[*MT evaluation: basis for future directions*. Proceedings of a workshop...2-3 November 1992, San Diego]

INTERNAL EVALUATION: Practical Use and Effect of the JICST Japanese-English Machine Translation System

Koji Tamura and Isao Tominaga
Japanese Information Center of Science and Technology (JICST)

The JICST Japanese-English machine translation system, MAJESTIC (Machine-Aided JICST's Exclusive System for Translating Information Comprehensively), was developed from April 1986 to March 1990 and has been in practical use since August 1990 for preparing the JICST-E File, an English version database of Japanese originated bibliographic information covering science, technology and medicine. MAJESTIC was developed by making effective use of the results from the national project "Research on Fast Information Services between Japanese and English for Scientific and Engineering Literature" (Mu Project), which was supported by the Special Coordination Fund of Science and Technology Agency (STA). In this paper the evaluation of MAJESTIC is described in comparison with a manual translation from the viewpoints of cost-effectiveness, the time-saving effect, the possibility of improving productivity, and employment chances for translators.

Finally, JICST's future plans and the evaluation of other translation systems in Japan are also described.

INTERNAL EVALUATION: LOGOS

Bernard E. Scott
Logos Corporation

Logos Corporation observes various procedures for testing, assuring, and measuring linguistic quality of the LOGOS system. Three distinct procedures are used:

(1) At the beginning of a development cycle, to identify and prioritize problems to be addressed in the forthcoming cycle;

(2) At the end of a development cycle, before the cut-off point for a new release, to insure a healthy improvement-to-degradation ratio;

(3) After a release, to measure how the release compares with previous releases, to measure degree of improvement, rate of change in improvement, etc.

Procedure 1 entails processing new corpora through the system, identifying and prioritizing problems, in order to maximize the benefit of linguistic development for the next release.

Procedure 2 will ensure that no system will be released that does not satisfy certain minimum standards regarding improvement/degradation ratios. Ratios must be at least four to one.

Procedure 3 will enable management to measure progress from release to release, measure rate-of-change in progress, and assess overall capability of a given release relative to such goals as Information-Only Translation and Fully Automatic High-Quality Translation (FAHQT).

Internal Evaluation Methodology for the Computing Research Laboratory's Multilingual MT System

David Farwell
Computing Research Laboratory
New Mexico State University

Introduction

Evaluation is always carried out from some perspective and, with respect to the multilingual machine translation system developed at the Computing Research Laboratory, that perspective was defined by the system's design objectives. These were to develop a demonstration prototype which:

translated between Chinese, English, German, Japanese, and Spanish,
was fully automatic,
provided high quality sentence level translation,
was general purpose (domain or text type independent),
was easily extensible (to new constructs and vocabulary, new domains, or new languages),
was robust (provided as good a translation as possible), especially in the face of unexpected input.

In addition, the system was to have an interactive interface with on-line facilities for editing draft translations (supporting Chinese and Japanese character sets), augmenting lexicons, and accessing various sorts of support materials (dictionaries, glossaries, and the like).

The system currently provides translations between five languages (Chinese, English, German, Japanese, Spanish) on a limited scale. It uses an interlingual approach so each of the language components is fully independent of the others. The components are implemented in Prolog as symmetric (or bidirectional) Definite Clause Grammars which use semantic and pragmatic, as well as syntactic, information to equate expressions with corresponding interlingual representations (IRs), or vice versa. Each component has about 200 rules and vocabularies of between 6,000 and 7,500 words based on approximately 10,000 word senses.

The system is interactive. For a given input sentence, the system produces multiple translations, each ideally a legitimate possible translation in some surrounding context and well formed and "natural" as a sentence in isolation. Translation variants are essentially pragmatic (deictic, referential, stylistic, inferential). Our goal is to provide at least one appropriate translation for a given sentence in an given text along with the other possible context-independent translations and then rely on the translator, who does know about the surrounding context, to pick out the best.

The general methodology for developing the system followed a five-step cycle. First, a text was selected and translated into the other four languages. Second, on the basis of a contrastive analysis of the text and its translations, IRs were generated by hand for each sentence, extending the IR apparatus where necessary. Third, the relevant text or translation and corresponding IRs were provided to the different language component developers who set about extending the components to cover the new material. Fourth, the extended components were substituted for the old components and the system was tested over the new set of texts (original and translations) for all possible language pairings. Finally, the new set of parallel texts was added to the regression test corpus and the system was tested and debugged in all directions for the entire corpus.

Evaluation

To assist in steps 3, 4 and 5, testing procedures were designed to ensure that the stated objectives were being reached. The two key questions that need answering were:

> Where are we now?
> Where do we go next?

In regard to the first question, we developed an automatic regression (or validation) testing program for the purpose of collecting data relevant to answering the following four questions:

> Is the system translating between all of the languages?
> Are the translations (for each language pair) accurate and appropriate?
> Is the coverage (lexical, structural) increasing?
> Is the system translating "quickly enough" for comfortable interaction?

In regard to the second question, we developed an automatic but weakly diagnostic testing methodology in order to collect data for answering the following two questions:

> What implemented translation procedures need fixing?
> What new translation procedures need to be added?

The focus in such testing is not on the core problems of translation or machine translation but rather on input-output performance, that is, on how often and how accurately and appropriately the system translates a text. In our case the central metrics gathered include how often an output (IR or translation) is provided, how often an appropriate output is provided, how often an inappropriate output is provided, and how long the process takes. These metrics are informative only to the extent that they guide the developers activities but are not, otherwise, especially interesting or useful. To know how well the system would perform in a particular application, we would first have to specify the application task and then set up a testing procedure within that context.

Regression Testing

As noted, the purpose of regression testing is to collect data that allows the first set of questions listed above to be answered in an informed manner. The testing is done in batch mode. The test corpus is made up of sets of five parallel texts, one in each of the five languages, which have been used at some point or another as a basis for developing the system. The procedure begins by selecting a source language and accessing the file containing the source language equivalent from one of the parallel text sets. Then, for each possible target language, it hands the first sentence of the input text to the source language component for analysis. It notes whether an IR is produced along with the CPU time and real time consumed in producing the result. If an IR is produced, it is counted and stored and, then, it is passed to the target language component for generation. It notes whether an output translation is produced along with both the incremental and cumulative CPU times and real times consumed. If a translation is produced, it is counted and stored and, then, the component is forced to backtrack to look for an alternative translation. This is repeated until all the translations for all the IRs for all the sentences in the initial text have been produced. The entire process is then repeated for each target language, for each source language, and for each set of parallel texts in the test corpus.

There are several options in regard to what data to keep and what data to ignore. For instance, rather than require that all the IRs and all the translations be produced, the procedure can stop after only the first IR or the first translation of a given sentence is produced. Similarly, specific source languages or target languages or parallel text sets may be selected for testing.

Finally, the results may be compared with results of prior tests automatically or they can be distributed to the various language component developers for review by hand.

In regard to whether the system is translating between all the different languages, the output of the regression test is inspected for “no translation can be provided for this sentence” messages. Where these are found, the target language, source language, sentence and text are noted.

To see if the translations are accurate and appropriate, we begin by comparing the output of the regression test with previous regression test results for “X translation for current input IR” and “Y IRs for current input sentence” messages. Any differences in the number of translations or of IRs produced signal changes in the input-output behavior that need to be examined. These differences are noted along with the target language, source language, text and sentence. Where there are divergences, the set of translations or the set of IRs are reviewed to order to qualitatively identify what changes occurred and, in particular, to see if the differences arise from overgeneration or undergeneration of output.

Next, for each text, we select, sentence by sentence, the best target language translation in the test results and assemble them into a target text. In regard to accuracy, we then compare the original source language text and output target text for similarity of content. In particular, we are looking for any source language sentences that give rise to misinformative or underinformative translations. The target language, source language, sentence and text are noted. In regard to appropriateness, we inspect the target text for readability or naturalness of expression. Here, we are looking for any source language sentences that gives rise to awkward translations. Again, we note target language, source language, sentence and text.

As for coverage, if the extended system is providing appropriate translations for all source language-target language combinations for all texts, then increased coverage is determined by inspecting notes kept by the language developers regarding changes made (to rules or lexicon) in order to extend the components to process the new development text set. Essentially, coverage is extended if the changes to a component’s rule system are motivated by the addition of new constructs in the language or if new vocabulary has been added. All other changes are assumed to be revisions.

Finally, in regard to the system's response time, we compare the output of the regression test with previous regression test results for “X cpus/seconds to produce current IR” and “Y cpus/seconds to produce current translation” messages. We note any significant differences in the number CPUs or seconds used to produce same results along with the target language, source language, sentence and text. Where there are significant differences, we check if they give rise to “uncomfortably long” interaction (roughly 5 seconds in real time).

That regression testing shows that the system is growing is based on the tacit and, strictly speaking, incorrect assumption that if some sentence in the test corpus exhibits certain lexical and structural properties, then those properties have been adequately addressed. When this is the case, then the system probably is growing. When it is not the case, the testing tells you nothing. In reality, there is no way of knowing that any rule, no matter how well considered or well implemented, will not have to be changed. Nor do the results inform the developer how easy it will be to debug the component, to transpose the system to a new domain or language, to increase the size of the lexicon or rule system. Such estimates are based on performing those sorts of tasks and keeping records of the efforts involved and then drawing an analogy between the task to be done and one of those documented tasks.

Diagnostic Testing

The objective of diagnostic testing is to provide data to inform decisions about what needs improvement and what to focus on next in extending the system. The method of testing is somewhat different from above in that normally subparts of the language components corresponding to any of four different structural levels (the clause, the major constituent, the phrase and the lexical) are applied to the input since the goal is to identify what and where internally the analysis process breaks down. Also, the text and the source language are given at the outset (i.e., the new text is, in fact, in only one of the five languages in the system).

The general procedure is to iteratively apply successive subsets of the rule system, each corresponding to a constituent category at some level of analysis (clausal, major constituent, phrasal, lexical), beginning at the first word of the first sentence in the input. If there is a successful analysis, the initial part of the input that is covered by the analysis is removed and the processes is repeated on the remainder of the input. If there is no successful analysis, the initial word of the input is skipped and the process is repeated beginning with the second word.

Focusing, for instance, on the major constituent level, the rules for analyzing each constituent type (predicates, arguments, circumstantials, etc.) are applied type by type to the initial part of the input following the most common order of call from the clause level. Thus, the rules for predicates are applied first and the time of processing (CPU and real) is noted. If the rules fail, the rules for arguments are then applied to the initial part of the string. If the rules for predicates succeed, the result is both stored and handed to the corresponding set of rules for predicates of each possible target language component, one language after the next, and they attempt to generate a target language expression noting the processing times (CPU and real). If the rules of the target language succeed, the results are stored and the system is caused to backtrack to look for another translation. Eventually, the rules for predicates of the source language component fail at which point processing begins on the remainder of the input.

The possible constituent types that can be tested in this manner include: sentences, major constituents as defined by the IR system (i.e., predicates (actions, states), arguments (subjects, direct objects, indirect objects, etc.), circumstantials (locatives, temporals, causals, purposives, manner, modes, etc.)), phrasals (entities, relations (activities, properties, relationships), and lexical level (entity names, entity specifiers, cases, relation names, relation specifiers, relation modifiers, conjunctions).

Again, there are several options in regard to what information to keep and what information to ignore. For instance, rather than go for all IRs and all translations, you can store only the first IR and/or the first translation. Rather than apply the rule subsets at all levels you can focus on only one level. Rather than apply all the subsets at one level, you can focus on the subsets for specific categories, and so on.

In this case, the results must be inspected by hand by the various language component developers since there are no prior test results to compare the results with automatically.

With respect to the questions of what implemented translation procedures need fixing and what new translation procedures need to be added, the best we can expect from the results of diagnostic testing is to have our attention directed toward one rule subset or another by way of “no translation (or IL) can be provided for this expression” messages or by way of poor translations. But the inspection process is time consuming and the amount of data to be sifted through can be massive. In the end, even with pointers to problematical input-output behavior, extending the system requires locating the problems in the code, designing and implementing (or reimplementing) solutions to those problems, and running the tests again.

Summary Comments

There are three major observations that we can make on the basis of this experience. First, there is a significant gap between identifying the system's inadequacies in terms of throughput (i.e., the results of an evaluation) and fixing those inadequacies. The internal evaluations helped focus our attention on problems areas in the code, but they did not tell us what the problems were or what alternative approaches would be better. Second, there is a logical leap between the an analysis of the results of some testing program and conclusions about increased coverage or improved functional characteristics. The analysis tells about improved input-output behavior but not whether that improvement is due to extending the vocabulary and rules systems of the components or to qualitative revision of existing vocabulary and rule systems. Third, there is a difference between testing and evaluation for the purposes of developing the system or improving its performance and testing and evaluation for the purposes of establishing the system's potential application to some specific translation task. Specific text types, specific topics, specific constraints on the type of user and type of interaction, specific uses for the output, and so on must all be taken into account for informative evaluations of the latter type.

The multilingual machine translation system at the Computing Research Laboratory is a research prototype providing translations between five languages (Chinese, English, German, Japanese, Spanish). It uses an interlingual approach for which each of the independent language components, implemented in Prolog as a symmetric DCG rule system, maps expressions onto interlingual representations, or vice versa, without regard to how the representation will be used.

With respect to evaluation, we have developed two basic testing mechanisms:

- Regression testing: providing evidence that the system is growing;
- Diagnostic testing: providing data for analyzing requirements for extending the system to cover a novel corpus.

All system evaluation based on data provided by these tests is informal. That regression testing shows the system is growing is based on the tacit, and incorrect, assumption that if some sentence in the test corpus exhibits certain lexical and structural properties, then those properties have been adequately addressed. As for diagnostic testing, the data only directs attention to existence of "failures." Extending the system requires locating problems, analyzing them, and implementing (or reimplementing) solutions to them. Informal evaluation consists of keeping notes about the problems and solutions encountered while modifying a component to map between such expressions and proposed IRs.

INTERNAL EVALUATION: Quality Analysis, an Internal Evaluation Tool at SYSTRAN

Elke Lange and Laurie Gerber
SYSTRAN Translation Systems, Inc.

A quality analysis methodology was developed at SYSTRAN in 1986 to serve as a tool for internal evaluation and to inform current and potential customers regarding the status of translation quality to be expected for each language pair.

The SYSTRAN quality analysis (QA) measures linguistic aspects of translation quality by identifying error categories and weighting them according to severity. The result is expressed as an overall percentage of correct translation per number of words translated.

The test corpus is a text of about 350 sentences taken from three unrelated documents of a technical/scientific nature. Every translation error is marked by the evaluator and entered on a tally sheet under error categories which fall into the following groups: (a) analysis errors (incorrect part of speech, syntactic relations, clause boundary), (b) synthesis errors (incorrect target language form, article usage, word order), (c) words not found in the dictionary, and (d) incorrect meaning or capitalization.

The latter two categories refer mainly to the quality of the dictionary and transfer rules, whereas the first two reflect the state of the analysis and synthesis modules. This, however, is only an approximate correspondence to the system's modules, since the QA evaluator judges only the surface translation without performing any in-depth error analysis.

In addition to categorizing the errors, the evaluator also assigns one of the following severity weightings: "1" for a minor/cosmetic error, "2" for an error with medium impact, and "3" for a high-impact error. The criteria for judging impact on the translation of each sentence are: correctness of semantic transfer, readability, and comprehensibility. Style is not judged, since our QA does not evaluate raw MT output against an ideal human translation.

The errors are tallied by type and severity with the number of errors of severity "2" or "3" is multiplied by 2 or 3, respectively. The resulting error points are assessed as a percentage of the total number of words in the text. This percentage is then subtracted from the maximum possible accuracy of 100%

Typical results range from 70% for a "young" language pair to 85% and above for more mature language pairs.

The same test corpus is re-run periodically, and since no corrections are made specifically for this text, the subsequent results give an accurate picture of the improvement that has taken place in the language pair tested.

Long-term experience with this method of evaluating SYSTRAN translations, as well as knowledge of evaluations done by users of SYSTRAN, has shown the advantages as well as the limitations of our approach.

Advantages

Strength of the method. The QA results are fairly consistent, with only about 3% variation for a given text in a single language pair system, even for different evaluators.

Usefulness of the results. The percentages produced by our method of evaluation are meaningful. We know that a language pair becomes a productivity-enhancing tool at around 80% accuracy. Periodic quality analyses provide a measure of the quality of SYSTRAN language pairs relative to one another, as well as of individual language pairs over time.

Limitations

The method was developed with two objectives in mind: (1) to inform potential users; and (2) to identify problem areas for development work. However, one method cannot fully meet both objectives. Our method works well for benchmarking, but because it attributes errors to system modules only approximately, we employ other methods to direct our development efforts (e.g., periodic glass-box evaluations, systematic collection of categorized errors, regression testing, and user "improvement requests").

SYSTRAN language pairs are also systematically evaluated by our customers. Xerox Corporation, for example, tests our new language pair systems for a minimum percentage of accuracy using a method similar to our own, and they have even incorporated percentage points in their contract specifications. They will accept a pilot system at a 70% level of accuracy, and a production system at an 80% level of accuracy. These numbers are based on their experience with the level of accuracy that must be achieved in order to yield cost savings in translation.

In contrast to our long experience with evaluating linguistic quality, we have only recently begun to consider the requirements in the production environment, where linguistic quality is only one of many factors to be considered. We recognize the need to develop further evaluation measures for a full commercial evaluation of MT systems.

INTERNAL EVALUATION: METAL

Winfield Scott Bennett
Siemens Nixdorf Information Systems, Inc.[1]

All machine translation systems require constant evaluation from the start of research and development to the end of their life cycles. From the beginning of our work in development of the system to the present when METAL is a product, it has been apparent that we have to keep track of the state of the system constantly. The initial difficulty was that there were no existing standard ways of measuring the state of any system, which led us to create our own internal evaluation metric. This metric has never been a static device, however, since we have found it necessary to refine it at various points within the history of METAL Our internal evaluation continues to this day with some further refinements to the evaluation metric as necessary.

This paper will present an overview of the internal evaluation metric for METAL. Since our metric has developed throughout our 14-year history, the paper will discuss the issue from a historical perspective, indicating the issues that motivated our choice of particular approaches at given stages in the development of METAL.

[1] Now at Logos Corporation, Mt. Arlington, N.J.