The LRC Machine Translation System

Winfield S. Bennett[1]
Siemens Communication Systems, Inc.

Jonathan Slocum
Microelectronics and Computer
Technology Corporation (MCC)

The Linguistics Research Center (LRC) of the University of Texas at Austin is currently developing METAL, a fully-automatic high quality Machine Translation (MT) system. After outlining the history and status of the project, this paper discusses the system's projected application environment and briefly describes our general translation approach. After detailing the salient linguistic and computational techniques on which METAL is based, we consider some of the practical aspects of such an application, including experimental results that imply the system is now ready for production use. Two exhibits are appended: a German original text and its raw METAL translation. (This is not the best translation ever produced by METAL, but it is better than average.) We close by indicating some future directions for the project.

1 History and Status

Machine translation research at the LRC began with the founding of the center in 1961. For much of the history of this research, funding came from the U.S. Air Force's Rome Air Development Center and other U.S. government agencies. In January 1979, Siemens AG began funding the current development phase of the METAL system; the project then comprised one full-time and five or six half-time workers. As a result of Siemens's support, the existing system was scrapped and a new implementation effort was undertaken in the spring of that year; the project staff grew slowly at first, but with a recent substantial increase now numbers seven full-time and five half-time workers. The first operational version of the system was delivered to the sponsor for market testing in January 1985.

The current system translates only from German into English, although work to add other Target Languages (Spanish and Chinese), as well as a second Source Language (English), is underway. The German grammar in its present form contains more than 600 rules; the lexicon has well in excess of 20,000 monolingual entries for German and English, and is expected to double in size in the near future.

2 Application Environment

Like any modern MT system, METAL is to be used in a technical translation environment where human revision of the output is expected – just as is the case with human translation around the world. The justification for using an MT system is a combination of pure economics (cost reduction) and necessity (to achieve the desired speed, perhaps required to get the job done at all); the trade-off between these two differs, depending on the organization and circumstance. In general, we expect that the LRC MT system must prove cost-effective in order to gain user acceptance: speed per se will be a secondary consideration. This means that the cost of using METAL for draft translation, and a human revisor thereafter, must be significantly [not marginally] less than the cost of using humans for draft translation, and a human revisor thereafter; the cost of "using" METAL must include its full amortization, etc.

In an environment of technical translation, particularly of operation and maintenance manuals, one of the big problems is text format. As a glance at any technical manual will show, it is not the case that all material in a document must or can be translated. Large portions of a

[1] Current address: Linguistics Research Center; P.O. Box 7247; University Station; Austin, TX 78712

0362-613X/85/020111-121$03.00

formatted text (up to 50% of the characters, in our experience) may not be translatable; 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., flowchart box boundaries, sentence-internal formulas and acronyms, and various attention-focusing devices). The translatable units (not always "complete sentences") must then be extracted, translated, and finally reconstituted so as to appear like the original text.

In the LRC MT system, special programs have been developed to handle the formatting problems associated with technical manuals. This software automatically marks texts and extracts translatable units for input to METAL, and reformats the translation afterwards (Slocum and Bennett 1982; Slocum et al. 1984). The only human intervention expected prior to translation is checking and correcting the output of the routines that mark translatable units; the human does not, for example, resolve anaphora or disambiguate homographs or word senses. Nothing in the LRC MT system provides for human intervention during the actual translation phase.

Text processing is presently done on a DEC-2060, post-editing on a PDP-11, and the actual translation on Symbolics Lisp Machines. For the foreseeable future a Lisp Machine will continue to be used as the "translation engine." The project sponsor is constructing a microcomputer front end supporting four to six translator workstations (as well as OCR devices, floppy-disk drives, printers, etc.), on which texts will be prepared and sent to the batch translation unit, and on which the output will be reformatted and revised; software specially suited to text preparation and post-editing is being programmed. Thus, the production version will constitute a complete translation environment.

3 General Translation Approach

In METAL, translation proper consists of four successive phases: Analysis (parsing), Integration, Transfer, and Synthesis (generation). The Integration phase works with analysis tree structures, performing (at the present time) inter- and extra-sentential anaphora resolution. Until recently, the Transfer and Synthesis phases were essentially a single phase, but work is in progress to split this phase, and introduce a much more powerful Synthesis phase. In this section we describe "Transfer" and defend it as our general translation approach; in the next section we discuss our linguistic techniques more fully.

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. Carbonell 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") approach by the CETA group at Grenoble (Boitet and Nedobejkine 1980). After an enormous investment in time and energy, including experiments with massive amounts (400,000 words) of text, it was decided that the development of a suitable pivot language (for use in Russian-French translation) was not yet possible. 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 seems necessary in a practical setting. This being the case, it is 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.

The standard alternative, which we have adopted, is to have a **transfer component** that maps "shallow analyses of sentences" in the SL into "shallow analyses of equivalent sentences" in the TL, from which synthesis then takes place. This assumes the form of a transfer dictionary and a transfer grammar. 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, we believe, proceed to construct usable systems amenable to progressive enhancement as linguistic theory becomes able to support deeper models.

4 Linguistic Techniques

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 discuss our use of the following linguistic techniques:

- allomorphic lexical analysis;
- a phrase-structure grammar;
- syntactic features;
- semantic features;
- scored interpretations;
- transformations indexed to specific rules; and
- attached procedures to effect translation.

4.1 Allomorphic Lexical Analysis

Entries in METAL monolingual dictionaries are indexed by both **canonical form** (the usual spelling one finds in a printed dictionary) and **allomorph** (the stem, without productive affixes). The affixes themselves are separate dictionary entries; although their semantics is necessarily different in kind from content morphemes, they are treated identically by the system software. If a particular stem exhibits internal inflection (e.g., German nouns that umlaut in the plural), or varies for other reasons, then multiple entries are stored, one for each stem variation [allomorph]. At first this may seem wasteful, but the majority of such cases in our dictionaries are German strong verbs – which sometimes behave differently, depending on inflection, and thus would require separate entries anyway.

At system-generation time, the allomorphs are entered into a **letter tree,** which is searched during lexical analysis. The analysis of a word occurrence, then, is normally one or more sequences of morphemes (stems and affixes, mixed), each morpheme being an allomorph corresponding to one or more dictionary entries. These are fed to the parser as if they had been separate [alternative sequences of] "words" in the text (except that each morpheme is marked according to whether it was word-initial and/or word-final), which parses them [back] into words while it is parsing the words into a sentence. Lexical ambiguity (including homography and polysemy, as well as ambiguity in morphological decomposition) is tolerated as a natural phenomenon in the system, and is resolved according to a scoring scheme, discussed below, which handles syntactic ambiguity as well.

4.2 PHRASE STRUCTURE GRAMMAR

In the LRC MT system we employ a phrase-structure grammar, augmented by strong lexical controls and extensive use of transformations. The LRC MT system is currently equipped with over 600 PS rules describing the best-developed Source Language (German), and over 10,000 lexical entries in each of the two main languages (German and English). The current state of our coverage of German 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 a few 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 – and certainly not for sentence forms never 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 linguistic techniques (which we doubt).

4.3 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 operators 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.

4.4 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: one set of operation and maintenance manuals we have worked with (describing a digital telephone switching system) is part of a document collection that is expected to comprise some 100,000 pages of text when complete. A typical NLP research group 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, in order to understand the text.) 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 during Transfer 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 have been surprised at the results achieved using only syntactic features during analysis.

4.5 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 preferencing and grammatical phenomena – plus the types of any spelling/typographical errors, which can sometimes be "corrected" in more than one way.

Scoring begins at the lowest level of the tree – at the morpheme level, based on lexical preference coded for dictionary entries (one per allomorph) and any spelling correction factors – and propagates upwards as the analysis proceeds. Homography and polysemy are dealt with as a natural consequence of the selection, from among all alternatives, of the most plausible [highest scoring] reading(s). Thus, nowhere in the system is there special provision for dealing with these problems: all sources of ambiguity are handled by the identical mechanism.

Our experiences relating to the reliability and stability of heuristics based on this technique are decidedly positive: we employ only the highest-scoring reading for translation (the others are 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.) This does require some careful "tuning" by the linguists, but this has been a manageable problem.

4.6 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, or even lexical entries, 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 or words 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. A transformation tied to a particular syntactic rule may be written as part of that rule, or called by name if the linguist wishes several rules to share the same transformation (e.g., a 12-21 constituent reversal is common).

4.7 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 matching structural transfer rules is eliminated), it prevents the "wrong" procedure from being 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 translation operations.

5 COMPUTATIONAL TECHNIQUES

Again, our separation of "linguistic" from "computational" techniques is somewhat artificial, but nevertheless useful. In this section we discuss our use of the following computational techniques:

- a "some-paths", parallel, bottom-up parser;
- associated rule-body procedures;
- spelling correction;
- another fail-soft analysis technique; and
- recursive parsing of parenthetical expressions.

5.1 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;
- 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
- last, 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 that 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.

5.2 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 et al. 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.

5.3 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 regarding 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 substitutions, 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 that have converted one word into another (valid in isolation) should not be tried.

5.4 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. Weischedel 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 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 & Heidorn 1982).

5.5 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, which, of course, actually performs grammatical analysis. Besides sheer elegance, this has the added 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.

6 Practical Considerations

From a user's viewpoint, there are four aspects on which MT systems should be judged: text preparation, dictionary update, actual translation, and post-editing. Other than dictionary update, these aspects are discussed elsewhere in this paper. We will therefore comment on our lexical maintenance procedures, and the users' acceptance thereof, before proceeding to our experimental results.

6.1 LEXICAL MAINTENANCE

The factors important to the terminologists who maintain lexical data bases include the kinds of information that one is required to supply, and the method used to enter that information in the dictionary (Whiffin, in press). If the lexical coding process is too complex, semantic errors will multiply and translation quality will suffer. Menu schemes wherein one selects the proper value of a feature from a short (10 item) list of options are greatly preferred over long lists of options or, worse, a scheme wherein one must volunteer the information via type-in. Even better is a scheme wherein the system, with minimal clues (e.g., root form and part of speech) generates an entry likely to be mostly correct, which the terminologist then verifies and edits (again, via menu selection) as necessary. Needless to say, arcane codes that one must have a manual to keep track of are to be avoided at all cost.

The lexicon for METAL is stored in an on-line DBMS written in LISP. Input of lexical entries is facilitated by an INTERCODER, a menu-driven interface that asks the user for information in English and converts the answers into the internal form used by the system. An integral part of the INTERCODER is the "lexical default" program that accepts minimal information about the particular entry (root form and lexical category) and heuristically encodes most of the remaining necessary features and values. Entries may also be created using any text editor, without the aid of the INTERCODER or lexical defaulter.

Interfacing with the lexical data base is done by means of a number of functions that permit the user to access, edit, copy, and/or delete entries individually, in small groups (using specific features), by entire categories, or in toto (essentially no longer done for reasons of size). In order to assure a high degree of lexicon integrity the METAL system includes validation programs that identify errors in format and/or syntax. The validation process is automatically used to check lexical items that have been edited or added, to ensure that no errors have been introduced.

Our terminologists indicate substantial subjective satisfaction with METAL's lexical coding scheme. Performance measurements indicate that, for categories other than verbs, a very few (2-5) minutes is all that is required to enter a pair of terms (in two languages, thus three dictionaries including Transfer); for verbs, the process is more complex, requiring as much as 20 minutes per word pair. But these times include the terminology research per se – i.e., the process of discovering or generating a proper translation of a term – so the overall burden of lexical maintenance seems quite acceptable, and cost-effectiveness is not adversely affected.

6.2 EXPERIMENTAL RESULTS

In the last five 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. (It is not obvious that this would be relevant, or of significant benefit.) However, some stimulating quantitative and qualitative statistics have been gathered.

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 post-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. Since the errors committed by an MT system seldom resemble errors made by human translators, the possibility of a "Turing test" for an MT system does not exist at the current time.

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 who then write ("code") entries for any novel words discovered in the list. 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.

6.3 TRANSLATION SPEED

On our Symbolics 3600 LISP Machine, with 512K 36-bit words of physical memory, preliminary measurements indicate an average performance of about 2+ seconds (real time) per input word; this is already 10 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; for other (predictable) reasons, as well, further speed increases are anticipated.

6.4 CORRECTNESS

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 five 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. During a recent "blind" test, for example, METAL achieved a 75% "correctness" figure on a moderately long text (ca. 10 pages).

6.5 INTERPRETATION OF THE RESULTS

Certain objections have been raised concerning the present feasibility of MT. It has been argued that, unless an MT system constitutes an almost perfect translator, it will be useless in any practical setting (Kay 1980). As we interpret it, the argument proceeds something like this:

(1) there are classical problems in Computational Linguistics that remain unsolved to this day (e.g., anaphora, quantifiers, conjunctions);
(2) these problems will, in any practical setting, compound on one another so as to result in a very low probability that any given sentence will be correctly translated;
(3) it is not in principle possible for a system suffering from malady (1) above to reliably identify and mark its probable errors;
(4) if the human post-editor must check every sentence to determine whether it has been correctly translated, then the translation is useless.

We accept claims (1) and (3) without question. We consider claim (2) to be a matter for empirical validation – surely not a very controversial contention. As it happens, a substantial body of empirical evidence gathered at the LRC to date refutes such claims: the "correctness" figures reported above (measured by our sponsor's post-editors) establish this contention. [In point of fact, we consider "correctness" figures to be virtually meaningless, aside from being unreliable, as will become obvious. But Kay's claim (2) assumes that "correctness" is a valid measure, and thus falls in either case.]

Regarding (4), we embrace the assumption that a human post-editor will have to check the entire translation, sentence-by-sentence; but we argue that Kay's conclusion ("then the translation is useless") is again properly a matter for empirical validation. Meanwhile, we are operating under the assumption that this conclusion is patently false – after all, where translation is taken seriously, human translations are routinely edited via exhaustive review, but no one claims that they are therefore useless! In other words, Kay's claim (4) also falls, based on empirical evidence that relates to HT directly – but, by extension, to MT as well.

6.6 ACCEPTANCE AND COST-EFFECTIVENESS

There is a meaningful, more-or-less objective metric by which any MT system can and should be judged: overall (man/machine) translation performance. The idea is simple. The MT system must achieve two simultaneous goals: first, the system's output must be acceptable to the post-editor for the purpose of revision; second, the cost of the total effort (including amortization and maintenance of the hardware, the software, and the dictionaries) must be less than the current alternative for like material – human translation followed by post-editing.

Regarding user acceptance, we can relate that the editors revising METAL translations in a series of experiments spanning the past few years have recently stated that the system has achieved a level of quality that they find acceptable for their day-to-day work (Whiffin, personal communication). From our experimental evidence, it would seem that this is a harder goal to reach than mere cost-effectiveness; i.e., "cost-effectiveness" can be demonstrated in experimental settings, with an output quality that the editors will not accept on a daily basis. In addition, translators have noted that the time required to create a triple of METAL dictionary entries – monolingual German term, monolingual English equivalent, and bilingual Transfer pair – varies from two to twenty minutes, depending on the part of speech and the amount of terminology research [needed for human translation in any case] required. On an on-going average basis, a new term can be expected once per page of text.

Until METAL is evaluated by unbiased third parties, taking into account the full costs of translation and revision using METAL versus conventional (human) techniques, the question of METAL's cost-effectiveness cannot be answered definitively. 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 system supporting four to six post-editors who can sustain an average total output of about 60 revised pages per day. At 275 words per page, and eight hours per day, this works out to 1.7 seconds per word, minimum real-time machine performance. Our mid-84 real-time performance figure of 2+ seconds per word on a Symbolics 3600 approaches this goal; it also compares very favorably with the human translation rate (experienced at Siemens, for decades) of four to eight pages per day for like material. If this level of performance can be slightly increased while maintaining a high enough standard of quality so an individual revisor can indeed edit 10 to 15 pages per day, on a daily basis, METAL will have achieved cost-effectiveness.

Most important, 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 to 17 pages per day (depending on the particular editor). In a second experiment, conducted in mid-83, 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 editors reported further improvement in their subjective impression of the quality of the output, and their revision rates were much higher: almost 30 pages per day. In a fourth experiment, conducted in mid 1984, their average revision rate climbed to over 40 pages per day; this figure also compares favorably with the revision rate of human translations experienced at Siemens: eight to twelve pages per day for like material (not including original translation time: four to six pages per day). These MT revision figures are surely biased by virtue of the experimental setting itself (i.e., one-shot measures of post-editing performance on human translations would be significantly higher than the on-going eight to twelve average quoted above), but nevertheless these numbers indicate that we have probably reached the goal of cost-effectiveness.

7 FUTURE DIRECTIONS

The METAL German-English configuration was ready for market testing in January 1985. Current plans are to continue improving the present system and to branch off into other target languages, specifically Spanish and Chinese. If our estimates are correct, a German-Spanish system should be ready for testing sometime in 1986, and a German-Chinese system sometime thereafter. There are also plans to begin serious work on an English-German system during 1985. If the planned work is successful, we will initiate work on English-Spanish and English-Chinese MT systems.

We anticipate retaining most of the system as-is, aside from the usual sorts of maintenance modifications; however, as mentioned above, we are in the process of upgrading the power of the Synthesis component. In addition, there are plans to change the format and content of Transfer lexical entries so as to standardize their format (verbs are structured differently from other parts of speech) and increase their ability to control structural transfer. These changes, we anticipate, will allow further improvement in the quality of the raw output of the LRC MT system, and so further enhance its attraction and cost-effectiveness.

8 References

Boitet, Ch. and Nedobejkine, N. 1980 Russian-French at GETA: Outline of the Method and Detailed Example. Proceedings of the Eighth International Conference on Computational Linguistics, Tokyo.

Carbonell, J.; Cullingford, R.E.; and Gershman, A.V. 1978 Knowledge-Based Machine Translation. Research Report #146 (December). Department of Computer Science, Yale University.

Hayes, P.J. and Mouradian, G.V. 1981 Flexible Parsing. *American Journal of Computational Linguistics* 7(4): 232-242.

Hendrix, G.G.; Sacerdoti, E.D.; Sagalowicz, D.; and Slocum, J. 1978 Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems* 3(2): 105-147. Reprinted in Larson, J.A. and Freeman, H.A., Eds., *Tutorial: Data Base Management in the 1980's*. IEEE Computer Society Press, Los Alamitos, California: 89-131.

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

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

King, M. 1981 Design Characteristics of a Machine Translation System. Proceedings of the Seventh International Joint Conference on Artificial Intelligence [7th IJCAI]. Vancouver, B.C., Canada: Vol. 1, 43-46.

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

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

Paxton, W.H. 1977 A Framework for Speech Understanding. Ph.D. dissertation available as Technical Note 142. Artificial Intelligence Center, SRI International, Menlo Park, California.

Pratt, V.W. 1973 A Linguistics Oriented Programming Language. Proceedings of the Third International Joint Conference on Artificial Intelligence [3rd IJCAI]. Stanford University, California: 372-381.

Robinson, J.J. 1982 DIAGRAM: A Grammar for Dialogues. *Communications of the ACM* 25(1): 27-47.

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

Slocum, J. and Bennett, W.S. 1982 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.

Slocum, J. et al. 1984 METAL: The LRC Machine Translation System. Presented at the ISSCO Tutorial on Machine Translation, Lugano, Switzerland (2-6 April). Also available as Working Paper LRC-84-2 (April). Linguistics Research Center, University of Texas.

Weischedel, R.M. and Black, J.E. 1980 If the Parser Fails. Proceedings of the 18th Annual Meeting of the ACL, University of Pennsylvania.

Whiffin, L. 1985 Machine Assisted Translation Systems: User Acceptability. Submitted to the Second Conference of the European Chapter of the ACL (28-29 March 1985).

Exhibit A: A German DP Text

EINTEILUNG DES PLATTENSPEICHERS

BLOCKSTRUKTUR

Die kleinste adressierbare Informationseinheit ist ein Block = 1 Sektor. Zu jedem Block gehoert ein Header. Der Header enthaelt die gesamte Adresse, sowie Angaben ueber den Zustand des Blockes (Benutzbarkeit!). Zur Sicherung der Header-Information und der Daten befindet sich am Ende des Headers und des Datenfeldes ein Pruefzeichen von 16 Bit.

Vor dem Headerfeld befindet sich eine Praeambel von 42 Byte Laenge fuer den Ausgleich aller Toleranzen.

Vor dem Datenfeld befindet sich eine Praeambel von 5 Byte Laenge zur Aufsynchronisierung der Leseverstaerker. Vor und hinter dem Datenfeld befindet sich eine Luecke. Die Luecken sind aus folgenden Gruenden notwendig:

Luecke 1: 56 Bit wegen Schreib-Loesch-Kopfabstand.Zu Beginn der Daten-Schreiboperation muss gewaehrleistet sein, dass der Loeschkopf den Header nicht zerstoeren kann.

Luecke 2: 316 Bit im Normalmodus wegen der Toleranzen in der Umdrehungsgeschwindigkeit. Es muss die Moeglichkeit beruecksichtigt werden, dass das Schreiben des Blockes (Header + Datenfeld) an der unteren und oberen Grenze der Umdrehungsgeschwindigkeit erfolgen kann.
Im Spezialmodus wird diese Luecke wegen der kleineren Bloecke 1340 Bit lang.

Am Ende des Header- und Datenfeldes befindet sich 1 Postambel von 8 Bit Laenge.

SPURSTRUKTUR

Eine Spur wird eingeteilt in 4 bzw. 8 Sektoren. Die Unterteilung der Spur in Sektoren erfolgt durch Index- und Sektormarken.

Die Indexmarke wird magnetisch durch einen Schlitz auf der untersten Platte des Plattenstapels erkannt und dient als allgemeiner Bezugspunkt fuer den Aufbau der Spurstruktur. Vom Indexpunkt ausgehend wird die Spur mit einem eigens dafuer vorgesehenen Dienstprogramm (oder Simulator!) mit Headern beschrieben. Die Bitzahl fuer das Datenfeld wird so bemessen, dass auch bei unguenstiger Drehzahl (= 2448 U/min) immer noch 4 bzw. 8 vollstaendige Bloecke Platz finden. (Siehe Abschnitt 4.1 Luecke 2). Je nachdem bei welcher Geschwindigkeit die Spur beschrieben wird, entsteht zwischen Ende des Datenfeldes und Indexmarke bzw. Sektormarke eine mehr oder weniger grosse Luecke.

Exhibit B: A Raw METAL Translation

DIVISION OF DISK STORAGE

BLOCK STRUCTURE

The smallest addressable information unit is a block = 1 sector. A header is part of every block. The header includes the entire address, sowie specifications about the state of the block (usability!). A check character of 16 bits is found for the saving of the header information and the data at the end of the header and the data field.

A preamble of 42 byte length for the adjustment of all tolerances is found in front of the header field.

A preamble of 5 byte length is found in front of the data field for the synchronization of the read amplifier. A gap is found in front of and behind the data field. The gaps are necessary from the following reasons:

Gap 1: 56 Bit because of distance between write and erase heads. At the beginning of the data write operation it must be guaranteed, that the erase head can not destroy the header.

Gap 2: 316 Bit in the normal mode because of the tolerances in the rotational speed. The possibility must be considered that writing the block (header + data field) at the lower and upper limit/boundary of the rotational speed can occur.
This gap becomes 1340 bits long in special mode because of the smaller blocks.

The 1 postamble of 8 bit length is found at the end of the header and data field.

TRACK STRUCTURE

A track is divided into 4 and/or 8 sectors. The subdivision of the track into sectors occurs through index label and sector marks.

The index label is recognized magnetically by a slot on the lowest disk of the disk pack and is used for the track structure as the general reference point for establishing. By the index point, the track with a utility program designated especially for that (or simulator!) is described with headers. The number of bits for the data field is calculated then that always still 4 and/or 8 complete blocks do also find space with unfavorable rotational speed/number of revolutions (= 2448 r.p.m.s). (See section 4.1 gaps 2.) Depending on with which speed the track is described, a more or less large gap occurs between the end of the data field and index label and/or sector mark.

SEKTORMARKIERUNG

Die Sektormarke wird ebenso wie die Indexmarke von der Schlitzplatte, die sich als Bodenplatte an jedem Plattenstapel befindet, magnetisch abgenommen. Im Handel werden Plattenstapel mit 32 und mit 20 Schlitzen angeboten. Im vorliegenden Fall soll der Plattenstapel mit 20 Schlitzen beim WSP 411 und mit 32 Schlitzen beim WSP 414 verwendet werden. Eine Maske, dargestellt durch einen Zaehler blendet aus den 20 bzw. 32 Sektormarken 4 bzw. 8 aus, so dass 4 bzw. 8 gleich grosse Sektoren entstehen. Die Maske bzw. der Zaehler wird von der Herstellerfirma (CDS) in jeden Wechselplattenspeicher fest eingebaut.

SEKTORMARKIERUNG

Dargestellt ist die Struktur des WSP 411.

Ein Plattenstapel umfasst 6 bzw. 11 Platten mit 10 bzw. 20 benutzbaren Oberflaechen. Pro Oberflaeche befindet sich ein Kopf. Jeder Kopf ueberstreicht 203 Spuren. Die Gesamtheit aller Spuren mit gleichem Radius nennt man Zylinder.

Der Plattenstapel umfasst also 203 Zylinder. Die Zylinder haben die Adressen 000-202, die Koepfe haben entsprechend den Plattenseiten die Adressen 0-9 bzw. 0-19.

Bei einer fortlaufenden Uebertragung wird die Adresse in der Reihenfolge Sektor, Kopf, Zylinder erhoeht.

SECTOR MARKER

Likewise the index label is read in the sector mark as by the slot disk which is found as a bottom disk at every disk pack magnetically. The disk packs with 32 and with 20 slotting are offered on the market. The disk pack should be used in this case with the 20 slots with the WSP 411 and with the 32 slots with the WSP 414.

A mask, represented through a counter masks out 4 and/or 8 from the 20 and/or 32 sector marks so that large sectors result similar to 4 and/or 8. The mask and/or the counter is incorporated by the Herstellerfirma (CDS) into every removable disk storage.

STORAGE STRUCTURE

The structure of the WSP 411 is represented.

A disk pack contains the 6 and/or 11 disks with 10 and/or 20 usable surfaces. A heading is found per surface. Every heading covers 203 tracks. The entirety of all tracks with same radius calls one cylinder.

Therefore the disk pack contains 203 cylinders. The cylinders have the addresses 000-202, the headings have the addresses 0-9 and/or 0-19 corresponding to the disk surfaces.

Sector, heading, cylinder is increased the address during a continuous transfer in the sequence.