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

PANEL: ASSESSING THE LABOR-INTENSIVE ELEMENTS OF MT

Introduction: *Marjorie León,* Moderator
Pan American Health Organization

We've assembled a small group of people here with a large amount of experience in using machine translation systems, and we would like to let them convey to you some of their experience of how much work it is, at the different points in the process, and how, once you find a system whose quality you deem to be acceptable or adequate for your purposes, you must then consider how much work it's going to take to *get* the output of that quality. The developer may have been able to produce it for you when you sent in your corpus, but how much work is it going to take for *you* to produce it on the next selection?

We're going to discuss four major topics: (1) the work involved in preparing the input text; (2) the work involved in updating the dictionary; (3) what's involved in actually running the text—getting the output back on the screen; and, finally, (4) postediting. Doris Albisser will start by giving us her heartfelt opinion on the problems of input preparation.

Panelists

Doris Albisser, Union Bank of Switzerland: At Unibank I head the Computer-Integrated Translation and Terminology Department. We have done extensive tests with most of the commercially available systems that can be used out of German into English, French, and so on, and we are currently working with the METAL machine translation system, which we introduced this year.

As you might know, working for an international bank means we have to provide high quality. When we tested machine translation systems, there was no point in having a system that was just used for information-only translation. Every translation has to be postedited to the point that it's a high-quality product.

This morning Scott Bennett said that all the customer wants to see is the output. I think that's true to some extent. But in the beginning, we would be very happy to get the text into the system first. That proved to be one of the major problems that we've had so far, particularly this year. Before that we were working with the FrameMaker software; we could convert FrameMaker documents and submit them to the METAL machine translation system without any problems, and they came out with a good layout afterwards. Otherwise, the input can be a major problem. Currently we're working in a heterogeneous environment with WordPerfect documents and other word-processing formats.

We have kept statistics on how much time each translation job takes. At the very beginning, our first translation with METAL was a 21-page document, and it took about 8 hours to get the whole text into the machine translation system. This was due to at least three factors: first, the text was not in a form that could be read in automatically; second, we had to insert brackets to mark those portions of the text that we did not want to have translated; and, third, we had difficulty getting the job to run because of problems with the METAL internal translation software. So the difficulties in this case were not so much on the linguistic side of METAL but in the pre-editing stage, because the text formats could not be read in. When the text was incorrectly converted, the translation units themselves were incorrectly analyzed and the output was rubbish. Another problem was that the tables in the documents were being read in incorrectly as well. As a result, the translation units were not being handled correctly and we were getting lousy output afterwards. So the input stage was slowing down the process, which was very frustrating for the translators, who felt that reading a text into the computer was a tedious and low-qualified job. The software supplier told us, "If everything fails, just read in ASCII files," but ASCII files did not capture the format for us, which was part of the point of using machine translation.

The second time-consuming task, of course, is building up the domain-specific dictionaries. I will not go into very much detail here. The only thing that I will tell you is that *our* translators prefer that kind of work to preparing the input text. Coding a dictionary is at least qualified work.

It seems to me that, to speed up the translation process, the solution is for software suppliers to focus more on providing a system that fits into the document production process. In the end, translation is part of an entire document production process and it has to fit into that environment. For that, it needs front-end tools for inputting text and getting it out in a good format.

Marjorie León: I can tell that Doris starts with text that is already in electronic desktop publishing format. Perhaps some of you may be considering beginning with text that is in hard copy, in which case you'd have to deal with the problem of scanning, which is the first thing on our list. We receive many inquiries from people asking "Can we use machine translation? What kind of machine translation system should we get?" We ask them what they want to translate, and they often want to translate letters that have been received from abroad. Of course, they're not receiving this correspondence on a disk, and it may even be a Xerox copy or maybe even a fax. So, the problem, then, is that we will be dependent on scanning technology if we want to try to use machine translation on material that is not machine-readable to begin with. We've seen an advertisement for a machine translation system that says you can "receive a fax, press a button, and it will be translated in one minute" or something like that—maybe some of you have seen that advertisement. Well, if any of you have ever tried to use an optical scanner on a fax, you know that there's a lot of proofreading and correcting to do, and it's not a matter of minutes before you'll get your translation.

Now, we'll move on to dictionary updating. Some systems recommend, or even require, that you run a software utility first to get the list of not-found words before the text can be translated. In this case you *must* do your dictionary updating next. In other cases, you can run your translation and do your dictionary updating after the fact. Gudrun will address dictionary updating.

Gudrun Magnusdottir, University of Gothenburg: I have basically been involved with EUROTRA evaluations, which were political things, and I'm currently involved in that again. I did a survey of machine translation systems for the EUREKA project in Europe, which meant that I looked at METAL, LOGOS, and ALPS.

I consider that you're *lucky* if you get to do dictionary updating. Sometimes the user is not allowed to. There are systems around that aren't prepared to rely on the capability or intelligence of the user to do the lexical updating. I think this is probably the only market in the world where the sellers say, "Our systems are so complex that you can't use them."

Even though you're lucky to be able to do it, dictionary updating is still the task that takes the most amount of time. Moreover, it is a long-term aspect of using the system. Terminology is always growing, so if you're in a company that is producing mechanical products, you can count on the fact that there will be new words every week; this task of updating the terminology is continuous.

The result of your dictionary updating is not very satisfying when you know that the grammatical encoding you're *allowed* to do is only superficial and that it will not give as good a result as the rest of the dictionaries in the system. It's not a very gratifying situation. You know you're doing a lot of work, but it's not really getting to the point—you're not *allowed* to get to the point—so I would say that it's a frustrating experience. I hope some of you here agree with me that the companies that sell MT systems should be a bit more open about their content and a bit more generous in their opinion of the user after selling the system to somebody.

Larry Childs, NOVELL: At NOVELL I deal with adapting our software for foreign countries. Previously I worked on the WEIDNER system. I worked for 5 years on development of the German-English program and then did the same thing with the ALPS translation system.

I’d like to tell a little anecdote to support what Gudrun said. I have a friend who developed the dictionary entry system for French at WEIDNER. He did a lot of dictionary coding. WEIDNER had a user version of dictionary entry which was a subset of the *real* version of dictionary entry. In the real version of dictionary entry you could put codes in a lot of fields that we didn’t think the user would know how to handle. Also, we had a pretty clunky way of entering those fields, and we were a bit ashamed of that, so we didn’t want to let users see that. My friend left WEIDNER to work for a company in England that was using the WEIDNER system. Because he was suddenly a user of the system and no longer a developer, WEIDNER did not entrust him with the in-house dictionary-encoding schemes. He was no longer smart enough to know what to do with it! So it really is a problem—I agree.

Margarita Baena, International Center for Tropical Agriculture: I’m in charge of the Translation Service at CIAT. We basically translate material from English into Spanish, using ENGSPAN, the PAHO program.

As a user, I don’t mind updating a dictionary and investing the time as long as it results in quality—as long as the output reflects my updating work. I’d rather have a system that allows for updating and enlarging—inputting different meanings for different words—than stick to a system that is so rigid that I can’t do anything with it. I think that dictionary updating is a necessary evil, and as a user I don’t mind doing it as long as it improves the quality of the output.

Marjorie León: I think, as Doris said in the beginning, that dictionary updating is a more intellectual task. And it does have a certain *permanence* about it—the translation goes out the door, but the dictionary stays home. So there is some satisfaction in it.

In PAHO, we’ve been increasing the complexity of our dictionary coding because we’re linguists and we want to be able to control *everything* and to be able to get every nuance into the dictionary. We are now contemplating, perhaps, having other people use the system. Right now the only user outside of PAHO is Margarita, who has learned her dictionary updating, but we’re not positive that every other user is going to be able to catch on as quickly and as easily as she did.

That brings up two questions, “How much does the novice *need* to know?” and “How far can the expert go?” We would not want to restrict access to our dictionary, but we feel that we ought to have a separate coding manual for people who just want to know how to get in and do the basic thing. We’re finding that if we include all the information in one place it gets much too complicated. We feel that the user doesn’t have to know the internal representation of the codes, but he or she should be able to activate any code for any feature that is offered by the system. If the system can recognize and give you a translation for a transitive, an intransitive, and a bitransitive verb, then you ought to be able to code them and get these different translations.

Gudrun Magnusdottir. I was going to say that if a system is so ad hoc that it will not allow a user to update the dictionary, then, I think that system probably won’t—or shouldn’t—last too long on the market. And if a user who uses a system—does postediting and so forth—is not capable of doing the dictionary entries, there’s definitely something wrong with the way the system is constructed. It has nothing to do with the intelligence of the user, it is simply something wrong in the construction of the system.

Marjorie León: I know that some systems have a number of restricted entries. To say that verbs would be restricted might be going too far, but prepositions and conjunctions probably should not be re-coded. Yet I think that the user should be able to see how they *are* coded, whether they can modify them or not. If they’re being told that the system can handle certain types of conjoining, then they ought to be able to see what types of conjunctions they’re dealing with.

Gudrun Magnusdottir. I can understand that you would not want to let the user update the verbs in order to maintain the quality of the parse of the system. However, I find it *wrong* if you cannot explain to a user how to do all of it, because these are language people who usually have gone through 5 years of translator's school. I mean, is it so *difficult* to be communicative with a user?

Marjorie León: Obviously, we are all thinking of users of the type of systems that *we* are dealing with—not the users of MICROTAC—at this point. Mike Tacelosky clearly says that MicroTac's users are *non*language people. I guess our comments need to be taken in context. I don't know if there's an answer to the question, "Can the user modify the supplier's entries?" As I said, conjunctions and prepositions, perhaps, ought to be protected, but other words that are supplied in the dictionary might have some sort of a code on them that would trigger a message like "Are you sure?," so that the users would know that if they make changes, they may encounter problems. Also, perhaps there could be a little backup file—a way to restore the original supplier's entry—if the user's brainstorm didn't turn out to be the right one.

I think we'll move on to our next topic, which is running the translation. Since we have two or three different modes of translation—ranging from batch to very interactive—we'll have to discuss different kinds of things. Larry will start out.

Larry Childs: I think I was put on this panel because I had extensive experience at ALPS running a fully interactive system. But in order for you to understand my remarks, I need to put them in context and talk about my professional life. At WEIDNER I spent 5 years working on a batch approach—a fully automatic approach with no interaction. In the mid-1980s WEIDNER and ALPS were fierce rivals, and I spent 5 years pooh-poohing the idea that interaction could be any good for anybody. Well, as it happened, I ended up at ALPS working as a German linguist, even though I still had some philosophical differences on the idea of interaction. However, they began to be allayed after I had been at ALPS for a while. So I would like to give you a few arguments in favor of using an interactive system. But first, let me just preface this by saying that you have to do some editing on the text at some point, either fixing up the text before you put it in, editing the text *during* the translation process, or postediting it afterwards—or some combination of the three.

We talked a lot about dictionary entry as a way for the user to influence the quality of the translation. This is also one of the arguments in favor of interaction. If you discover in a batch translation that you have mistranslated one of the words in your dictionary, and you have already run 1,000 pages of translation, you have a lot of searching and replacing to do at the end, and you just have to hope you have caught all the instances of this mistranslation. The ALPS system, on the other hand, worked on a sentence-by-sentence basis. It would bring up the source sentence on one side of the screen and the target sentence on the other side. If you didn't like the translation of the target sentence, at that point you could go into the dictionary and update it—either make it more accurate with new grammatical coding rules, or change the translation—and then the change has been made for the rest of the translation. So that is one of the strongest reasons, I think, for doing interaction. It helps you *prevent* mistakes before you've run the entire batch and then have to go back and correct things that could have been corrected as you were going along.

The other thing about interaction that I did not understand when I came to ALPS, and I think many people don't understand this, is that the goal at ALPS was not to ask rules that had to do with the syntax of the source text but rather to interrogate the user with regard to the style of the target text, so that you could actually do some stylistic editing (what you would normally do in postediting) as you were going along.

Let me give you an example of a source language question. If you had the verb "chew" in your dictionary and you had the noun "gum" in your dictionary, and there was a phrase that had the words "chewing gum" in your document, the ALPS system was smart enough to say, "Ah, this is a participle and this is a noun. I don't quite know how they relate." So you'd get a question like, "Is this gum that is chewing, or is this gum for chewing?" And then you'd have to answer that kind of question.

On the other hand, an example of a target language question in the German-English system might be the genitive case, which in modern German has the genitive object coming after the main noun, whereas in English you have the choice of either using "'s" or the preposition "of." For example, you don't necessarily talk about the "table's leg"; you'd want to say "the leg of the table". You could ask that question during interaction, and that would influence the style of the target language. Those are the kinds of questions we were trying to develop. We tried to spare the user from answering a lot of parsing questions—i.e., helping the system to parse the text—although we did have those, too.

Do I think that interaction was really successful? Actually, no. I think there are some theoretical reasons why it *could* be successful, but in practice, no matter how sophisticated we made our interactive questions, it was still necessary to postedit the text. Therefore, if you're going to have to do postediting, you might as well do it all at one time. ALPS gave up on the idea of a fully transactive system before they gave up on selling their machine-aided human translation aids.

Finally, let me just make one more comment on the modern interactive approach—systems that claim to "learn" as you go along. If you don't like a translation, you're supposed to be able to tell the computer—give it new rules so that it will know how to translate better. I haven't dealt with these, but let me just mention it because they're a form of interaction. My impression is that there is no way that you can give questions to a user that are really going to affect the internal workings of the system. You cannot reprogram your system by asking those kinds of user questions—it just will never work. So I really think that these heuristic approaches are too simplistic to be of any value.

Marjorie León: Okay, I think we'll move on to postediting, which is the alternative. Margarita will talk about postediting, since she's been using a batch system that has no possibility of interaction.

Margarita Baena: You never miss what you've never had, so I don't miss interaction. There's nothing you can't fix at the postediting level. That's an advantage. Looking at machine translation from the point of view of the user is looking at how it works in the real world. Users are not expecting perfection from the system. Users are expecting to be able to work with it—to team up with the system and be able to meet the needs of the customer. We want customer satisfaction, so we buy a product depending on how we can use it to meet the needs of an end user. If we can translate the benefits of our product in terms of volume, in terms of time saved, in terms of money, then we buy. Otherwise, sorry, we don't buy.

Users speak the language of words, not megabytes. I would say that output is what we look for. Postediting depends on output. Postediting involves making the necessary changes so that the end user will have an acceptable product. We look at it from the point of view of such questions as: How many changes? How many keystrokes? How many pages of machine-translated output can we process in a day? How fast can posteditors be trained to handle the material? and, eventually, How fast can we update our dictionary in order to be able to get better product each time?

It's important to see translation not as an end in itself but as a means to an end. You have to consider whether your program is compatible with word processors, what interfaces you develop, and how user-friendly they are—how much formatting and reformatting there is of the text after it has been translated.

I would also say that it's important to make a distinction between the requirements of a program and those of a posteditor. If you are a good editor, you can work with whatever output you have. But you can have the best program in the world and take a lot of time postediting if you're not computer literate and familiar with word processing. It's important to use macros—if the developer can come up with macros, so much the better. But it's not impossible to learn how to become a good posteditor. It only requires flexibility. Users have to be flexible in order to be able to work with MT; otherwise, they tend to blame the program, and it's unfair to developers, I would say.

I think the user and the developer should be a team. It’s like a concert: You’ve got your piano, you’ve got your music and you’ve got your musician. If you put them all together and it sounds good, then, it works. If you can’t get them together, then you have no concert. So, for us, what counts is what *we* can do with the output that the developers provide.

Doris Albisser*:* I just would like to underscore Margarita’s point about having macros for postediting. That significantly reduces postediting time, especially if you have to deal with system-specific errors that come up all the time and that might be solved with macro programs. So I think that point should be emphasized from the developer’s side.

General Discussion

• *(Magnusdottir)* I would just like to mention that the Swedish government funded a project to translate documents of the European Community which has cost them several million dollars, and the results are unusable. So human translation can be just as bad as machine translation.

• *(León)* I don’t know that that’s any solace! I’d like to make just a brief comment about formatting and how important it is to link into the publishing chain and to have the machine translation program preserve all of the format. We definitely *agree* with it, but it’s sort of a nightmare for the developers who would rather concentrate on the linguistic aspects. You just get an interface for one version of a word processing program or a desktop publishing program, and then the users go and upgrade to systems that offer a whole lot of new features, which then requires that the next people down the line also change their interface. But I guess that’s a fact of life.

• *(Bud Scott,* Logos Corporation) Going back to the question of users and dictionary coding, at Logos we originally restricted the users’ access to verbs. We believed that we could put in all the verbs that they might want. However, we later learned that that’s not the case. Even though we have probably 10,000 English verbs, English verbs are being created every day. Also, we didn’t want to reveal how we coded them—that’s proprietary. We don’t feel that way any more, either. But that was the original motivation. Also, if the verb was miscoded by the user, it would destroy the parse. That was another motivation. However, we do plan to offer verb updating capability in the near future.

• *(Muriel Vasconcellos,* PAHO) I have heard colleagues at SYSTRAN say they are concerned that users could “wreak havoc” with the system if they didn’t have a full command of the coding, and I think that that may also be a concern. There are too many rippling effects in the rest of system if the coding isn’t sufficiently mastered. Perhaps someone from SYSTRAN would like to comment on that.

• *(Jeanne Homer,* SYSTRAN) I've been with SYSTRAN for 20 years. The way SYSTRAN has solved this is to have a temporary holding spot that the customer alone uses, and it doesn’t interfere with the basic SYSTRAN dictionary that all the other customers use. This is for customers who want their own terminology different from our meanings or want to input words that aren’t in our dictionary yet. We call it a customer-specific dictionary.

• *(Elke Lange,* SYSTRAN) When the system is installed at the customer site and it’s a larger company, they have their own dictionary staff and they have the full training of dictionary coding that is usually given at our company. They can enter anything: nouns, verbs, expressions—they’re even trained to enter really complex expressions. Sometimes they need some more tutoring, though. That’s one group of people, though. That would not be possible for SYSTRAN Express or for a small user, because the training is fairly costly. It is also labor-intensive for the customer, but they have full control. What Jeanne was talking about is a customer-specific dictionary that medium-type users can use. Their translations are run at SYSTRAN on SYSTRAN machines, but they can have their own dictionary, in which they enter nouns, noun phrases, and some verbs. However, they cannot do complex expressions.

So it requires quite a bit of training to do all the really complex things and, yes, if somebody added an expression incorrectly, it could wreak havoc with the system.

• *(Albisser)* I think coding a dictionary is not all that complicated, providing you give your translator training. After all, the translator has got to know the grammar. I mean, if you can't expect that from your translator, he's definitely in the wrong job. In our company we give our translators two three-day trainings with an interval of two months in between, during which they go through their first trials and errors. Then they have their second training and do the real coding. It's all done with pull-down menus and you don't get inside the system.

If you have a user-friendly interface, with grammatical categories and an indication of how you should code your word frames—with examples and so on—it's basically just knowing the grammar.

• *(Elliott Macklovitch,* CWARC) It's quite clear, as Gudrun said, that if a system doesn't allow its users to update itself, then it has no business in the marketplace. But it seems to me that there's a correlation between the system's ambitions—that is, the depth of analysis that an MT system will attempt—and the complexity of the lexical entries. Notwithstanding pull-down menus and all kinds of default values, the more ambitious a system—that is, the deeper it attempts to analyze the input—the more information it's going to have to solicit from the users, and that may not always be obvious. Of course, if you have a simple word-for-word translation system, then dictionary updating is no problem at all.

• *(Vasconcellos)* I tend to be with Elliott on this. There are two kinds of entries that I can think of which would be extremely difficult to train a user to use. Each system has a sui generis way of encoding expressions. You don't find any two systems, at least that I'm aware of, that use the same rules for encoding context-sensitive expressions. So I think that is the sort of thing that might take a more intensive amount of training, as Elke mentioned for SYSTRAN.

The other kind is for knowledge bases. If you have a system that's sensitive to a knowledge base, you need to be aware of how the knowledge base is constructed and what node you're attaching your piece to. I'm not sure that those are the kinds of entries that should be available. If they are made available, they will require an extensive amount of training.

• *(León)* When we bring in the knowledge base idea, one is inclined to agree. We probably *don't* understand how the knowledge base is organized. Then what is the alternative? If the user cannot add a new word and fit it into the knowledge base, what is the alternative when the user *needs* to add a new word? They can't do it? They have to wait for the next release?

• *(Baena)* I would say that, in general, what a user would like to have as a possibility for updating a dictionary would not be that complex as conjunctions and prepositions. It's basically nouns, adjectives, verbs, different meanings for different parts of speech, and one word of the same part of speech with different meanings. That's more or less the kind of additions a user would like to be able to do.

• *(Macklovitch)* This is question for Larry Childs. Why do you think that you can't modify your rules by questions you ask your users? Is it because of the nature of software, because of the nature of the knowledge, or what?

• *(Childs)* Partially it's the nature of knowledge. I think that you probably *could* devise a system—a rule-based system—where you could get the users to change the rules. We talked about it being dangerous for a user to enter a preposition? I guarantee it is an order of magnitude more dangerous to start changing the rules of the parser, at least of the types of parsers we were using at WEIDNER and ALPS. It would be too dangerous to give the user that kind of power. He'd fix his one problem, but I guarantee he would cause problems down the road if you gave him that much power.

• *(Henry Thompson,* University of Edinburgh) We heard this morning about the need for regression testing. If you're going to let users change fundamental aspects of the system, you're going to have to give them the ability to do regression testing every time they make a change.

• *(Childs)* It's extremely time-consuming. That's what we did in our development labs at ALPS and WEIDNER. We did regression testing all the time. Every time we changed one little rule, we went through huge corpora to see what kinds of problems it caused us, and that is a bit much to ask the user to do.

• *(Jack Benoit,* MITRE) If it's the user's money he wants to spend to do the regression testing, by all means, allow him. In some cases, the users either have the intelligence or the knowledge to do this (or think they have)—either way, it seems like a good business decision to allow them to do that and let the user beware.

• *(Childs)* It will be interesting to see what the market actually does with these types of systems.

• *(Macklovitch)* As I understand it, TOVNA "learns" in the way you've described, Larry, but not by asking the user questions. I think, with regard to parsing at least, there are statistics—the system keeps statistics—on parsing frequencies and it learns from posteditors' modifications. So it's not by asking them questions they may not be equipped to answer, or by directly modifying the system's rules, but by repeatedly noting the same kind of postediting changes, the system is said to "learn."

That was one comment. I also wanted to ask you, Larry, about the chewing gum example of interaction under ALPS. I understand that, on a sentence-by-sentence basis, if you're not happy with a given translation, you can go into the dictionary, modify it, and that thereafter in the document you might get the translation you would prefer. Would ALPS "learn" from your answer on the chewing gum that "No, it's not gum that's chewing, but it's gum *for* chewing"? Would it learn it the first time, or would you have to keep chewing that question over and over again?

• *(Childs)* I'm afraid you had to keep chewing on that question over and over. We actually did a lot of studies and looked at stochastic parsing algorithms that would allow us to learn based on repeated answers to typical questions—in other words, if you answer a question a certain way a number of times then you get the idea that that's really what you want. But we found that that's a pretty poor predictor. When you hit a single instance of that type of phrase, it's very hard to predict from previous phrases how to translate that one. I mean, you can, but it's not very accurate.

• *(Shola Aboderin,* consultant to TOVNA) I can address the question of TOVNA's "learning." It *does* "learn." It's a matter of pattern-matching. It says, "Okay. This is the correction I've got regarding a certain modification that the translator made, so any other thing that looks similar, syntactically or otherwise, will be translated in the same way. There's a problem that you mentioned, that I agree with: it's time-consuming for certain structures. You find that you have to deal with a lot more structures than you expected, and keep teaching. But the idea, I think, is the right idea, because you find as you work with machine translation systems that the only way you can ever make them work is to have something that you can influence—that you can teach. And it does learn—you say, “This is how it is,” and that’s the way it learns.

As Elliott Macklovitch said, after you’ve got your patterns in, the system uses statistics to decide which is the more “like” pattern when you have a fairly similar situation. Right now, TOVNA is coming out with a new version. The idea is to shorten the time that it takes to learn. But the learning, I think, is an established fact.

• (*Eduard Hovy*, ISI) I’m curious about your conclusion that interaction is not a useful thing to do in helping the system. It seems to me that, if you're going to have to make a change and you have the choice to make the change inside and have the system do the rest of the sentence properly, as opposed to making the change at postediting and then maybe having to make bigger changes because the first problems gave rise to other things the system couldn’t do. I certainly would prefer to make a small change while the system asks me. Could you say a little bit more?

• *(Childs)* The reasons that I gave for interaction I firmly believe in. The conclusion that I came to was based on what happened in the real world when we tried to sell the system. It didn’t fly. Not enough people liked that to really make it successful.

• *(Hovy)* Did they say why?

• *(Childs)* Doris Albisser has tested the system fairly thoroughly. She might have something to say.

• (Albisser) We tested ALPS for several months. What we noticed was that after a while coffee consumption increased dramatically. Finally, we ended up with decaf . . . The problem is, it asks it the same question over and over again. And when it asks you a question, you have a choice between A, B, and C, if you’re lucky. Then you answer that question. Then you hear a rumble in the system, and then you get final result—which is a draft translation and which you definitely have to revise again. So in the end, you have to have two interactions. It definitely was not a time-saver. It doesn’t learn from a mistake. If you correct the mistake it does not memorize it and keep it for the next translation.

• *(Childs)* That’s one advantage that TOVNA has. Some of the changes are hard to predict: there are some patterns that vacillate frequently, even within a given text, and it’s hard to predict what the user is going to do. There are changes where it would have been an improvement on our system if we had learned from the first time, and I think that’s where TOVNA is a little better than ALPS was, because it does learn from that sort of thing.

• *(George Doddington,* DARPA) The topic of this session is very interesting to me. I guess I have to apologize, before I ask this question. I’m an engineer, and I tend to be sort of focused on facts and objective things. I was hoping to get more of a sense of a quantitative labor break-down in MT than I have so far gotten out of this session. What I have gotten is that the labor associated with updating dictionaries is the majority of the time, and that reformatting is 50% of the time, and that postediting is about 60% of the time. I was wondering if we could talk about this a little bit?

• *(León)* I think the problem is that we are considering the total time to be several different things. If we’re talking about the time it takes from when the translation comes in the door to the time it goes out the door, that's one thing. If we're talking about the life of the translation system, and building up the dictionaries, which is not actually part of getting out any one translation, we're talking about another, totally different piece of time. And then we're saying that different systems require more work at the front or more work at the end. There are some systems that require no work to get the text in. On the amount of time required for post-editing, there are general statistics that can be cited, at least from our organization. Do you want to give some, Muriel?

• *(Vasconcellos)* They vary a lot; that's the problem—it depends on the purpose of the translation. We've been doing MT since 1980—that's 12 years. If it's going to our Governing Bodies and we've got 32 countries sitting around voting on the text, we're very careful and we take a long time. If it's just for information, we can do a very quick job. It also depends to a great extent on the quality of the input text. Even when the subject matter is the same and the format is standardized, we find that some texts are a lot harder and slower to postedit than others.

The other parameter is the posteditors themselves. Some of them are quick at making decisions and some are not. Some of them are quite expert at using word processing. So we get a very wide spread even among posteditors. At each point along the way, we're going to get different results, depending on the circumstances. Overall, I think it's safe to say that MT postediting shows an improvement of 30% over human translation in terms of time spent. In more difficult situations it may be 20% or only 10%. In some cases, we can get as much as 100%—again, depending on all these variables. We simply can't predict and make a generalized statement about how long it's going to take us.

• *(León)* The dictionary work, of course, is much heavier when you first get your system. At this point, when we do a translation, we don't do any prior dictionary work. We run the translation, give it to the posteditor and wait for the posteditor to return the side-by-side output indicating the terms that need to be added or changed in the dictionary. In a 2,000-word translation, we may get six or seven suggestions. Three or four of them will involve changing translations to make them more context-sensitive. Not-found words are less than one per 1,000. For those of us who are more familiar with the capabilities of the dictionary, if we look more closely at the text we can find some more things we'd like to do to it.

• *(Baena)* One more thing about postediting time. Besides the skill of the posteditor, it would also depend on the type of change the posteditor is making. It often happens at the beginning that a posteditor makes changes out of preference instead of changes out of need. When you become familiar with the program and with its output, then you only make those changes that are necessary for the purpose of the translation.

• *(Coleman Harrison,* CompuServe) It seems to be generally agreed that the comprehensiveness of the dictionary is one of the keys to a successful MT system. I have three questions for system developers, or for this developer community as a whole. One is: What are the chances of hooking up MT systems to existing on-line bilingual dictionaries so as to get vast coverage, even if they don't have all your complex annotations? Might that not be a good place to start? Second: Which systems make it a practice to gather the users' contributions to the dictionaries into a central place and re-release them as an expanded dictionary? And then third (maybe the most paranoid): If I commit to an MT system today and spend the next 5 years typing tens of thousands of words into it, what happens when DARPA's project does succeed in revolutionizing MT and I decide to go to a different system? Do I have to type in all those words again? In other words, can I actually exchange vocabularies between MT system designs? If not, Why not?

• *(Childs)* On the question of hooking up with on-line bilingual dictionaries, that is happening—at least it was happening at ALPS. I don't know where it stands now, but companies like Collins and Elsevier. which publish a lot of technical dictionaries, have been negotiating with some of the commercial MT developers.

• *(León)* I think I understand what you have in mind: just accessing these resources on-line. If it isn't in the system dictionary, it would be looked up somewhere else. I can envision that being possible. However, you would still end up treating it syntactically the same way you would a not-found word. If all the information you've gotten is the part of speech, you would still be forced to assign default values to the word, but at least you would have a translation for it when it came out at the end.

• *(Magnusdottir)* Entering terms directly from a bilingual dictionary, is quite out of the question, but if you're referring to multilingual terminology banks, then that could be a different matter. But just open a dictionary and you will see the complexity. You cannot use a bilingual dictionary in a machine translation system because bilingual dictionaries are meant for humans and for human disambiguation. If you try to use it in MT, you will have a system that is totally garbled unless you sort it out very carefully.

• *(Henry Thompson)* Although it was a very reasonable question to ask about machine-readable dictionaries, the experience, even on the monolingual level, is depressing from this perspective. A number of people, ourselves included, have worked with the *Longmans Dictionary of Contemporary English* (LDOCE), and we're very grateful to Longmans for allowing us to do this. This is meant to be an analytically accessible monolingual English resource. The problem is that it is also a print dictionary and it's meant for the print dictionary market. It is meant to cover the language. Accordingly, each entry for a given word covers all the uses of that word. This means, for example, that there is an adjectival entry for the word "in," and it's right up there with all the other entries for the word "in," because we have the "in crowd"—Right? What that means is that every prepositional phrase in your input receives two analyses—one in which "in" is a preposition and the other in which it's an adjective. That's a tiny example of a systematic problem with "real" dictionaries: they cover *all* the meanings for *all* the words they have in them to the extent they can do that. As Gudrun just said, that problem is, literally, multiplied in a bilingual dictionary. To date, the basic language processing technology can't cope with the wealth of information that there is in a real dictionary.

On the re-use issue, I should just say that the European Commission is *desperate* to see the reusability of linguistic resources. This has been a banner the Commission has been waving for the last couple of years. Either in terms you might call "reclamation reuse"—Is there any way of quarrying usable information out of historical data banks of material for systems that are dead and gone now?—more or less along the lines that you mentioned. The other thing is "designed for re-use."

• *(León)* We all know it would be a good thing. DARPA is also funding a lexical consortium here in the United States. But pooling lexical resources is going to be difficult. Everybody thinks that their format is the best and that their information is the best, so I would guess that the people who are going to get the most use out of these discarded dictionaries are the ones who are starting with nothing. At least they can look around and see something to get started with. I don't think that you're going to find someone who already has a big dictionary or lexical data base switching to somebody else's. But 5 years down the line, if you decide to switch systems, one of the requirements should be that the developer of that system convert your dictionary. Thank you all.