Chapter 5

GENERAL PRINCIPLES OF USER-ORIENTED EVALUATION

Margaret King
ISSCO – School of Translation and Interpretation, ETI
University of Geneva, Switzerland
Maghi.King@gmail.com

Abstract This chapter is concerned with a particular perspective on the problem of evaluation design. User-oriented evaluation takes as primary some user or set of users who need to accomplish some task, and sets out to discover through evaluation whether a given software system will help them to do so effectively, productively, safely, and with a sense of satisfaction. (Note that, following ISO, user here is used in a very wide sense and encompasses much more than what has conventionally been called end-user.) There is a clear tension between taking specific user needs as primary and seeking common principles for the evaluation of particular software applications. The chapter suggests that this tension may be resolved by using an ISO standard for the evaluation of software as an appropriate level of generalization (ISO 9126). Quality models reflecting the characteristics of specific software applications (machine translation, document retrieval, information extraction systems, etc.) are then built on the skeleton set out in the ISO standard. Particular user needs are taken into account by picking out those parts of the appropriate quality model which reflect the needs, where necessary imposing a relative order of importance on the parts picked out. Execution of the evaluation then concentrates on the parts of the quality model chosen as pertinent to the user and the context of work. The focus of the chapter is on general design questions rather than on the strengths and weaknesses of specific metrics. However, there is some discussion of what it means for a metric to be valid and reliable, and of the difficulty of finding good metrics for those cases where system performance and human performance in interaction with the system are inextricably linked. A suggestion is made that it might be possible to automate an important part of the process of evaluation design, and an attempt to do this for the case of machine translation evaluations is briefly sketched.

Keywords User-oriented evaluation; Quality models; ISO.

L. Dybkjær et al. (eds.), Evaluation of Text and Speech Systems, 125–161.

1 A Historical Note

I could not claim personal authorship of any of the ideas put forward in this chapter: they are the fruit of an effort started over 10 years ago through the launch by the European Commission of a series of initiatives whose main aim was to stimulate the production of linguistic resources for the European languages. This was to be achieved by creating standards, so that resources could be shared. The initiatives were the two EAGLES[1] initiatives (1993–1996 and 1997–1999), which were followed by the ISLE[2] project (1999–2002), a joint project of the European Union and the National Science Foundation of the United States. Swiss participation in all three initiatives was directly funded by the Swiss Federal Office for Education and Science.

EAGLES took the form of a number of working groups, who essentially organized their own work. Some of the working groups operated in areas which were ripe for standardization, such as the collection of speech data or written corpus collection: others were asked to do preliminary investigations, working towards pre-normative guidelines in a specific area. One of these latter was the working group on evaluation, whose remit was to find a general methodological framework for the evaluation of human language technology products and systems. The first EAGLES initiative set out a general framework which recurs throughout this chapter (EAGLES Evaluation Working Group, 1996). The second EAGLES initiative organized a series of workshops through which knowledge of the basic framework was disseminated and further refinement of it took place. The EAGLES work concentrated on relatively simple language technology products such as spelling checkers, grammar checkers, and translation memory systems as test beds for the evaluation methodology. The ISLE project moved on to more complex systems, concentrating on the construction of an evaluation framework for machine translation systems. This fairly substantial, but still incomplete, example of an evaluation framework can be found at the following URL: http://www.issco.unige.ch/femti.

Work on the machine translation framework (baptized FEMTI) is being carried on through a project of the Swiss National Science Foundation which began in early 2005.

The ISLE project continued the tradition of organizing workshops where intermediate results could be discussed and new ideas put forward. Several of these workshops were "hands-on" workshops where the participants worked directly on specific problems of evaluation or on constructing parts of the framework. Some of the preparatory documents for various workshops can be found at http:/www.issco.unige.ch/projects/isle/. Over the years, well over 100 people must have been actively involved in EAGLES or in ISLE work, and since most of the effort was collaborative, it would be almost impossible to say who first suggested some new idea. It is for this reason that the present

author, who was chair of the evaluation working group throughout its lifetime, claims only to be the reporter of common work and not its originator.

Another historical and intellectual debt is very important in the work reported here. ISO/IEC published in 1991 the first of its standards concerning the evaluation of software (ISO-9126/91). The normative part of this document set out a quality model for the evaluation of software. It also contained pre-normative guidelines for how the process of evaluation should be defined. The standard was brought to the attention of the first EAGLES evaluation working group by Kirsten Falkedal, one of its members, and subsequently became a primary inspiration for EAGLES work. The link with ISO work on evaluation was consolidated during the second EAGLES initiative, with the technical editor of the standard participating directly in an EAGLES workshop and contributing to the draft final report. The evaluation framework for machine translation systems produced as part of the ISLE project is structured around the quality model set out in the ISO 9126 standard. Recently, ISO/IEC has published two new series of standards on software evaluation (see bibliography for full ISO references). Defining the quality model and metrics related to it has now been separated out from defining the process of evaluation, giving rise to a revised 9126 series (quality model and metrics) and the new 14598 series (evaluation process and management).

Both the ISO work itself and the EAGLES work based on it were influenced by work on quality assurance in the software industry: one assumption here is that the sort of thinking underlying the assessment carried out in the context of producing a piece of software carries over to evaluation in a wider context. Work on evaluation of software in the context of technology acquisition around the time of the EAGLES projects also brings out the importance of how the software will be used and in what context, thus falling into the user-oriented philosophy. (See, e.g., Brown and Wallnau, 1996). This general intellectual current is reflected too in the similarity between the general philosophy of user-oriented evaluation and recent developments in software design, as typified by the formulation and deployment of use cases in drawing up software specifications in languages like the Unified Modeling Language (UML; see, e.g., Booch et al., 1999).

Much of the rest of this chapter is intended to be directly based on the ISO standards, although of course only the author is responsible for any misrepresentation of them.

In the specific context of this chapter, I would like to thank the two anonymous reviewers, whose perceptive and helpful remarks have, I hope, contributed to the improvement of the first draft.

Finally, I want to acknowledge a personal debt to two of my ISSCO/TIM colleagues, Andrei Popescu-Belis and Nancy Underwood. Over the past

several years we have spent much time in discussion of evaluation questions: they have been generous with their time and with their insights.

2 What is User-Oriented Evaluation?

Many academic evaluation exercises concentrate on a software system taken in isolation, looking primarily at what it is supposed to do, and ignoring the context in which it will do it. User-oriented evaluation adopts a radically different perspective, taking as primary a user or set of users who need to accomplish some task and asking whether the system will help them to do so effectively, productively, safely, and with a sense of satisfaction. This implies looking at a large and complex set of factors which will contribute to whether, in the end, a decision to acquire and deploy the system will seem to have been a good decision. Frequently, the factors involved are not independent of one another, either conceptually or in the ways that each factor may contribute to an overall judgement. Thus, an evaluation designer working in the user-oriented perspective may often find himself/herself saying something like "well, it would be nice to have x, but if not, y might compensate for the lack, but whatever happens with x and y we must have z".

This implies that right from the start evaluation cannot be seen as a search for a single magic metric that will tell all that needs to be told about the system or systems being considered and which, when used alone, will allow direct comparison of competing systems.

Because of this chapter's focus on the philosophy and design of user-oriented evaluations there is very little direct discussion of particular metrics. Some metrics will be used as illustrations of specific points and will be briefly described, but no metric will get the thoroughgoing discussion of its strengths and weaknesses that a proper account focusing on the definition and choice of metrics would deserve. In mitigation of this weakness, most metrics are really only of direct interest to someone involved in the process of concrete evaluation design for a particular application of language technology – summarization, information retrieval, term extraction, or any other of the by now very many applications available. Such a person needs a detailed analysis and critical discussion of the specialized metrics applicable in his/her area of interest, and such a discussion can best and most easily be found in the technical literature, where interest in evaluation and in suitable metrics has been constantly expanding in the past few years.

Making a conscious decision not to treat specific metrics in any detail should not be interpreted as dismissing the importance of metrics: indeed, it will be argued later that it is the choice of metrics which determines the operational content of any specific evaluation, and a sister paper to this one (King, 2005) has much to say about that. But what concerns us here is all that has

to happen before the evaluation designer can even begin to think about what metrics he/she will choose. This preliminary – very great – labour will be set out in the form of a number of principles underlying evaluation design in the user-oriented perspective.

3 A First Principle: Quality is Decided by Users

In the vast majority of cases, it is impossible to say in absolute terms whether something of its kind is or is not good. This is true of objects, processes, study programmes – of almost anything we can think of. In the case of software this otherwise rather sweeping statement can be justified fairly easily. Software is not created for its aesthetic value: it is meant to help in achieving some task, and its value is to be judged precisely in terms of whether it does so. There is thus always a user of the software, someone or some process who needs to get something done, and who makes use of the software as a means to that end.

Users can come in all shapes and sizes: they are not necessarily what are conventionally thought of as "end-users". Drawing up an exhaustive list of people who might be users in some given situation is not a practical proposition, so let me illustrate this with a few examples. Quite obviously, as I sit typing this into a text processor, I am a user, both of the text processing system itself and of the whole platform in which it is embedded, and, in this case, I am also an end-user. But imagine now the university computer committee who decides what hardware to buy and what software to put on it. They too are users in the sense of this section. They use the computer facilities they have decided to purchase by putting them at the disposal of a community of end-users, and, just as I may be more or less satisfied with what I am using, they may be more or less satisfied with the provision they have made.

Other users may not be using a commercial product at all. If a research worker is developing a research prototype, he/she is a user of that prototype and of the modules that go to make it up: as such, he/she will be more or less satisfied with the prototype or its modules. It could even be plausibly argued that if one of the modules of his/her research prototype makes use of input provided by another module or interacts with it in some other way, the module of the prototype is itself a user. It cannot of course feel satisfaction, but an analogue to satisfaction can be formulated in terms of whether it gets the right input or the appropriate interaction.

An early ISO discussion illustrated the variety of potential users rather graphically by listing the users of an aeroplane, who include the cockpit crew who fly it, the passengers who travel in it, the cabin crew who look after the passengers, the company to which the aeroplane belongs, and even the control tower staff who give instructions for landing and take off. All of these have very different requirements which the aeroplane should fulfil.

Those who find this ISO-influenced extension of the sense of "user" rather counter-intuitive might feel more comfortable with a word like "stakeholder", as one of the reviewers suggests: indeed, the FEMTI evaluation framework described later quite consciously uses "stakeholder" in order to avoid some possible confusions between stakeholders in a general sense and users, who are seen as a subset of stakeholders. Whatever the word, the essential point is that the entities whose needs are to be satisfied or whose concerns have to be taken into consideration when designing an evaluation may be many and various: the evaluation designer should be very clear about whose needs and concerns are reflected in the evaluation design.

A user or stakeholder then is someone or something that has a set of needs: quality is to be judged in terms of whether or not those needs are satisfied. The goal of evaluation is to gather the data which will be analysed in order to provide a sound basis for that judgement. It follows from this that the first task of an evaluator is to find out what the needs of the particular user or users implied in the particular evaluation are. The second task is to formulate criteria reflecting those needs. On that foundation, the evaluator can decide what metrics, when applied correctly, will measure system performance with respect to the chosen criteria and work out the most reliable way of applying the metrics. The results of their application, when analysed and presented informatively and perspicaciously, will allow final judgement to be made.

4 A Second Principle: Users do not Have the Same Needs

4.1 Different Tasks, Different Needs

It is self-evident that a user may need different pieces of software in order to fulfil different tasks: a spelling checker cannot be expected to solve polynomial equations, or a search engine to play music. But it is slightly less obvious that different users may have different requirements even of the same piece of software. Machine translation systems can be used to illustrate this idea.

Let us imagine that I am thinking of hosting a future Olympic Games, and want to find out from the press what Greeks felt about Greece having hosted the games in 2004. Essentially, I need to comb the Greek newspapers looking for articles which report on popular reaction. I do not speak Greek, but I do have a limited budget to help in my search. I probably do not want to spend all my budget on hiring Greek speakers to check as many papers as they can before the money runs out; I would be much better off finding some cheap way to identify those articles particularly relevant to my search and using my budget to have their contents summarized. In this situation, a machine translation system may help: it can be used to produce a rough translation from which pertinent articles can be identified. The translation produced by the software

has to be only good enough to allow identification of interesting articles. In other words, the most important needs here are for speed (there are a lot of newspapers) and economy (the budget is not enormous) rather than for high-quality translation; in fact, measuring translation quality in this case can be reduced to discovering whether or not the machine translation output does indeed permit a satisfactorily large percentage of relevant articles to be identified as such.

Contrast this with the situation where my proposal has been accepted and I must now host the games. Athletes from all over the world will come to compete, and they will all need to be provided with information in a language they can understand, ideally their own. It may be quite difficult to find human translators with the necessary language combinations to produce this information, so I may once again have recourse to machine translation. But in this context, the needs have changed dramatically. The translation must be good enough to avoid problems of misunderstanding or the risk of giving offence, speed is less important given that there have been several years in which to plan the organization, and even, in all likelihood, economy is less important since the amount of work to be done is decided by the languages in which information will be provided, not by how many newspaper articles can be treated before the budget runs out.[3]

This is of course rather an extreme example, but the same reasoning can be applied to much more modest situations and much less complex software.[4] Translation students when working on a translation tend to get very indignant about spelling checkers which do not pick up as unidentified words slang and borrowings from other languages. When they are writing letters to their friends, a spelling checker that did pick up those same barbarisms would probably prove very exasperating.

On top of all this, even when the task remains unchanged, different users may have different needs simply because they are different users, with different backgrounds, different expertise, and different expectations.

In summary, the set of needs pertinent to an evaluation is decided by a combination of the users concerned and of the task or tasks they want to accomplish.

4.2 Different Evaluation Purposes, Different Needs

Furthermore, evaluations themselves are meant to respond to a set of needs, and those needs encompass more than just finding out whether a piece of software does or does not do a specified set of tasks. In early EAGLES work, we distinguished different purposes behind carrying out an evaluation, each of which imposes its own requirements on the evaluation design.

First, there is the kind of evaluation familiar to any programmer or system designer: the main focus of the evaluation is on discovering why the software behaves as it does and, in particular, what causes things do go wrong. We call this *diagnostic evaluation*. An example in practice comes from rule-based parsing systems. In the early 1990s a lot of effort went into the creation of test suites, sets of artificially created inputs to a parsing system where the aim was for each input to test the system's behaviour with respect to a single well-defined linguistic phenomenon (King and Falkedal, 1990; Lehmann et al., 1996). Thus, by looking at the output from running the test suite, the system designer could see, for example, whether simple noun groups were being properly treated, or whether sentences containing passives were causing problems. In the particular case of parsing systems, knowledge of what inputs were not properly dealt with could point directly to what linguistic rules were not functioning properly.

Test data can take many forms, and some of those forms also serve as the basis of one kind (there are others) of *comparative evaluation*. In this scenario, a collection of data which has been agreed upon as appropriate for the system being evaluated is typically divided into two parts. One part, the training data, is used to guide the development of the systems to be evaluated. The other part of the data serves as test data: the same inputs are given to a number of different systems, and their ability to treat the inputs appropriately examined. Both inputs and expected outputs are specified as part of the test; by definition the specified outputs are the "right" answers given the specified inputs: they constitute a "gold standard" against which any particular set of input/output pairs produced in practice may be assessed. When we discuss metrics, we shall return to the use of gold standards of this kind in evaluation.

This is the basic principle behind the vast majority of the evaluation campaigns organized by DARPA/ARPA and others, where system designers and constructors compete to produce the "best" results from a common set of data (see the bibliography for references to the MUC, TREC, ATIS, and MT campaigns, for example). The primary aim of such a campaign is usually stated to be the advancement of core technology in a particular area, coupled with the creation of a research and development community working in that same area.

It goes without saying that diagnostic evaluation based on test data and comparative evaluation based on test data are two very different things. In the case of diagnostic evaluation, using test data to probe for where a system breaks down is meant to help in identifying a deficiency in its working. In the case of comparative evaluation as defined here, test data serve as a way of quantifying to what extent a system succeeds in producing the results it has been designed to produce – they tells us nothing of any other virtues or weaknesses. Indeed, using test data in this way has sometimes been stigmatized as producing the illusion that apples can be usefully compared to pears.

Within the human language technology field, test suites have also been frequently used to measure progress in the development of a system: an increase in the number of test items successfully dealt with provides a measure of how much the system has progressed towards the ultimate goal of being able to deal with every item in the test suite. Another way of carrying out *progress evaluation* of this kind is to collect together a corpus which is held to be representative of the text or language the system should be able to deal with. The fact that corpora are by definition texts which occur naturally has advantages in terms of economy and also produces the comfortable glow that comes from dealing with the real world instead of with an artificial academic construct. On the down side, a corpus used for testing is only informative if it is in fact representative of the real world which the system will be expected to deal with: ensuring representativity raises issues that are sometimes difficult to resolve. Furthermore, the use of a corpus as test material ties the evaluation (and its results) to a specific "real world": there can be no guarantee that the quality of results obtained in the context of use reflected by the choice of corpus will carry over to other contexts of use.

The final kind of evaluation distinguished by the early EAGLES group was called *adequacy evaluation*: the term was meant to capture a situation somewhat parallel to that of a consumer contemplating a major purchase. The consumer knows what is wanted in a proposed new washing machine or new car; a specific product is examined with a view to finding out whether it offers what the consumer wants. The parallel with software evaluation is not difficult. But it is perhaps worth pointing out that once again, different consumers, different users in the jargon of this chapter, may have very different views on what is wanted. Machine translation can again serve as a concrete illustration. Imagine a translation service contemplating the purchase of machine translation software. It may well be that the manager of the service wants a machine translation system which will deal with the language pairs where there is trouble recruiting good translators, whilst the translators already in the service want a system which will relieve them of some of the burden of translating the 500-page activity report which appears once every quarter and where over half the text remains unchanged from one edition to the next. And, of course, both manager and translators may be quite wrong in thinking that the answer to their problem is a machine translation system: an evaluation taking into account the whole work context may well reveal other and more productive options.

This last possibility brings us to another point about the variability of user needs, or rather, about the user's perception of his/her needs. The process of eliciting needs and making them explicit in the form of a set of user quality requirements may well contribute to a realization that needs should be refined, modified, or perhaps changed all together. There is nothing surprising about this: in fact discovering that one has misjudged or misstated a set of needs

is a fairly common occurrence of daily life. (Who has not bought the wrong garment or ordered an ill-judged meal?) Elicitation and definition of needs is not, except in the simplest of cases, a linear process. This in itself constitutes a very strong argument for investing time and energy on drawing up explicit requirements based on acknowledged needs before expending the energy required to define ways of discovering whether a particular system can meet those needs.

The types of evaluation discussed in this section are not meant to be seen as impermeable categories. Diagnostic evaluation may be part of progress evaluation, comparative evaluation may be of two successive versions of a system and therefore also be progress evaluation, and it would be quite possible to see all the other types of evaluation as special cases of adequacy evaluation. The point in making the distinctions is twofold: first, to emphasize yet again that different contexts may impose very different requirements on both the software itself and on its evaluation; and second, to stress that defining the purpose of the evaluation is an essential preliminary to designing it.

On a more practical and even mundane level, it is also extremely important that all those involved in an evaluation share a common perception of its purpose. A stupid and rather costly mistake from my own experience will help to illustrate this somewhat obvious but too often neglected point. We had undertaken to build a system that would translate a limited set of sentences from German into French. This was in the days long before easy and convenient treatment of character sets, so our proof-of-concept demonstrator, whose evaluation was to determine whether we would continue with the project or not, made use of codes to replace the accented and special characters of the two languages. The evaluation was a disaster. We believed that its purpose was to show that we could in fact translate all the different linguistic phenomena contained in the agreed set of sentences, so using codes for French and German characters was irrelevant. The representative of the funding source thought that what had to be shown was that we could translate from German into French – and that we clearly could not do since we could not even deal with the appropriate character sets. Of course, anyone who knew about computing would simply say that our interlocutor did not understand the very minor importance of the character codes – but we still lost the contract. And, of course, this point carries through on a much larger scale once general management questions are an issue. To go back to our fictitious translation service, if the manager thinks the evaluation is being carried out in order to find out whether the service can offer new language pairs, but the translators think that the evaluation is aimed at finding out whether they can be replaced by a computer system, the potential for disaster is inescapable.

5 A Third Principle: Quality can be Characterized

5.1 Quality Models

Everything so far has been rather distressingly bottom-up. We have insisted on the idea that whilst quality can only be defined in terms of users, users have very different quality requirements, and we have aggravated the potential problems posed by that claim by adding that different kinds of evaluations designed with different aims in mind also affect the quality requirements which form the backbone of the evaluation design. The obvious conclusion is that every evaluation is necessarily a one-off exercise, carried out for a particular client in view of a particular set of user needs. If this were true, evaluation would also be a very costly exercise, since little if anything could be shared across evaluations.

The ISO 9126 standard constitutes a direct antidote to the slough of despond created by the idea of having to start afresh each time. The basic idea is that if we operate at a sufficiently high level of generality, there is a small set of characteristics of software which are likely to be pertinent to a judgement of quality in almost every case: listing these characteristics, breaking them down into sub-characteristics, and providing definitions of each item will provide the designer of an evaluation a way into examining the needs of particular (sets of) users and expressing their quality requirements in terms of the characteristics which make up part of the general quality model.

There is not enough space here to go into all the detail of the 9126 standard, but it is probably useful to give a brief summary, and some example definitions which are taken from ISO/IEC 9126 series, part 1, published in 2001.[5] The reader is urged to consult the standards directly for more detail and for further discussion.

ISO 9126 proposes six main quality characteristics of software. The first of these is functionality. Functionality is essentially concerned with what the software does, rather than how it does it. It is broken down into five sub-characteristics. The sub-characteristics of functionality are suitability, accuracy, interoperability, security, and compliance. We shall leave interoperability and security as intuitive notions, which will, of course, have to be fleshed out with concrete and detailed definitions in the context of any particular evaluation.

The distinction between suitability and accuracy, however, needs a little more commentary. Suitability is defined as "the capability of the software to provide an appropriate set of functions for specified tasks and user objectives", and accuracy as "the capability of the software product to provide the right or agreed results or effects": in other words, accuracy is based on whether the software conforms to its specifications. It is almost redundant to say that

what results should be produced is a key component of the specifications. If the software does indeed produce the results its specifications say it should, by this definition the software scores well on accuracy. But high accuracy does not necessarily mean that the results produced are in fact useful to a particular user with a specific task to accomplish. In the worst case, the software designer has simply got wrong what might be helpful – market failures provide empirical verification of the existence of this possibility.

A concrete example may help in grasping the distinction. A major need in the translation world is for terminology extraction tools. Modern technology moves at such a rate that vast amounts of new terminology appear all the time, and even specialized technical translators cannot be expected to keep up with the development of terminology in their fields. At the same time, it is rare that a new term appears only once in a single document; most frequently, once the term has been coined it will be used almost simultaneously in a number of documents and may even find its way into a term bank before our hypothetical translator comes across it as a new term. A software tool which extracted from a text all the terms it contains would give the translator a head start in preparing the translation. The list of extracted terms could be compared to the contents of the term banks to which the translator has access, thereby isolating remaining problem cases. If the software could go one step further and not only isolate the new term but also identify its probable translation in any texts that had already been translated, the usefulness of the tool would be increased even further. There are softwares on the market which claim to be of assistance in identifying potential terms. The most simple of these operate on the assumption that a term is a string of words that will appear more than once in a document. They therefore list all the sequences of words which appear more than once in the text. (Most frequently, the user may decide the length of the sequence of words – for example, two words or more – and on a lower bound for how many times the sequence must appear in the text, e.g., twice or more.) Given these specifications for how to identify candidate terms, any piece of software that produces from a text a list of all and only those word sequences matching the parameters is accurate in the sense described here – it conforms to its specifications. But it takes very little reflection to see that the results will be pretty well useless to any translator or terminologist. Sequences such as "all the", "any piece", or "given that" will appear far too frequently, and the time taken to sift any potential real terminology from all the dross of useless suggestions will be so great that no user will contemplate the investment. To make matters worse, since no morphological analysis is carried out, "Internet technology" and "Internet technologies" may not be picked up as possible variants on a single term. And worse again, unless one or other of them occurs more than once, with the example parameters given, neither will be picked up at all. To couch this in the jargon of the well-known precision and recall

metrics, there is both far too much noise and at least a strong risk of silence. In other words, the results, whilst totally accurate, are not suitable.[6] In user-oriented evaluation, suitability is likely to count for rather a lot more than conformity to specifications.

There are cases nonetheless where a failure to distinguish between accuracy and suitability may not be very important, simply because the nature of the software in question is such that the two do more or less coincide. A case in point is dictation software. Accuracy, in this case, is defined in terms of being able to transcribe correctly the words spoken by the user. For users who want to be able to dictate their text, it is a reasonable assumption that the fewer mistakes in transcription the software makes, the more suitable will they find the results. (A user may have other reasons of course to dislike the software, but in terms of this aspect of its functionality, all should be well.)

At the other extreme, software products are becoming available where the link between accuracy and suitability is far more problematic. A first example comes from search engines. Most of us have experienced the awful moment of being presented with a million or more results in response to a query. The search engine is perfectly accurate, however, according to its own specifications. The results are just not suitable once our information needs are taken into account. And, of course, the search engine cannot be blamed: the software has not functioned badly; we have failed to formulate our query in a satisfactory way. Less familiar examples are data-and text-mining softwares. They too deal with a very large mass of data, trying to find connections and associations that could not be found by any human. Such tools may well come up with totally accurate but completely uninteresting and therefore unsuitable insights, like an association between pregnant women and having children.

The problem is complicated by two further factors. The first is the quality of the data: if the data are poor, the software cannot be blamed for coming up with conclusions which are not very useful. The second is the competence and flair of the users. Typically, with these kinds of software, there is interaction between an expert user and the software in searching for a useful result: on the basis of a preliminary set of results the user will refine his search for information or instruct the software to ignore certain characteristics of the data. Some users are better at this than others. Once again, a piece of software cannot be blamed if a particular user cannot formulate and direct an information request appropriately. One of the current challenges in evaluation theory is to come up with a sound methodology for user-oriented evaluation of softwares where problems of this kind are inherent in the nature of the software (see King and Underwood, 2004 for more discussion).

A lot of time has been spent on only two sub-characteristics of functionality. Fortunately, the remaining quality characteristics and their sub-characteristics are intuitively more accessible, especially in these days of PCs and portable

computers when many people serve as their own computer administrator. For this reason, Table 1 produces a summary of the six characteristics and their sub-characteristics. The right hand column gives a brief gloss of the definitions given in the ISO standard, leaving it to the reader to flesh out appropriate definitions for the terms used.[7]

Table 1. Summary of characteristics and sub-characteristics.

Quality characteristic	Sub-characteristics	Comments
1. Functionality		**Providing functions to meet needs**
	a. Suitability	Provision of an appropriate set of functions for specified tasks and user objectives
	b. Accuracy	Provision of the right or agreed on results
	c. Interoperability	Interaction with other specified systems
	d. Security	Protection of information and data
	e. Compliance	Adhesion to appropriate standards etc.
2. Reliability		**Maintaining performance**
	a. Maturity	Avoid failure as a result of faults in the software
	b. Fault tolerance	Maintain performance in spite of faults
	c. Recoverability	Re-establish performance and recover data in case of failure
3. Usability		**How easily can the user understand, learn, operate, and control the system? Is it attractive to users?**
	a. Understandability	Can the user understand whether the software is suitable, how it can be used for particular tasks, and what the conditions are for using it?
	b. Learnability	Can the user learn to use it?
	c. Operability	Can the user operate and control it?
	d. Attractiveness	Does the user find it attractive?
	e. Compliance	Adhesion to appropriate standards etc.
4. Efficiency		**Appropriate performance relative to resources used**
	a. Time behaviour	Response, processing, throughput
	b. Resource utilization	Amounts and types of resources (excluding human resources, which are part of quality in use)
	c. Compliance	Adhesion to appropriate standards etc.

5. Maintainability		**Correcting, improving, or adapting the software**
	a. Analysability	Can faults be diagnosed?
	b. Changeability	Can specified modifications be implemented (by a programmer or by the end-user or both)?
	c. Stability	Avoidance of unexpected side effects
	d. Testability	Can modified software be validated?
	e. Compliance	Adhesion to appropriate standards etc.
6. Portability		**Transferring the software from one environment to another**
	a. Adaptability	Adaptation to different specified environments
	b. Installability	Installation in a specified environment
	c. Coexistence	Coexistence with other independent software
	d. Replaceability	For example, is up-grading easy?
	e. Compliance	Adhesion to appropriate standards etc.

The glosses given here are meant only as mnemonics for the much fuller definitions of the standard. However, even in this very abbreviated (although, hopefully, not deformed) version it is immediately clear that the definitions are at a very high level of generality – they are, after all, meant to apply to any kind of software. But this means that they have to be made much more concrete in order to design an adequate evaluation for any particular type of software. We return to this issue in Section 5.3

Before moving on, the **Usability** quality characteristic deserves some commentary, if only because talk of user-oriented evaluation is so often misinterpreted as meaning evaluating usability. Usability, as shown in the table, breaks down into **understandability**, **learnability**, **operability**, **attractiveness**, and, as always, **compliance**. The notes given in the ISO standard on the various definitions make a number of interesting points. First, they make it clear that quality characteristics are interdependent. For example, some aspects of functionality, reliability, and efficiency will clearly affect usability, but are deliberately excluded from mention under usability in the interests of keeping the quality model tidy and well structured. Similarly, aspects of suitability (from functionality), changeability (from maintainability), adaptability (from portability), and installability (from portability), may affect the sub-characteristic operability found under usability. Trying to capture the intricate potential relationships between sub-characteristics would be very difficult, and especially so since often they are only potential rather than

necessarily actual: when a specific evaluation is being designed, a potential relationship between two sub-characteristics may turn out not to exist in the particular context. Avoiding unnecessary complications in the interests of mental hygiene may impose a certain artificiality in the definition of a quality model.

This brings us to a central and critical point, which has already been hinted at: the real meaning of any quality characteristic or of its sub-characteristics is operational, and is given by the metrics used to measure system performance with respect to that characteristic. Furthermore, it is the decomposition of the top level characteristics and sub-characteristics in order to arrive at measurable attributes which allows the general quality model to be specialized for specific software applications. This will become clearer when we discuss the formal structure of a quality model in Section 5.2.

Second, the notes emphasize that usability issues affect all the different kinds of users: "Users may include operators, and users and indirect users who are under the influence of or dependant on the use of the software. Usability should address all of the different user environments that the software may affect, which may include preparation for usage and evaluation of results." This again emphasizes the great variety both of users and of the environments in which they work, stressing that there may well be users other than end-users whose needs have to be taken into account when designing an evaluation.

All of the quality characteristics making up the quality model contribute ultimately to what the ISO standard calls **quality in use**. This is the quality of a piece of software as it is perceived by an actual user, in an actual work situation trying to accomplish an actual task. ISO/IEC 9126-1/01 defines it as "the capability of the software product to enable specified users to achieve specified goals with effectiveness, productivity, safety and satisfaction in specified contexts of use". Quality in use can only really be evaluated in situ, although much effort is invested by manufacturers of commercial software into trying to control the eventual quality in use of a product before it is released on the market, and a central tenet of this chapter is that by careful examination of users and of the tasks they will perform it is possible to evaluate a piece of software in such a way as to be able to predict its potential quality in use.

Thus a basic assumption underlying both the ISO 9126 standard and the kind of evaluation discussed in this chapter is the existence of a sort of quality chain: good specifications will contribute directly to production of good code (internal quality), good code will contribute directly to good system performance in terms of the quality characteristics (external quality), and good system performance will contribute directly to good quality in use. The particular slant on this assumption in EAGLES and ISLE work is that by looking at a combination of user needs and system performance in terms of the quality characteristics,

we can construct specialized quality models and thus, on the basis of an evaluation of external quality, go a long way towards predicting quality in use for the specific user.

5.2 Formalizing the Quality Model

The ISO quality model sketched briefly above is informal, in the sense that anything written in a natural language is informal: it names quality characteristics and sub-characteristics, and provides definitions in English for them. Both names and definitions are therefore open to different interpretations by different readers; this is not a fault in the standard, but a problem inherent in the use of natural language.

A major aim of the EAGLES work was to impose a more formal structure on the quality model with the double aim of facilitating clear thinking about quality models for particular types of software and of defining a structure which could serve as the basis for computer implementations of evaluation schemes based on the quality model principle.

Within EAGLES, a quality model was defined to be a hierarchical structure. The top-level nodes in the structure are the quality characteristics themselves. The sub-characteristics are daughter nodes of the top-level characteristics. The ISO definition legislates for only these two levels. The EAGLES version, however, allows sub-characteristics to be broken down in their turn, with the hierarchy descending to whatever level is needed to bottom out into attributes to which at least one metric can be associated. In other words, the leaves of the structure must contain attributes which are measurable.

Each node in the hierarchy is then defined to be a feature/value pair of the sort familiar from computational linguistics. The name of the quality characteristic or sub-characteristic is the name of the feature. When an evaluation is executed, the value of the feature is obtained by propagating values upwards from the leaves of the tree. The values on the leaves are obtained by applying the metric associated with that leaf. (For simplicity, we shall imagine that there is only one metric associated with each level: it is a fairly simple step to generalize to the case where more than one metric is associated). Values on higher nodes are obtained by combining the values from the next hierarchical level down according to a combining function which is part of specifying the particular evaluation.[8]

Perhaps the most interesting feature of this formalization is that the quality model has now gained a precise semantics. Where just saying that part of functionality is suitability does not say much and does not say it unambiguously, once suitability is tied directly or indirectly through a branch of the hierarchical structure to a metric or metrics, it acquires a clear and unambiguous interpretation: its meaning is given by how its value is to be

obtained from the lower nodes in the quality model. This is what was meant by saying that the semantics of an instantiated quality model was operational, and determined ultimately by the choice of metrics. To this we should now add "and by how the values obtained through applying those metrics are combined to give values for upper nodes in the structure". The quality model can still of course be ill-conceived, but we are now much more likely to discover that it is, and discussion about its correctness or incorrectness is anchored in empirical observation.

5.3 From the General to the Particular: Specializing a Quality Model

The quality model defined by the ISO standard is situated at a very generic level. In order to produce from it a model useful for a particular evaluation we need to make it more concrete. This involves first specializing the model to take into account the particular kind of software to be evaluated and secondly making it more concrete by relating the model to the specific needs of a user.

If we look at the names and definitions of the ISO quality characteristics, functionality leaps out as the characteristic needing further specification in terms of the particular type of software to be evaluated. As the reader will remember, its sub-characteristics are suitability, accuracy, interoperability, security, and compliance. Of these, accuracy seems most closely to reflect the nature of the software to be evaluated, and it is therefore perhaps no accident that the majority of evaluation campaigns concentrate on evaluation of accuracy almost exclusively.

To start illustration with a fairly simple case, accuracy for a spelling checker plausibly breaks down into two sub-characteristics. The first of these is being able to identify strings of characters which do not constitute legal words of the language in question, signalling them and only them as potential spelling mistakes. The second is being able to propose plausible corrections. Proposing plausible corrections in its turn breaks down into two sub-characteristics. The first concerns whether the checker proposes the right correction; the second concerns the position of the right correction in the list of suggestions, assuming that more than one suggestion is made. With recent spelling checkers, a third sub-characteristic of accuracy might be the ability to identify correctly the language of a passage of text.

All of this sounds relatively straightforward, and we can rather easily imagine completing the model by associating metrics to the terminal nodes. For example, we might create a list of words, generate from that list a set of mistaken words, and use the list of mistakes to discover what percentage of our mistaken words are identified as such. Then, using the original list of words to provide us with a definition of what the right answer should be, we can discover in what percentage of cases the right word is proposed. It is a relatively

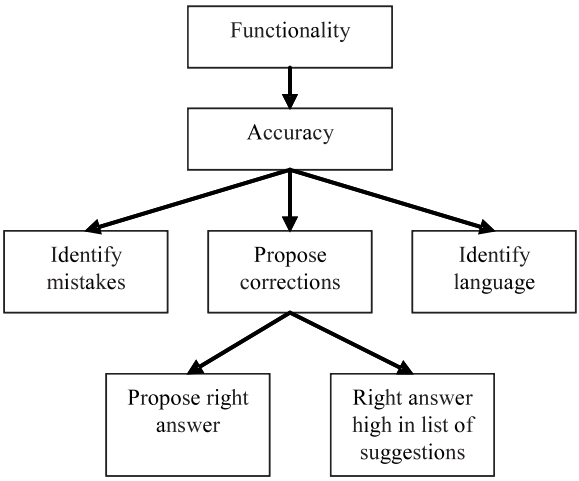

Figure 1. Substructure for the fragment of the quality model.

easy matter to check what position in the list of suggestions is occupied by the right proposal.[9] Finally, we can construct a text composed of fragments of text of a reasonable length for each of the languages which interests us, and use that as test data to discover whether the languages are correctly identified (for actual evaluations along these lines, see TEMAA, 1996 and Starlander and Popescu-Belis, 2002).

This gives us the substructure for the fragment of the quality model we are currently concerned with (Figure 1).

Unfortunately, even though superficially this looks convincingly tidy, defining accuracy for human language technology software is seldom as straightforward as it seems. What counts as a legal word of the language is mainly built into the software when it is released: in the most common case the software consults a built-in dictionary of legal words and if the string in the text does not correspond to an entry in the list, it is signalled as a mistake. Thus, accuracy in the ISO sense of conforming to specifications only depends on the software being able to identify correctly words which are not in the dictionary. But in the caricature case, the dictionary of legal words may be so impoverished as to be practically useless, thus rendering the software unsuitable for a large class of users. So, even in this apparently very simple case, accuracy in a user-oriented evaluation is considerably less important than suitability. (Fortunately, the metrics proposed can be implemented in such a way that they reflect user needs through an appropriate choice of words included in the test material.)

All we are really doing here, of course, is reinforcing a point already made at some length in Section 5.1. The main reason for labouring the point is that,

as we noted there, academic evaluation has tended to concentrate on accuracy, usually defined in terms of a set of inputs and related outputs. (I have even heard it claimed that limiting evaluation in this way is the only respectable way for an academic to work on evaluation). To adopt the limitation does, however, assume that those responsible for the system specifications have been successful in forcing accuracy and suitability to coincide: they have correctly predicted what users will need. It would be an unwise evaluator who failed to examine this assumption.

So even in specializing the functionality characteristic to take account of the type of software to be evaluated, user needs play a major role. But the user needs are still being expressed in terms of what the software should do, rather than how it should do it. Most of the other quality characteristics take what the software should do for granted, and look at questions like how fast it is, what memory resources it needs, how easy it is to install and maintain, how easy it is to learn and to use, and so on – all issues of obvious importance when specializing the model to account for a particular user's needs.

An interesting exception for human language technology is maintainability, where a note to the ISO definitions makes it clear[10] that maintainability includes adapting a piece of software to meet end-user requirements. Many language technology products critically include adapting the software to meet specific needs. A spelling checker allows the user to enter items in a personal dictionary, thus avoiding new terminology, being constantly flagged as unknown. A dictation system usually gives best results if trained to a particular voice. The results of an alignment algorithm improve if the user is allowed to specify a list of abbreviations which should not cause segmentation to take place. A machine translation system performs better if the user can influence the contents of the dictionary. None of these change the basic functioning of the software, and so are not, in that sense, part of functionality. In a way, they are simply more radical examples along a continuum that starts with being able to customize software by changing colours or creating personalized tool bars, but they have a more direct influence on suitability: they offer a first example where some might find the classification under a particular quality characteristic rather arbitrary.[11]

Other examples where an evaluator might be unsure as to where an attribute fits in the quality model can be found if we think about particular applications. For example, one feature of machine translation systems which is likely to be of interest to many users is the speed with which the translation is produced. Put like that, this attribute looks as though its natural place is a subcharacteristic of efficiency, under time behaviour. But then, if we take into consideration that the time to produce a usable (for whatever purpose) translation may have to include reading through the output and perhaps modifying it to improve translation quality, it begins to seem that the interesting metric is not

how many words an hour of raw output can be produced, but how many words of usable output. And once that move has been made, there will be some who think that how quickly usable output can be produced is part of suitability rather than efficiency, or others who think that two attributes are needed rather than one.

This latter stance is reflected by those who have used two metrics, the first typically measuring the number of words of raw output produced in some specific period of time, the second measuring how long it takes to produce useable output. The first of these metrics is rather easy to define, but it might be worth anticipating some later discussion by dwelling briefly on the difficulty of defining the second. The problem is twofold: first someone has to decide what counts as usable output; second, producing useable output (under any definition) from raw machine translation output necessarily requires human intervention, and human behaviour is affected by physical and emotional condition as well as by attitude to the task in hand. (An anecdote once popular amongst machine translation evaluators recounts that it was possible to reverse human judgements about which translations had been produced by a machine and which by humans simply by presenting the former beautifully typed on A4 paper and the latter badly laid out on line printer paper.) In practice, the two problems have usually been confounded and compounded: the most usual definition of a metric based on the time needed to produce useable output requires human beings to edit the raw output in order to produce what in their opinion is useable output, and measures the time they take to reach this goal. (Discussion of this and related questions can be found in Slocum et al., 1985). We shall come back to the difficulties posed by metrics which inherently require human involvement in a later section.

To return to quality models at a more general level, it has already been pointed out (Section 5.1) that mental hygiene does indeed impose a certain artificiality on the structure of quality models, but it is far more important in these cases to insist yet again that the real meaning of any substructure of the hierarchy is given by the metrics associated with the terminal nodes. The names of the features are, as a philosopher once said in a slightly different context, merely pegs on which to hang descriptions, the descriptions being the expression of a node by the nodes depending on it and, ultimately, by the measurable attributes found on the terminal nodes.

5.4 Combining Metrics in a Quality Model

It may seem that some of the quality characteristics have received very cavalier treatment so far, having been dismissed with a remark that they constitute constraints on the acceptable performance of the system rather than a description of what the system actually does. They come into their own when we start to consider the final way of tailoring the quality model to reflect specific

user needs, since they carry the main burden of capturing the specific intended context of use.

The key notion here is that of the relative importance of nodes at the same level in the hierarchical structure. As a straightforward example, let us take the quality of portability and its sub-characteristic replaceability, where replaceability covers the capability of the software to be used in place of another software product for the same purpose in the same environment, e.g., when a software is upgraded. Some companies producing translation memory software produce new versions of their software at very frequent intervals. Translation memory systems make use of an archive of previous translations, where each sentence translated is linked to its translation. These translation archives represent an investment of considerable value: there is a direct relationship between the richness of the memory and the productivity gains resulting from using the memory for translation. If, then, installing a new version of the system means that memories created with the previous versions can no longer be used, no matter what other advantages the new version might offer, changing to the new version loses much of its attraction.[12] In other words, replaceability becomes a critical attribute, whose value may even determine the outcome of the evaluation as a whole. Of course, for someone who is thinking of buying his/her first translation memory software and who has no resources to exploit, replaceability is of no importance at all: what may be critical for one user may be totally irrelevant for another.

The combining function mentioned briefly in Section 5.2 is meant to allow expression of this notion of relative importance. For the user for whom replaceability is critical, the evaluation designer will give it a combining value such that it outweighs any other sub-characteristics. For the user for whom it does not matter at all, he will give it a value equivalent to saying that it should be neglected in the evaluation. Thus, part of tailoring the evaluation to the needs of the specific user is defining how the values from each level of the quality model are to be combined in order to pass them to a higher level. By definition, the combining function is specific to a particular evaluation: it is only the existence of such a mechanism which forms part of the definition of the model itself under the EAGLES extension. In terms of the ISO standards, the definition of a combining function corresponds to a part of defining the process of evaluation, as set out in the ISO/IEC 14598 series. It is part of the step described there as specifying the evaluation, where, after metrics have been chosen, rating levels for those metrics are established and criteria for assessment are also established. We discuss this step in Section 5.5. We shall come back to the discussion of metrics in more detail in Section 6.

5.5 Relating Performance to User Satisfaction

A very common type of metric typically involves producing a score on some scale, reflecting the particular system's performance with respect to the quality characteristic in question. This score, uninterpreted, says nothing about whether the system performs satisfactorily. To illustrate this idea, consider the Geneva education system, where marks in examinations range from 1 to 6. How is it possible to know, other than by being told, that 6 is the best mark and 1 the worst? In fact, most people from other systems will probably have guessed that it is so: they may then have difficulty in some other cantons where 1 is the highest mark. (I have been told that the lack of consistency in how examination marks are awarded in Switzerland is at the root of an urban myth about Einstein's performance in secondary school.) Establishing rating levels for metrics involves determining the correspondence between the un-interpreted score and the degree of satisfaction of the requirements.

Not all attributes acquire a numerical value when their metrics are applied. For example, the attribute reflecting which language pairs a machine translation system covers has a non-numerical value, as does the attribute covering what platform the software needs. Rating levels are also a way of ironing out differences in type across metrics that have to be combined. Since quality refers to given needs, there can be no general rules for when a score is satisfactory. This must be determined for each specific evaluation.

Each measure, interpreted by its rating level, contributes to the overall judgement of the product, but not necessarily in a uniform way. It may be, as we have seen earlier, that one requirement is critical, whilst another is desirable, but not strictly necessary. In this case, if the system performs badly with respect to the critical characteristic, it will be assessed negatively no matter what happens to all the other characteristics. If it performs badly with respect to the desirable but not necessary characteristic, it is its performance with respect to all the other characteristics which will determine whether the system is acceptable or not.

This consideration is familiar from discussion of the EAGLES/ISLE combining function. In ISO 14598 it feeds directly into establishing criteria for assessment, which involves defining a procedure for summarizing the results of the evaluation of the different characteristics, using, for example, decision tables or weighting functions of different kinds.

6 A Fourth Principle: Quality can be Measured

6.1 Defining and Validating Metrics

By now, the reader will need very little persuading that the utility and worth of a quality model depends critically on the metrics associated with the measurable attributes forming the terminal nodes of the quality model structure.

A primary constraint on a metric is that it should be valid, i.e., it should in fact measure what it purports to measure. This sounds blindingly obvious, but the evaluation literature abounds in metrics which fail to meet this stipulation. The social sciences literature is rich in discussion about validity. One distinction made there which was picked up by early EAGLES work is a distinction between internal validity and external validity. A metric is internally valid if its validity is guaranteed by the nature of the metric itself. It is externally valid if the results obtained by applying the metric correlate with the feature of interest without directly measuring it. An informal example of an internally valid metric is given by the way reading ages are tested. Reading age is first extensionally defined by drawing up lists of the words a child should be able to read at a given age. The reading age of a given child is then determined by asking him to read aloud texts which contain the vocabulary defining a specific age. His ability to do so determines whether he has reached the reading age defined by the vocabulary. The definition, in other words, is circular: reading age is defined by being able to read a certain set of words, and is tested for by asking that those words be read: validity is internal to the metric. An informal example of an externally valid metric comes from the questionnaires that life insurance companies ask potential customers to fill in. They clearly cannot sensibly ask how long the person to be insured will live, so they ask what his weight is, whether he smokes, if he has diabetes, if he has ever had major surgery, and so on – all factors which correlate closely with average life expectancy.

In human language technology evaluation, the word error rate metric used with speech recognition systems seems to offer a clear example of a metric which relies on internal validity. The speaker speaks a known word: if that word is correctly transcribed, the system produces the right answer. The number of right answers out of the total test set determines the system's score. In evaluation jargon, there is a gold standard which determines what the right answer should be.

Most evaluation campaigns have been based on the creation of gold standards. Their production is frequently a costly and contentious business, simply because there are relatively few applications where the right answer is easily defined. A couple of examples will illustrate this. Fact extraction systems take text as input and produce as output information extracted from that text, often in the form of a template where the system's task is to fill in slots in an

appropriately chosen template. For example, from the sentence "The minister for foreign affairs will visit Paris on January 4th", a system might be expected to produce a structure[13] like:

(ACTION: visit
AGENT: minister for foreign affairs
LOCATION: Paris
DATE: January 4th)

The system would probably be expected to produce the same template from the sentence "January 4th is the date set for the visit by the minister for foreign affairs to Paris" or even from "A visit to Paris on January 4th is part of the schedule planned for the minister for foreign affairs". A collection of texts and a set of filled templates based on those texts constitute the gold standard for the evaluation of such systems.

The problem is obvious: how is it decided what templates should exist, what slots they should have, and what the fillers for those slots should be? Furthermore, how are the limits on what the system can be expected to do decided? If the sentence is "Utopia's most notorious minister is expected to cause major controversy by visiting the capital of France on the 4th of next month", can the system still be expected to extract the same slot fillers? Within an evaluation campaign, a common solution is to seek consensus amongst interested parties in these cases (Lehnart and Sundheim, 1991 discuss some of the issues raised by consensus seeking). Creating the test data is in itself expensive: when the cost of producing consensus is added in, test data of this kind can become a resource of considerable monetary value. Expense also helps to explain why test data is frequently reused.

Similar problems arise with applications like document retrieval, where judging the relevance of a retrieved document is of major importance in evaluating the system's success. Relevance judgements can be challenged, so some way of convincing both evaluators and those being evaluated of their acceptability has to be found. The TREC conferences[14] have been prolific in discussion of this issue and ingenious in ways of getting round it (see, e.g., Voorhees, 2000, 2003; Sparck Jones, 2001).

The root of the problem, of course, is that there is, in these cases, no answer which is indisputably right. The gold standard is achieved not by looking for absolute truth, but by seeking a wide enough agreement on what will count as right. Nonetheless, once the consensus has been achieved, the gold standard forms an intrinsic part of the metrics using it: the metrics achieve an internal validity.

There are, however, applications where even creating a right answer by consensus is problematic. One such is machine translation. It is in the nature of translation that there can be no single correct translation of a source text: the chances that any two human translators would come up with exactly the same

translation for a sentence of reasonable length are very slim, but both their translations may be equally acceptable.

For this reason, most of the metrics historically used in machine translation evaluation have tended to rely critically on human judgement. Many ask human subjects to give a score to a segment (usually a clause or a sentence) of machine translation output based on a judgement of its intelligibility, fluency, accuracy, or some similar characteristic. These metrics suffer from several weaknesses. First, there is the problem we have already alluded to: human beings are not robots. They are impatient when they get tired, they may love or hate machines, they may resent having to take part in an exercise where they think they already know the outcome, they may believe that their future employment depends on the outcome of the evaluation exercise – the number of factors which might influence their behaviour is so large and so various that it is almost impossible to control for. Second, the instructions on how to apply the metrics are usually expressed in natural language and therefore interpretable by different people in different ways. Even if the decision is seemingly quite simple, in the style of "score 1 if the output is intelligible, 0 if it is not" experience has shown that intersubject reliability is far from guaranteed. A growing awareness of such problems (discussed, e.g., in King, 1996a, b, 1997) led to attempts to circumvent some of the problems by asking subjects to read the raw output and then to complete a comprehension test where the questions were based on the content of the original texts (see, e.g., White and O'Connell, 1994) . Even these metrics, however, are not exempt from the human interference syndrome: at the very least, comprehension tests are used in other areas in order to assess general intelligence. By definition then, some humans will be better at working out the correct answers than others, even when the machine translation output is unchanged.

And of course all these metrics suffer from one great weakness: they are expensive to implement. Setting up the tests cost money, human subjects have to be found and perhaps paid for their participation and human analysis of the raw results is required and must be paid for.

A number of recently proposed metrics, foreshadowed in Thompson (1992) but in practical terms starting with the BLEU metric (Papinieni et al., 2001), try to overcome the problems sketched above by applying quite complex statistical analysis to determine how close a candidate translation is to a set of what are called reference translations. Essentially, the metric looks at small stretches of the machine translation output (typically three or four words) and determines whether the stretch being examined also occurs in the reference translation(s). The overall score for the candidate translation is based on how many small stretches have their equivalent in the reference.[15]

It is clear even from this very brief and informal description that BLEU and other measures like it which depend on comparison with a (set of) reference

translations do not really resolve the problem of finding a gold standard metric for translation quality, since the validity of the metric depends critically on the quality of the reference translation(s): in terms of earlier discussion, the validity of the metric is internal – if all (or even some of) the translations in the reference set are poor, the scores produced by applying the metric will not reflect what would normally be thought of as acceptable quality in the translation. For this reason, there has been much interest in checking, for particular evaluations, whether the results correlate with human judgement of the same machine translation outputs,[16] thus bringing us back to issues of intersubject reliability and economy.

Sidestepping the translation quality issue by using a set of reference translations is the real potential merit of these metrics, but in turn raises the practical problem of acquiring multiple reference translations in the appropriate domain, style, and register. This, of course, is not a new problem; it is closely akin to the problem of acquiring suitable training corpora for any empirically based system. But that it can be hard to solve is shown by the fact that theoretical work on the metrics has sometimes been forced to use literary or religious texts, and the perception is reinforced by the number of applications of the metrics which in the end use only one, or at best a very small number of, reference translations.

BLEU and related metrics are far from universally accepted for other reasons too: sentence length may adversely affect their general validity, and relatively little work has so far been done on how they function with languages where word order is free, making it more unlikely that even a short segment of the candidate translation text will exactly correspond to a segment of the reference translations. Thus, there is still much controversy about these metrics, as can be seen from the proceedings of almost any recent conference on machine translation.[17]

So far, the problems we have discussed come from the nature of the application. Other problems come from the data over which a software system is supposed to work. Document retrieval on the web offers a familiar example. A search engine responds to a query by searching the web for documents which match the query terms. Neither user nor evaluator can know what documents are available: the web is both vast and shifting – what is there in the morning may be gone by the afternoon, and new documents will certainly have appeared. A consequence of this is that although it is possible (at least in theory) to check that all the documents retrieved by the search engine do in fact match the query terms, it is not even theoretically possible to determine whether there were other documents available at the time of the search which should have been retrieved and were not.[18] If called upon to evaluate a search engine, all we can do is constrain the document collection, as is conventionally done in the evaluation of document retrieval systems, and assume that by external validity

behaviour over the constrained collection correlates with behaviour over the unconstrained set of documents available (see TREC-2004 web track for an example of this strategy).

Another issue raised by a document collection which is constantly changing is that of reliability: a metric should be reliable in the sense that if it is applied in the same context on different occasions, it should produce the same result – in experimental jargon, the results should be replicable. The problem with searching on the web is exactly that we cannot guarantee that the context will remain the same. Once again, we are forced to constrain the context artificially in order to ensure reliability of the metric.

Reliability is a general issue which deserves much more discussion than the brief mention it will get here. We shall only add that pilot testing can help to ensure that a metric has no intrinsic reliability weaknesses, and paying particular attention to reliability issues when execution of the evaluation is being planned can help to eliminate practical problems.

6.2 Interaction between Humans and Metrics

Both validity and reliability are involved in the very delicate issue, already referred to, of human participation in defining and applying metrics. The problem is brutally simple: human beings are neither standardized nor automata. Behaviour varies from one human to another and even a single individual will perform differently depending on his/her state of health, how tired he/she is, and other inescapable natural factors. We have already discussed this problem to a certain extent, using metrics from machine translation evaluation as examples, such as metrics based on humans completing comprehension tests (see White and O'Connell, 1994 for discussion of these and similar issues). In that same discussion we pointed out that evaluators have often sought to eliminate human participation from their metrics. But it is not always possible to do so, essentially for two reasons. The first is that, as we have already mentioned, there are softwares which depend on interaction with a human; they are simply not designed to produce satisfactory results without the intervention of a human to guide their functioning. In these cases it is a major challenge to devise metrics that test the performance of the software independently of the ability of the human partner. In many cases, all that can be done is to be aware of the problem and to choose the population of human partners very carefully. Second, there are quality characteristics which cannot be measured at all without making use of humans. How, for example, can attractiveness (a sub-characteristic of usability) be measured except by asking humans for their judgement? In these cases too, all the evaluation designer can do is to be aware of potential problems and define a population of test subjects accordingly.

The choice and definition of metrics is a very thorny business about which much more deserves to be said than there is space for here. The recent

publications in the ISO 9126 series have much to say on the matter, and discussion of a set of formal coherence criteria for metrics can be found in Hovy et al. (2002b).

Discussion of particular metrics can be found widely in almost any recent conference on computational linguistics or on applications of human language technology. Discussion of machine translation metrics in particular can be found in the documents pertaining to the ISLE workshops, available at http://www.issco.unige.ch/projects/isle.

7 Combining the Particular and the General: The Ideal

Sections 3 and 4 laid emphasis on the need to take into account the quality requirements of individual users. Section 5 then tried to compensate for a strongly bottom-up flavour by suggesting that a quality model conceived at a sufficiently high level could be designed, and that such a model could offer the evaluation designer a way into being systematic about defining what a particular user might need.

This section attempts to pull these two strands of thought together, by suggesting that by thinking in terms of classes of users, it should be possible to create a fully worked-out quality model that would in some sense be the union of the needs of all users. Designing a particular evaluation would then become a question of picking out from the general model just those requirements which are relevant to the specific evaluation being designed in order to create a tailor-made evaluation – a little like the pick n' mix sweet counters in the supermarket.

This is exactly the idea behind the FEMTI model for evaluation of machine translation systems, mentioned in Section 1. FEMTI sets up two taxonomies. The first is a classification of contexts of use in terms of the user of the machine translation system and the translation task to be accomplished, including characteristics of the input to the system. The second is a classification of the quality characteristics of machine translation software, detailed into hierarchies of sub-characteristics and attributes, bottoming out into metrics at the terminal nodes. The upper levels coincide with the ISO 9126 characteristics. The model is completed by a mapping from the first classification to the second, which defines (or at least suggests) the characteristics, sub-characteristics, and attributes or metrics that are most relevant for each context of use. The nodes of the two taxonomies frequently contain additional information in the form of bibliographic references or explicit mention of the type of user or stakeholder whose interests might be represented by the node.

In an ideal world, the structure described briefly above would be entirely automated. An evaluation designer would click on a section of the user needs/context of use taxonomy and would thereby bring up the relevant nodes

from the quality characteristics taxonomy, together with a choice of relevant metrics. All he/she would have to do to complete the evaluation design would be to reply, when prompted, with information on the rating levels for this particular evaluation and on the combining function. Estrella et al. (2005) give a more detailed account of FEMTI and of preliminary work on establishing links between the two taxonomies.

At the moment, this is a utopian dream. Constructing even the current sloppy version of FEMTI has been long and arduous, and its constructors are well aware of lacunae and the continuing existence of inconsistencies. Perhaps even worse, in its current state it is almost totally uncritical about the metrics attached to the terminal nodes: the metrics have simply been collected from the literature and very little has been done to validate them or to investigate relationships between them – this is on the agenda for the next round of work.

There is also a strong sense, of course, in which work on defining such generic quality models can never be finished. Technology moves at ever-increasing speed, and systems change in consequence. Interest in the development of new metrics and their validation has not ceased to grow over the last few years, and with the economic stakes growing ever larger as systems become evermore complex, there is no reason to think that this interest will wane.

Nonetheless, it is by striving towards the construction of the utopia that we deepen our knowledge of what evaluation is all about.

8 Conclusion

The direction in evaluation work reflected in this chapter started with a desire to share expensive resources. The obvious question as we reach the end of the chapter is whether that has in any way been achieved by the work reported on here. I think it cannot be denied that the EAGLES-inspired work on user-oriented evaluation has been stimulating to the large community of research workers and other interested parties who have participated in it: empirical reinforcement of this claim comes from the fact that there is never any lack of potential participants whenever a new workshop is announced. The most obvious result is the growth of a common framework for thinking about evaluation which goes further than concentrating on what the software is supposed to do. Then too, the scientific community has become much more sophisticated about metrics and their application over the last decade or so, partly under the influence of a continuing interest in evaluation campaigns, partly through discussion stimulated by work in the EAGLES, and other similar contexts. We have not found any magic recipes for evaluating natural language software: it would have been naive to imagine that we might. We have made a lot of progress towards being able to justify or criticize a particular

evaluation on reasoned and reasonable grounds, and we have made it easier for the evaluation designer to set about his/her job in a systematic fashion, with the confidence that what he/she is doing is grounded in accepted standards.

Notes

1. Expert Advisory Groups for Language Engineering Standards.
2. International Standards for Language Engineering.
3. To foreshadow later discussion, it is perhaps interesting to notice here already that the change in needs has direct consequences on what metrics might be suitable. In particular, a measure of translation quality based on whether or not relevant newspaper articles can be identified is, in this new context, useless.
4. The examples here are complete systems, but in a context like that of a research the same reasoning would apply to individual modules of the overall system; what would change would be the kinds of users.
5. Quotations from ISO/IEC documents are made with ISO permission, granted in the context of the EAGLES and ISLE projects.
6. This is of course the caricature case. Products actually on the market use a variety of devices to cut down the noise and avoid silence. Even so, producing suitable results remains a major issue for current terminology extraction tools, and even more so when they also try to extract a potential translation.
7. Whilst encouraging him, of course, to consult the more detailed definitions of the ISO standard itself.
8. The combining function is not as simple as it is being made to seem here.
9. This is very similar to the word error rate metric (see Section 6).
10. By pointing out that if the software is to be modified by the end-user, changeability may affect operability.
11. A point reinforced by one of the reviewers suggesting that a clearer distinction between maintainability (in the sense of it being possible for people other than those who wrote the original code to make straightforward adjustments to it) and adaptability (in the sense of being able to extend the software to do things that were not originally foreseen) is required. The ISO definition of maintainability includes both as part of the same sub-characteristic, the notes on that characteristic making it clear that this is a deliberate choice.
12. The TMX exchange format standard for translation memories was developed in order to avoid this kind of problem.
13. This example has been invented for the purposes of exposition here: any correspondence to the structures produced by a particular system is entirely accidental.
14. Text Retrieval Conference (TREC) TREC-9 Proceedings are available electronically at http://www.trec.nist.gov.trec9.t9-proceedings.
15. This is a ridiculously simplified account. The reader is referred to the literature for a more accurate and more detailed description.
16. See Lin and Och (2004) for a discussion of several automated machine translation metrics and of how they correlate with human judgements, together with a proposal for evaluation of the metrics themselves. A comparison of a number of metrics and their results when applied to working systems can also be found in Surcin et al. (2005).
17. Proceedings available electronically at http://www.amtaweb.org/summit/MTSummit/papers.html.
18. To state this in terms of well-known evaluation metrics: precision, first used as metric in document retrieval, is based on what proportion of the documents retrieved are actually relevant to the search request. In the context described here, it is theoretically (if not always practically) possible to measure precision. Recall, on the other hand, is based on measuring how many, out of all the relevant documents existing in the document set being searched, are actually retrieved. Measuring recall is not even theoretically possible in the web context: there is no possible way of knowing either what the collection of documents being searched over is, or what the relevant documents in that collection are.

References

AMTA (1992). MT Evaluation: Basis for Future Directions (Proceedings of a Workshop held in San Diego, California, USA). Technical report, Association for Machine Translation in the Americas.

Ankherst, M. (2001). Human Involvement and Interactivity of the Next Generation's Data Mining Tools. In *Proceedings of the DMKD Workshop on Research Issues in Data Mining and Knowledge Discovery*.

Blair, D. C. (2002). Some Thoughts on the Reported Results of TREC. *Information Processing and Management*, 38(3):445–451.

Boisen, S. and Bates, M. (1992). A Practical Methodology for the Evaluation of Spoken Language Systems. In *Proceedings of the Third Conference on Applied Natural Language Processing (ANLP)*, pages 162–169, Trento, Italy.

Booch, G., Rumbaugh, J., and Jacobson, I. (1999). *The Unified Modeling Language: User Guide*, Addison Wesley, Reading, USA.

Bourland, P. (2000). Experimental Components for the Evaluation of Interactive Information Retrieval Systems. *Journal of Documentation*, 56(1): 71–90.

Brown, A. and Wallnau, K. (1996). A Framework for Systematic Evaluation of Software Technologies. *IEEE Software*, 13(5):39–49.

Canelli, M., Grasso, D., and King, M. (2000). Methods and Metrics for the Evaluation of Dictation Systems: A Case Study. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC)*, pages 1325–1331, Athens, Greece.

Church, K. W. and Hovy, E. H. (1993). Good Applications for Crummy MT. *Machine Translation*, 8:239–258.

Cowie, J. and Lehnert, W. (1996). Information Extraction. *Communications of the ACM, Special Edition on Natural Language Processing*, pages 80–91.

Doyon, J., Taylor, K., and White, J. S. (1998). The DARPA MT Evaluation Methodology: Past and Present. In *Proceedings of the Association for Machine Translation Conference (AMTA)*, Philadelphia, USA.

EAGLES Evaluation Working Group (1996). EAGLES Evaluation of Natural Language Processing Systems. Final report, Center for Sprogteknologi, Copenhagen, Denmark.

Estrella, P., Popescu-Belis, A., and Underwood, N. (2005). Finding the System that Suits You Best: Towards the Normalization of MT Evaluation. In *Proceedings of the 27th International Conference on Translating and the Computer (ASLIB)*, London, UK.

Falkedal, K., editor (1994). *Proceedings of the Evaluators' Forum*, ISSCO, Les Rasses, Switzerland.

Flickinger, D., Narbonne, J., Sag, I., and Wasow, T. (1987). Toward Evaluation of NLP Systems. Technical report, Hewlett Packard Laboratories, Palo Alto, USA.

Grishman, R. (1997). Information Extraction: Techniques and Challenges. International Summer School on Information Extraction (SCIE). New York University, New York, USA.

Hartley, A. and Popescu-Belis, A. (2004). Evaluation des systèmes de traduction automatique. In Chaudiron, S., editor, *Evaluation des systèmes de traitement de l'information*, Collection sciences et technologies de l'information, pages 311–335, Hermès, Paris, France.

Hawking, D., Craswell, N., Thistlewaite, P., and Harman, D. (1999). Results and Challenges in Web Search Evaluation. *Computer Networks*, 31(11-16): 1321–1330.

Hirschman, L. (1998a). Language Understanding Evaluations: Lessons Learned from MUC and ATIS. In *Proceedings of the First International Conference on Language Resources and Evaluation (LREC)*, pages 117–123, Granada, Spain.

Hirschman, L. (1998b). The Evolution of Evaluation: Lessons from the Message Understanding Conferences. *Computer Speech and Language*, 12:281–305.

Hovy, E. H., King, M., and Popescu-Belis, A. (2002a). Computer-Aided Specification of Quality Models for Machine Translation Evaluation. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, pages 729–753, Las Palmas, Gran Canaria, Spain.

Hovy, E. H., King, M., and Popescu-Belis, A. (2002b). Principles of Context-Based Machine Translation Evaluation. *Machine Translation*, 16:1–33.

ISO/IEC 14598-1:1999. Information Technology – Software Product Evaluation, Part 1: General Overview. International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland.

ISO/IEC 14598-2:2000. Software Engineering – Product Evaluation; Part 2: Planning and Management. International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland.

ISO/IEC 14598-3:2000. Software Engineering – Product Evaluation, Part 3: Process for Developers. International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland.

ISO/IEC 14598-4:1999. Software Engineering – Product Evaluation, Part 4: Process for Acquirers. International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland.

ISO/IEC 14598-5:1998. Information Technology – Software Product Evaluation, Part 5: Process for Evaluators. International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland.

ISO/IEC 14598-6:2001. Software Engineering – Product Evaluation, Part 6: Documentation of Evaluation Modules. International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland.

ISO/IEC 9126-1:2001. Software Engineering – Product Quality, Part 1: Quality Model. International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland.

ISO/IEC 9126:1991. Information Technology – Software Product Evaluation, Quality Characteristics and Guidelines for Their Use. International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland.

ISO/IEC CD 9126-30. Software Engineering – Software Product Quality Requirements and Evaluation, Part 30: Quality Metrics – Metrics Reference Model and Guide. International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland. In preparation.

ISO/IEC TR 9126-2:2003. Software Engineering – Product Quality, Part 2: External Metrics. International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland.

ISO/IEC TR 9126-3:2003. Software Engineering – Product Quality, Part 3: Internal Metrics. International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland.

ISO/IEC TR 9126-4:2004. Software Engineering – Product Quality, Part 4: Quality in Use Metrics. International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland.

King, M. (1996a). Evaluating Natural Language Processing Systems. *Special Edition of Communications of the ACM on Natural Language Processing Systems*, 39(1):73–79.

King, M. (1996b). On the Notion of Validity and the Evaluation of MT Systems. In Somers, H., editor, *Terminology, SLP and Translation: Studies in Honour of Juan C. Sager*, pages 189–205, John Benjamins, Amsterdam, The Netherlands.

King, M. (1997). Evaluating Translation. In Hauenschild, C. and Heizmann, S., editors, *Machine Translation and Translation Theory*, pages 251–263, Mouton de Gruyter, Berlin, Germany.

King, M. (1999). Evaluation Design: The EAGLES Framework. In Nübel, R. and Seewald-Heeg, U., editors, *Evaluation of the Linguistic Performance of Machine Translation Systems, Proceedings of Konvens'98, Bonn*, Gardezi Verlag, St. Augustin, Germany.

King, M., editor (2002). *Workbook of the LREC Workshop on Machine Translation Evaluation: Human Evaluators Meet Automated Metrics*, Las Palmas, Gran Canaria, Spain.

King, M. (2005). Accuracy and Suitability: New Challenges for Evaluation. *Language Resources and Evaluation*, 39:45–64.

King, M. and Falkedal, K. (1990). Using Test Suites in Evaluation of MT Systems. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, volume 2, pages 211–216, Helsinki, Finland.

King, M. and Maegaard, B. (1998). Issues in Natural Language System Evaluation. In *Proceedings of the First International Conference on Linguistic Resources and Evaluation (LREC)*, volume 1, pages 225–230, Granada, Spain.

King, M., Popescu-Belis, A., and Hovy, E. H. (2003). FEMTI: Creating and Using a Framework for MT Evaluation. In *Proceedings of MT Summit IX*, pages 224–232, New Orleans, USA.

King, M. and Underwood, N., editors (2004). *Proceedings of the LREC Workshop on User Oriented Evaluation of Knowledge Discovery Systems*, Lisbon, Portugal.

Kuralenok, I. E. and Nekrestyanov, I. S. (2002). Evaluation of Text Retrieval Systems. *Programming and Computing Software*, 28(4):226–242.

Lehmann, S., Oepen, S., Regnier-Prost, S., Netter, K., Lux, V., Klein, J., Falkedal, K., Fouvry, F., Estival, D., Dauphin, E., Compagnion, H., Baur, J., Balkan, L., and Arnold, D. (1996). TSNLP – Test Suites for Natural Language Processing. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 711–716.

Lehnart, W. and Sundheim, B. (1991). A Performance Analysis of Text-Analysis Technologies. *AI Magazine*, 12(4):81–94.

Lin, C.-Y. and Och, F. J. (2004). ORANGE: A Method for Evaluating Automatic Evaluation Metrics for Machine Translation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 23–27, Geneva, Switzerland.

Minker, W. (2002). Overview on Recent Activities in Speech Understanding and Dialogue Systems Evaluation. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, pages 337–340, Denver, Colorado, USA.

Nomura, H. and Isahara, J. (1992). JEIDA Methodology and Criteria on MT Evaluation. Technical report, Japan Electronic Industry Development Association (JEIDA).

Paggio, P. and Underwood, N. (1998). Validating the TEMAA Evaluation Methodology: A Case Study on Danish Spelling Checkers. *Natural Language Engineering*, 4(3):211–228.

Papinieni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2001). BLEU: A Method for Automatic Evaluation of MT. Research report, Computer Science RC22176 (W0109-022), IBM Research Division, T. J. Watson Research Center.

Slocum, J., Bennett, W. S., Whiffin, L., and Norcross, E. (1985). An Evaluation of METAL: The LRC Machine Translation System. In *Proceedings of the Second Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 62–69, Geneva, Switzerland.

Sparck Jones, K. (2001). Automatic Language and Information Processing: Rethinking Evaluation. *Natural Language Engineering*, 7(1):29–46.

Sparck Jones, K. and Galliers, J. R. (1996). *Evaluating Natural Language Processing Systems: An Analysis and Review*. Number 1083 in Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, Germany/New York, USA.

Sparck Jones, K. and Willet, P., editors (1997). *Readings in Information Retrieval*, Morgan Kaufman, San Francisco, USA.

Starlander, M. and Popescu-Belis, A. (2002). Corpus-Based Evaluation of a French Spelling and Grammar Checker. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, pages 262–274, Las Palmas, Gran Canaria, Spain.

Surcin, S., Hamon, O., Hartley, A., Rajman., M., Popescu-Belis, A., Hadi, W. M. E., Timimi, I., Dabbadie, M., and Choukri, K. (2005). Evaluation of Machine Translation with Predictive Metrics beyond BLEU/NIST: CESTA Evaluation Campaign #1. In *Proceedings of the Machine Translation Summit X*, pages 117–124, Phuket, Thailand.

TEMAA (1996). TEMAA Final Report. Technical Report LRE-62-070, Center for Sprogteknologi, Copenhagen, Denmark.

Thompson, H. S. (1992). The Strategic Role of Evaluation in Natural Language Processing and Speech Technology. Technical report, University of Edinburgh, UK. Record of a workshop sponsored by DANDI, ELSNET and HCRC.

VanSlype, G. (1979). Critical Study of Methods for Evaluating the Quality of MT. Technical Report BR 19142, European Commission, Directorate for General Scientific and Technical Information Management (DG XIII). http://www.issco.unige.ch/projects/isle.

Voorhees, E. (2000). Variations in Relevance Judgements and the Measurement of Retrieval Effectiveness. *Information Processing and Management*, 36:697–716.

Voorhees, E. (2003). Evaluating the Evaluation: A Case Study Using the TREC 2002 Question Answering Track. In *Proceedings of the HLT-NAACL*, pages 181–188, Edmonton, Canada.

White, J. S. and O'Connell, T. A. (1994). The DARPA MT Evaluation Methodologies: Evolution, Lessons and Future Approaches. In *Proceedings of the First Conference of the Association for Machine Translation in the Americas (AMTA)*, Columbia, Maryland, USA.

Yeh, A. S., Hirschman, L., and Morgan, A. A. (2003). Evaluation of Text Data Mining for Data Base Curation: Lessons Learned from the KDD Challenge Cup. *Bioinformatics*, 19(suppl. 1):i331–i339.

A note on the bibliography: Evaluation campaigns and projects can span many years and give birth to numerous publications. Here, only one reference is given to any single long-term effort, even though other publications may contain discussion which has been picked up here. The reference chosen is usually either the most recent or a retrospective summary. A much more detailed bibliography can be obtained directly from the author or from http://www.issco.unige.ch/.