and, with "auf Magnetband", a non-finite clause (NFCL), using the rules indicated in the following trace:

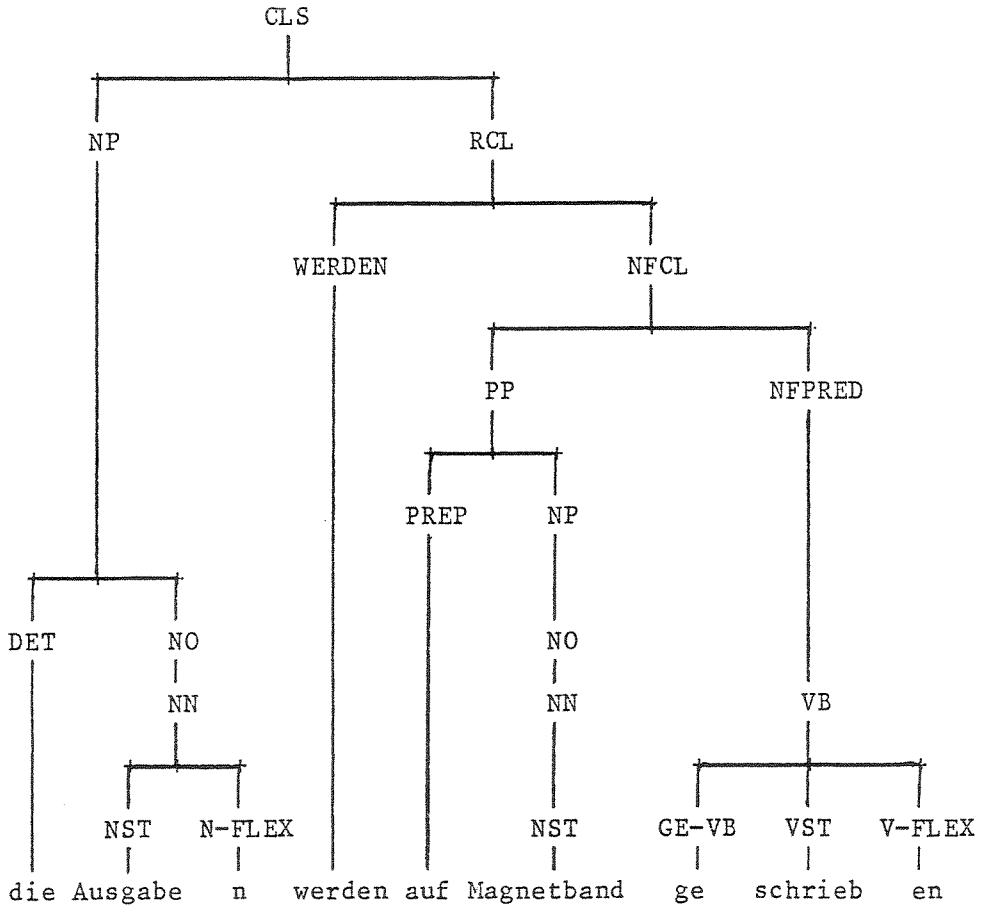
```
VB:8 -> GE-VB[GE] VST[SCHRIEB] V-FLEX[EN]
NFPRED:9 -> VB:8
NFCL:10 -> PP:7 NFPRED:9
```

The auxiliary `werden´ then combines with the non-finite clause (NFCL) to form a right-branching clause (RCL); a clause (CLS) is formed using the NP "die Ausgaben" and the RCL "werden auf Magnetband geschrieben". These rules are

indicated in the following trace:

RCL:11 -> WERDEN[WERDEN] NFCL:10
CLS:12 -> NP:3 RCL:11

The syntax tree built so far is represented below:



The TEST part of the clause rule (i.e., the rule CLS -> NP RCL) includes application of the case frame. The case frame will use the information coded on the lexical entry for the verb `schreiben´ to test for a valid clause (the presence and appropriateness of necessary arguments, etc.) and assign roles to the clause constituents. Only if the case frame succeeds can a clause (CLS) be built. In our example the CLS rule will succeed, and the clause may then be built-up to an S using the rule indicated in the following trace:

S:13 -> "$" CLS:12 "$"

(Dollar signs are inserted during the preprocessing of the text to mark S boundaries.) Once the S is built (actually, all Ss), analysis terminates.

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. 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 syntactic 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 a transformation, and/or lexically transfer a terminal son. Lexical transfer replaces a SL canonical form with a TL canonical form using the 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).

A case frame, when applied during the transfer phase, will order the case role-fillers as required by the TL 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 verb entry. In our example, the role-fillers for `write' are assigned to the Subject and Object positions.

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). The top-level node (phrase) in the newly constructed TL tree is returned by TRANSFER as its functional value.

Synthesis

The GENERATE function synthesizes the translation by simply taking the TL tree produced by TRANSFER and appending together all of the lexical allomorphs (words) 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: "The outputs are written on magnetic tape."

The 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 Cocke-Kasami-Younger bottom-up algorithm [Aho & Ullman, 1972], augmented with "top-down filtering" (similar to an "oracle" [Pratt, 1975]) for restricting the potential rules to at most those that a top-down parser would apply. This parser was shown to be highly efficient during an extensive series of experiments comparing a dozen parsers on the basis of their practical performance characteristics [Slocum, 1981]; the one parser determined to be slightly more efficient during those experiments (a left-corner parser augmented by top-down filtering, which closely resembles an Earley parser) will soon replace the current CKY 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. The parser adds the first word to the chart, draws all grammatical inferences from that addition, then repeats this process, adding one word at a time. 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 processed all the words in the input, the chart is examined for phrases which span that input, and whose syntactic categories appear in the user-definable list of acceptable ROOTCATEGORIES: usually (S). These phrases constitute the interpretations of the input sentence.

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.

Morphological analysis is a relatively simple process, relying on a letter-tree 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 is not a process distinct from other forms of disambiguation.

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 acceptable scores the resulting 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 earlier). 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. Provided the number of such phrases falls below a threshold, an S phrase is built just as if there were a grammar rule with the discovered phrases listed in its right-hand-side. But if the number of such phrases exceeds a threshold, a terminological analysis -- phrases consisting only of technical terms in the SL sentence -- is produced instead. A default rule-body procedure is attached, to be invoked during the transfer phase; this procedure will simply invoke TRANSFER on the constituent phrases of the dummy phrase just built. In order for phrasal/terminological analysis to be possible when the parser is unable to continue an analysis attempt (i.e., when it blocks in the middle of the sentence), the top-down filtering process must be turned off. (Since the filter restricts the rules considered to those that a top-down parser would consider, the same fate would otherwise befall METAL's parser as befalls any top-down parser when a block occurs: it would have to halt.) With the filter turned off, METAL's parser reverts to being a pure bottom-up parser -- i.e., it applies any rule which matches the current phrase sequence in the chart, regardless of whether it could ever contribute to a sentential analysis in the current context.

Discussion

From the Artificial Intelligence standpoint, there are several perhaps controversial facets of the METAL translator. A few of these can be discussed in the available space: the existence of a transfer phase; phrase-structure syntax rules; feature-based semantics; and a bottom-up, all-paths parser. Before proceeding to the arguments, we must state the goal of our project: to demonstrate the feasibility of an operational MT system involving joint application of linguistic theory and state-of-the-art capabilities in natural language processing. The key words here are `operational' and `application'. Our intent is to employ such methods as seem most likely to prove effective in immediate large-scale application. The research activities of the LRC MT project are to a large degree directed toward refining and extending techniques which have been significantly, yet perhaps inadequately, explored; indeed, part of our project's intent is to approach an adequate exploration of sometimes overlooked methods. We are not involved in extending the frontiers of science into what might be called "deep understanding" since there is no possibility of immediate application of such techniques. Nevertheless we do feel that we are contributing to the science of Computational Linguistics: we are experimentalists attempting to validate linguistic theories (our own as well as others') in a realistic environment.

The Existence of a Transfer Phase

It is argued that machine translation should be a two-stage affair -- that an MT system should analyze SL into a "universal" deep structure and then generate TL sentences. This may indeed be the ideal solution in the best of all possible worlds, but is not a viable approach at the current time. No adequate description of universal deep structure has been proposed, and it is clear that such a theory will not soon be forthcoming. More cogently, reasonable arguments have been advanced that such an approach may not be necessary or even desirable, at least for certain families of languages

[Boitet, 1980a]. If we can demonstrate the lack of necessity for such depth, even if for our limited purposes, then a significant contribution to understanding natural language will have been made.

The Use of Phrase-structure Rules

The use of phrase-structure rules -- or syntax rules of any variety -- is castigated by some who favor other approaches, particularly when it is felt that their inadequacy has been demonstrated. However, several reasonably successful large-scale natural language processing systems have used, and continue to use, phrase-structure rules [Hendrix et al., 1978; Robinson, 1980; Sager, 1981], and recent linguistic research [e.g., Bresnan, 1977; Gazdar, 1981] advances theoretical arguments in favor of phrase-structure grammars. In the laudable interest of achieving the "deeper understanding" claimed to be necessary for successful MT, certain AI proposals have eschewed breadth -- without demonstrating that breadth is achievable within rational space/time constraints via such methods [e.g., Carbonnel et al., 1978]. Again, if real progress can be made in MT with phrase-structure rules, it is worth knowing.

The Use of Feature Semantics

The use of feature semantics may be questioned. But large-scale application of such a theory is not difficult, and the prospect of constructing literally thousands of "frames," "scripts," or large, complex semantic models by any other name, is very unattractive to say the least. Since there exists no mature semantic theory, in any guise, suitable for inclusion within an applied natural language processing system, one must seek a formalism allowing easy large-scale application even though the theory be incomplete. This is true of a feature semantics theory, but the same cannot be said of a theory requiring complex models. More to the point, while it may be true that complex models are necessary for understanding stories, the structure of technical texts such as we are dealing with is in no way comparable to stories, and complex models do not seem to be required for cost-effective translation. (Admittedly, they may be required for "perfect" translation.) Technical texts in particular are not written in narrative style, nor do they appear to exhibit the regular, predictable patterns on which complex models are predicated. If it happens that such models are indeed required for cost-effective technical MT, one can rest assured that success is yet decades in the future: no one has advanced an adequate proposal for automatically identifying the models applicable to a given text segment, for one thing, and for another, the production of such models is a very long-term proposition. In contrast, we are encouraged by the relatively successful applications of feature semantics in reasonably large natural language processing systems.

The Use of a Bottom-up, All-paths Parser

Finally, there is the matter of using a bottom-up, all-paths parser. It is true that bottom-up parsers tend to apply many more rules than top-down parsers. But they also tend to be much faster at applicable rule discovery. Recent experiments [Slocum, 1981] have shown that bottom-up parsers can be more efficient than top-down parsers in actual practice. In addition, there is no need to proscribe or discourage the writing of left-recursive syntax rules when a bottom-up parser is employed. However, a critical factor for applications like MT is that, if the grammar does not account for a sentence, a top-down parser cannot ordinarily continue, and it will fail to produce any

analysis for at least part of the sentence (usually all of it). A bottom-up parser is not so restricted. Of course, given the same grammar and the same "ungrammatical" sentence, a bottom-up parser cannot achieve a unified analysis any more than a top-down parser can, but it is possible to continue the analysis attempt through the entire sentence, which a top-down parser does not normally admit. (It could be done, in theory, but not in any straightforward manner naturally suited to the standard control structure for a top-down parser. To the writer's knowledge this has never been attempted, probably because the performance penalty would be severe in any implementation.) Thus, with a bottom-up parser and a suitable chart searching program, an analysis of the sentence in terms of its phrases can easily be found and employed for translation -- a decided advantage over failing to produce any translation.

Our phrasal analysis mechanism, which essentially requires a bottom-up parser, has several advantages besides allowing some translation to be synthesized when no analysis of the input sentence is possible with the available rules. First, it allows successful analysis of a sentence even if it contains a parenthetical expression which itself cannot be analyzed (except via phrases); thus a failure to fully analyze a parenthetical expression need not result in a failure to analyze the sentence containing it. Second, phrasal analysis has proven beneficial for debugging purposes: the resulting phrases provide a clue to the rule or rules which failed to operate correctly, or which were missing entirely. Finally, it allows us to explore the possibility of varying levels of translation quality with corresponding levels of expense. For example, by discarding or rendering inoperative certain rules, METAL might operate much more rapidly because of the decrease in ambiguity, with attendant loss in translation quality. (This has not yet been tried.)

We base our use of an all-paths technique (which is not necessarily characteristic of a bottom-up parser) on two observations: no one knows how to write a best-path parser; and a first-path parser has a characteristic problem that an all-paths parser can overcome. The first point is self-evident, so we will proceed to the second. A first-path parser depends on a static ordering of the syntax rules; if there is any ambiguity in the grammar at all, it will be the case that some rules might never be "seen" by the parser -- even though they may be required for (the correct reading of) a given sentence. An all-paths parser will miss no readings because it produces all of them. This permits a comparison of the alternatives, and an intelligent choice among them to be made. The METAL parsers are all-path parsers, and include a "scoring" feature whereby different readings of a morpheme, word, phrase, and finally a sentence, may be assigned different degrees of plausibility, allowing translation of the most likely -- rather than the only -- interpretation. The only remaining argument against the all-paths technique is that it is presumably too expensive computationally to be cost-effective. We regard this position as a matter for empirical validation; our data to date indicate that, contrary to popular opinion, an all-paths parser is not necessarily expensive.

Recent Experimental Results

In the last two years, METAL has been applied to the translation into English of over 43,000 words of German telecommunications text. 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

Since 1980, METAL has been implemented in INTERLISP and run on a DEC 2060. Due to the 18-bit address space limitation imposed by this combination of software and hardware, METAL has been split into two processes (one analysis, one transfer and synthesis) which communicate via a shared text file. In addition, special programs have been written to reduce the amount of address space devoted to data objects; these programs degrade runtime performance. Nevertheless, recent measurements have shown that METAL's translation speed in this environment is around two CPU seconds per word of SL (German) text. This figure encompasses all forms of overhead -- swapping, paging, text I/O (including inter-process communication), and storage management -- in addition to actual translation time. In particular, storage management (called "garbage collection" in LISP) accounts for approximately 45% of this CPU time, due to the fact that usable working space is scarce despite all our efforts to increase it via software techniques. In an environment with a larger address space this 45% figure should decrease drastically, resulting in a significant performance increase -- all other things being equal.

To eliminate the address-space barrier and its associated R&D problems, one of the project sponsors has recently provided the LRC with a Lisp Machine (a Symbolics LM-2) having a 24-bit address space. The LM-2 is much less expensive than a DEC 2060. But even though it is significantly slower in terms of raw CPU power, we expect it to closely approximate the 2060's real-time performance due to the elimination of the data compaction/expansion overhead, the virtual elimination of storage management (garbage collection) overhead, and the lack of resource competition with other users. METAL's throughput, then, should not suffer significantly. It may even improve.

The single quantitative assessment of greatest interest in the MT community is almost certainly cost-effectiveness. Until this figure is established by unbiased third parties, taking into account the full costs of translation and revision using METAL vs. conventional (human) techniques, this question cannot be adequately addressed. (This type of assessment should become available to us later this year.) However, we have worked out some minimum performance parameters that are of interest. It has been calculated that METAL should prove cost-effective if it can be implemented on a Lisp Machine supporting 4-6 post-editors who can sustain an average total output of about 60 pages/day. At 250 words/page, and 8 hours/day, this works out to 1.92 seconds/word, minimum realtime machine performance. If the upcoming 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 5.76 seconds/word, minimum realtime performance -- about what we now experience on a very lightly-loaded DEC 2060. If this goal can be reached while maintaining a high enough standard of quality that an individual revisor can handle 10-15 pages/day, METAL will have achieved cost-effectiveness.

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 one 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 may or may not include the full matrix sentence, depending on its length. 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. (The text is never retranslated in its entirety during this phase.) A few months 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 in any way whatever, with respect to the original machine translation, after revision is complete." So far, revision has been performed by members of our project staff.) More importantly, we measure revision performance: the amount of time required to edit the text.

In the course of experimenting with four different texts comprising some 43000 words, our "correctness" figures have varied from 55% to 85% (of full-sentence units) depending on the individual text and whether the experiment was of the "blind" or "follow-up" variety. In our only serious measurement of revision performance to date, taken after a "follow-up" translation of 50 pages of text, revision required 15 hours using a primitive on-line editing program -- 3.33 pages/hour, or 26.7 pages per day. Since the revision was performed by a project staff member, it cannot serve as the basis of an unbiased projection; nevertheless, we take it to indicate that METAL's translation quality at least approximates the minimum required for cost-effectiveness.

Conclusions

MT Research at the LRC 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 answer questions about their extensibility and limits of 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; 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 a more extensive use of semantics. By early fall, 1982, a semantic matrix and a more intelligent way of handling adverbials will have been implemented.

Of special scientific interest is this: the degree to which "shallow" techniques can profitably be applied to certain areas of language processing argues against the necessity of "deeper" (and much more expensive) AI methods for those application areas. Progress in Machine Translation at the LRC [Slocum, 1980] and elsewhere [Boitet, 1980b], generally using the more traditional NLP techniques, argues for a reconsideration of positions denying their adequacy and utility; indeed, it establishes the successful methods as exemplifying the state of the art in Computational Linguistics.

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.

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

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

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.

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," Tech. Note 205, SRI International, Menlo Park, Calif., Feb. 1980.

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.

Further Reading

Bruderer, H. E. Handbuch der maschinellen und maschinenunterstuetzten Sprachuebersetzung: automatische Uebersetzung natuerlicher Sprachen und mehrsprachige Terminologiedatenbanken. Vlg. Dokumentation, Munich, 1978.

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

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.

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.