[from *Linguistics Research Center*, University of Texas at:
<http://www.utexas.edu/cola/centers/lrc/mt/index.html>]

Machine Translation at Texas

Overview

Work on machine translation (MT) at the Linguistics Research Center was originally supported by grants from the U.S. government; unlike most MT projects in the U.S., which explored Russian-English translation, the LRC investigated German-English translation. The system came to be called called METAL; whether this stood for "META-Language(s)" or "MEchanical Translation and Analysis of Languages" is knowledge now lost in the mists of time. A personal commentary about this early MT work, by Winfred P. Lehmann, Director of the LRC, is found in: Machine Translation at Texas: the Early Years.

Funding by the U.S. government for MT work at the LRC ended ca. 1975, though very small amounts "for system documentation" became available a few years later, after all the systems personnel and most of the linguists had left the University. Knowledge of the software and how to run it was all but lost, and in any case major hardware changes at the University were soon to render that knowledge moot, and the system unusable.

Eventually, in late 1978, the German firm Siemens AG stepped in due to a growing corporate need for high-speed translation. (*Of course* MT would come *with* human post-editing, because post-editing is standard practice even for human translation at any self-respecting company. Americans who had scoffed at the feasibility of MT due to post-editing requirements were, at the very least, remarkably ignorant.)

Funding from Siemens enabled the LRC to hire new systems personnel in 1979, who delivered a full production prototype by the middle of 1984, to be followed by commercial implementation in C++. A personal commentary about this critical phase of system development by Jonathan Slocum, then MT Project Director at the LRC, is found in: Machine Translation at Texas: the Later Years.

As METAL evolved into a commercial product, both for in-house use and for licensing to other firms, Siemens gradually absorbed all development work, and LRC MT personnel either became Siemens employees or contractors, or engaged in MT research funded by other sponsors (such as Hitachi for Japanese/English translation), or found other work. By mid-1996, work on the METAL system at the LRC had come to an end, and general MT R&D did not continue for too very long thereafter.

The Early Years

Winfred P. Lehmann

On my return from directing the Georgetown English Language Program in Ankara, Turkey, during the years 1956-57, Léon Dostert suggested that I undertake machine translation (MT), with concentration on German since he was doing Russian. Having spent World War II at the predecessor of NSA dealing with translation of Japanese messages, where predecessors of the computer were lined up by scores, I had not been without interest in Weaver's suggestion that the newly developed instrument be employed for MT. But other obligations interfered with more than keeping abreast of reports on activities. Dostert sweetened his suggestion with a grant of $10,000, a sum large enough, in the days

when professorial salaries scarcely reached half as much, to employ a good group of graduate students.

The work consisted chiefly in learning what was going on and planning. The University's computer, an IBM 650, was preempted by physicists, for what it was worth. Our best informed member in computational areas was Nick Hopkins, a graduate student in archeology, who among other capabilities had learned on trips to Mexico how to make sandals by cutting strips from old tires and keeping them on with pieces of string. Most of the other members were linguists well acquainted with German. The best result of the year was an award of $300,000+ from the U. S. Army Signal Engineering Laboratories to investigate **the feasibility** of MT. The Army was planning to develop a computer, Moby Dick, to be located at division headquarters; MT was to be one of its functions.

The grant provided means to employ full-time programmers as well as linguists. One of these, Gene Pendergraft, had worked with Minsky, and soon became the chief systems specialist. We set up three groups: linguists, programmers, and mathematicians. Sponsorship by the Signal Corps gave us the advantage of using the IBM 709 at Fort Huachuca. Programs and translation algorithms were developed, prepared by our cardpunchers for entry, and taken out to Arizona to be checked. Our progress was recorded in quarterly reports. The third, of 31 January 1960, and the ninth, of 31 July 1961, state the theoretical basis of all our further work, which led to the production of the METAL translation system. Following the theoretical position of Charles Sanders Peirce we treat language as a system with three components: syntactics, semantics and pragmatics. Programs were designed to manage the linguistic rules and were kept distinct from the grammar and the lexicon.

Through personal contacts, conferences and publications, we kept in close touch with other groups and profited by their conclusions. Dostert generously sent down his specialists to inform us of his progress. Oswald at UCLA credited C. V. Pollard in our department with developing syntactic rules for scientific German, as did Bar-Hillel (1964:159), at a time when language teaching concentrated on morphology. Reifler of the University of Washington developed the notion of sub-languages, basically an application of pragmatics that recognized the presence of distinct types of language; MT could concentrate on technical language rather than attempt to control such sub-languages as that of literature. Our progress was recognized by further grants, and by the founding of the Linguistics Research Center in 1961.

The conferences kept us in close touch with members of an expanding technology. Dostert remained the operational father of MT; known at the highest levels of government through his war-time duty as French translator for General Eisenhower, when funds were running out he simply called on Allan Dulles, who turned him over to his staff with the injunction to 'give Léon what he needed.' Victor Yngve must be credited with producing the first appropriate computer language, COMIT, and with producing a journal for the field. Yehoshua Bar-Hillel, a logician who might be called the theoretical father of MT, insisted on FAHQT (Fully Automatic High Quality Translation), then becoming highly pessimistic and vocal about its realization. Curiously, one of the major proposed shortcomings was the need of post-editing, a regular procedure among translation agencies. His unfortunate insistence, clear still in the "Feasibility Study on Fully Automatic High Quality Translation" produced by the Center in December 1971, resulted in part from ignorance of translation procedures, as did the devastating Academy report of 1966 by the engineer John Pierce. By contrast the far-sighted supporter, Zbigniew L. Pankowicz of Rome Air Development Center, may be credited with keeping the technology from vanishing by providing the funding for LOGOS and the Center.

The measures required to achieve results hardly seem credible in recollection. Access to the computer was through punched cards; to cut costs Dostert established a key-punching center in Frankfurt, Germany, where labor was cheaper. Rules and algorithms were transferred to tapes, which whirred crazily in the computer room. In the absence of computer languages, coding was done initially in machine language, then in Fortran. When the University acquired a top-notch computer, the CDC 1604, we were allowed to use it Sunday mornings, because testing our programs required full use of it. Somewhat later the Center acquired its own, a lesser CDC, so that our systems personnel no longer had to defile the Sabbath. But the pitiful success provided as much assurance that MT was feasible as Newton's apple did for his views on gravitation.

Yet Bar-Hillel's vigorous negativism, coupled with unethical activities, succeeded in undermining success. As Bar-Hillel stated (1964:9-10), funds for MT were siphoned off, not least in the institution that was his home, so that the amount invested was larger than that applied. Another damaging effect was produced by "demonstrations" that had been pre-cooked, as by one defector from Georgetown at the International Conference on Machine Translation and Applied Language Analysis, Teddington, England 5-8 September 1961. Even the self-sacrificing support provided by Pankowicz could not offset the devastating effect of the shameful Academy Report of 1966 that eliminated almost all research in this country and abroad. Fortunately, with his support the Center was able to continue its work until Siemens found that their human translators could not keep up with the demands of quantity and time. Their funds and that of American agencies maintained research at the Center so that METAL became an operative system which Somers has credited as "a success story in the development of MT" (Whitelock: 1995:198).

Winfred P. Lehmann,
May 17, 1998

References

- Bar-Hillel, Yeshoshua. *Language and Information.* Reading, Mass.: Addison-Wesley, 1964.
- Whitelock, Peter and Kieran Kilby. *Linguistic and Computational Techniques in Machine Translation System Design.* London: UCL Press, 1995[2].
- Review of Whitelock & Kilby by W. P. Lehmann: *Machine Translation* 12:261-69, 1997.

The Later Years

Jonathan Slocum

When I was hired to direct the machine translation (MT) project at the Linguistics Research Center in January 1979, I knew virtually nothing about MT, although I was well-educated and highly proficient in the encompassing discipline of natural language processing (NLP) or, in a broader sense, computational linguistics. Only gradually, over a period of years, did I learn why I had remained so ignorant.

A report by the Automated Language Processing Advisory Committee (ALPAC), commissioned by the National Academy of Sciences, had been released in 1966; the report's conclusions were that applications of Machine Translation were infeasible, and that funding for applied MT projects should be redirected into research on more basic, underlying NLP problems. The not-so-veiled implication of the political uproar that ensued was that MT system developers were suspect, if not outright frauds.

But while MT development funding was indeed eliminated virtually across-the-board -- only a few MT *research* projects managed to survive, none unaffected -- funding for NLP projects *in general* was reduced or eliminated. Naturally, NLP workers blamed these broader funding cuts on MT system developers, who became pariahs in the NLP profession. Machine translation was not discussed in polite professional company, and universities (in the U.S., at least) tended to ignore the discipline altogether. Intellectually, I was a child of the next generation: I knew exceedingly little about MT, and nothing about events that led to its becoming taboo before my college days.

When I arrived at the LRC, the project was in shambles. The old METAL system, implemented in FORTRAN and at least hundreds of thousands of lines long, was semi-documented but useless: a run required several hours of access to a CDC 6600, once considered a supercomputer and still a campus mainstay, *with parts of the operating system removed* -- thereby making the hardware unavailable to anyone else. For the LRC in 1979, the days of having a dedicated supercomputer were gone. Tiny bits of LISP code, written by a graduate student, could cripple through some process that when successful, which was extremely rare, vaguely resembled MT; but it took an entire day to complete a "short" batch run on a time-shared computer. If an error was encountered, there were few or no clues as to what or why, and days of effort were generally required in order to identify a likely cause.

I proceeded to study system documentation, to the extent that any existed, and interviewed the project linguist about how the system was thought to work. I then wrote an entirely new METAL system in LISP, using all new algorithms, which could produce translations of sentences in seconds -- online, not in overnight batch mode -- and which, when an error was encountered, would emit diagnostics. It became possible to test linguistic rules online, and modify & test them again in short order. The light of progress began to fall on the project.

A curious thing happened when I commenced MT R&D: some of my professional colleagues seemed to stop speaking to me -- or worse. Dawn began breaking on my ignorance when, at a conference in 1980 where I presented my first MT paper, the question period was consumed by a husband & wife tag-team who proceeded to scream at me for the duration. *No question was being raised*. I realized that something was very wrong... My new interest, I later came to realize, was taboo. On the other hand, I had one great advantage: I was not aware that MT was infeasible, and that my hopes were doomed. Perhaps this ignorance accounted in part for the subsequent success of our project! But that's another story.

It became obvious, very early in our project, that analysis & translation problems were too often due to simple human mistakes, such as typographical errors in grammar rules and lexical entries. Once a usable translation engine was in place, I began to devote my programming efforts to *early detection* of "obvious" errors in linguistic rules, in order to point them out before they had a chance to spoil a translation run. This became systematic, so that **translation was attempted only with *validated* systems of rules**. It was amazing that few if any other NLP projects used "rule validators" to catch errors in new/revised linguistic rules; but, then, remarkably few NLP projects of any kind dealt with systems having much more than a handful of rules, so validators might seem unnecessary, even odd. Nevertheless, once I published a paper describing rule validators, I began to notice this technique being practiced elsewhere.

The next, seemingly logical step was the provision of special-purpose tools to help linguists *create and edit* rules -- especially lexical entries by their thousands, but other rules as well. Naturally, a well-written software tool would never define a rule containing

an error. (In this context, *a text editor is not well-written software*.) And since **humans are much better at selecting items from lists than typing them in**, possible "closed class" values were made selectable in a GUI interface, reducing typing to a bare minimum. Again, these techniques seemed to be unknown in the NLP field until I published them.

A subsequent "toolmaking" step was to create software that would *guess* what a lexical entry might look like, based on analyzing the new word form and finding any similar words already in the dictionary. **Humans are much better at verifying content than creating it**. With a "guesser" tool, a lexicographer seldom faced a *tabula rasa* when coding a new entry; rather, a plausible lexical entry would be "guessed" and made available for editing. Surprisingly often, guessed lexical content would be correct, leaving only "unguessable" (and therefore more interesting!) content up to the lexicographer. Our German and English lexicons grew to contain tens of thousands of entries. Abroad, other projects using the general-purpose METAL software extended coverage to Spanish, French, and Dutch.

At each step along the way, then, system-building tools made it yet easier for *non-programmers* to create, test, and debug linguistic content. Indeeed, as our project grew in terms of personnel, the level of programming expertise remained much the same: instead of programmers, linguists were hired to use tools crafted especially for them. Another toolmaking project concerned the use of the MT system in practice, i.e. not by system developers or linguists, but by actual translators. Early on, tools would extract text from formatted documents and then replace that text with its translation. Later, other tools were developed to manage the translation process: searching texts for unknown terms before translation commenced, scheduling texts for translation when the lexicon was updated, etc.

The key to developing a practical software system is to "program oneself out of a job" by developing tools for non-programmers. After all, there are always other systems to create! In the summer of 1984, a full production prototype translation system was delivered to Siemens, our sponsor, and I moved on to create another system...

Jonathan Slocum,
November, 2006