Test Suites: some issues in their use and design

Lorna Balkan
Department of Language and Linguistics
University of Essex
Wivenhoe Park
Colchester
UK

1 Introduction

Evaluation has always been a subject of interest to the MT community. [1] It has also been a source of grief, as witnessed by the damning ALPAC Report (see Pierce and Carroll, 1966) that led to the virtual end of government funding for MT in the USA in the sixties since it concluded that there was no immediate prospect of MT producing useful translation of general scientific texts. However, MT and evaluation techniques have advanced since then, and evaluation in particular has experienced a renaissance in the last ten years. It is a topic that is currently attracting a great deal of interest in the Natural Language Processing community at large, not just in MT. Within the European Community, several evaluation projects and initiatives are being funded, including EAGLES (Expert Advisory Group on Language Engineering Standards), which was set up in 1993 to improve evaluation methods as a step towards setting up standards for language engineering products. The European Community is also jointly funding the project Test Suites for Natural Language Processing (TSNLP) which is described in detail below.

We begin by defining the notion of test suite and describe its role within the evaluation process. We then look at current approaches to test suite construction, with particular reference to test suites for MT systems. Finally we introduce the TSNLP project and discuss the improvements it proposes for test suite design and construction.

2 Test Suites: an introduction

It is useful to begin by defining what we mean by test suites, and to say something about their role in evaluation. (Useful background discussion can be found in inter alia Arnold et al. 1993a). There are basically three types of test material:

- Test corpora: a collection of naturally occurring text, increasingly in electronic form.

[1] I would like to thank my colleagues in TSNLP for fruitful discussions on this topic: Doug Arnold, Eva Dauphin, Dominique Estival, Kirsten Falkedal, Sabine Lehmann, Siety Meijer, Klaus Netter and Sylvie Regnier-Prost. Special thanks are due to Kirsten Falkedal

- Test suites: a collection of (usually) artificially constructed inputs, where each input is designed to probe a system's treatment of a specific phenomenon or set of phenomena. Inputs may be in the form of sentences, sentence fragments, or even sequences of sentences.

- Test collections: a set of inputs associated with a corresponding set of expected outputs. This type of test material is increasingly common and has been used in the evaluation of parsers (in the Parseval project, for example - see Thompson 1992) and other Natural Language Processing applications.

The problem with test collections is being able to specify an appropriate output for a system. Output from parsers can be many and varied. The Parseval project, in common with other parser evaluation projects, uses hand-produced ideal parses of sentences from the Penn Treebank, a parsed corpus, to compare parser output against. Machine translation shares a similar problem - there is no one correct output. While at present no test collections exist for MT, it is possible to imagine producing an ideal translation, in the same way as an ideal parse. Support for this idea is based on a proposal of Henry Thompson (see Thompson 1991) that is currently being investigated in a research project in Edinburgh, that though a wide range of translations may be possible, "good" translation will tend to be more similar to each other than bad ones – "good" translation will tend to cluster together.

At present, the two principal test methods for MT evaluation at any rate, are test suites and test corpora. These techniques have different roles to play in evaluation and are often both required to perform a full evaluation of a system. Test suites are useful for presenting language phenomena and combinations of phenomena in an exhaustive and systematic way. Furthermore, negative data can be derived systematically from positive data by violating grammatical constraints associated with the positive data item. Test corpora, on the other hand, lack the exhaustivity and systematicity of test suites, but their strength lies in the fact that they contain naturally occurring data (test suite examples are often seen as being "contrived"). So, if one is interested in testing a specific phenomenon (eg. relative clauses) in depth, the test suite method is to be preferred. If on the other hand, one is interested in seeing how one's system performs on real life text, the test corpus method is preferable.

It has to be stressed that not all linguistic phenomena are equally amenable to the test suite method. Its use is for the most part limited to the testing of syntactic phenomena, mainly because syntactic phenomena are relatively well understood and well documented. This is not the case with semantic and pragmatic phenomena. Furthermore, semantic and pragmatic phenomena are often context sensitive. This means that they require to be tested within a sequence of sentences, rather than in isolated sentences. This is where test corpora are useful, because they just are a sequence of sentences. Some suggestions for what should go into a semantic test suite are discussed in Hoard (1991).

Traditionally, test suites and test corpora have been associated with different types of evaluation. EAGLES, for example, distinguishes the following three types of evaluation:

- diagnostic evaluation, which aims at localizing deficiencies;

- progress evaluation, for a comparison between successive stages of development of a system; and

- adequacy evaluation, to determine whether and to what extent a particular system meets some pre-specified requirements.

Test suites are particularly well-suited for diagnostic evaluation, while test corpora will be necessary to test a system's overall performance on some text type. Diagnostic evaluation is typically what the system developer does, although he might also be interested in performing an adequacy evaluation if he is aiming at a market product. Likewise, the end user is typically associated with adequacy evaluation, although he too may be interested in performing some kind of diagnostic evaluation on his system to locate errors and judge the system's potential from an error analysis. Thus test suites are a useful tool for developers and users alike.

3 Test suite construction: state of the art

There are three main approaches to test suite construction:

- the bottom up approach
- the top down approach
- the mixed approach

3.1 The bottom up approach

The bottom up approach starts with the system under test, and analyses it in terms of its function. The approach is exemplified by EAGLES which advocates that a system be analysed in terms of its functions. By way of example, the functions of a spelling checker might be:
– detection of mis-spelt words where the mis-spelling does not correspond to a legitimate word of the language
– a proposal of plausible corrections.
These functions are translated into reportable "attributes" that can be used to give a quality profile for the system. For the spelling checker example, appropriate attributes might be:
– all mis-spelt words which do not correspond to a legal form of the language are detected
– the correct form of the mis-spelt word is among the corrections proposed.
Each attribute is associated with a value (eg. a percentage) which is arrived at via some method (eg. test suites).

The bottom up approach can be used for an application type (eg. spelling checkers, MT systems, etc) or for a specific system or component. If the aim is to write test suites for an application type, then the phenomena included will be of particular importance for that application. In the case of question answering systems, question types will predominate. The question of what constitutes relevant phenomena for MT systems has been addressed by various authors. King and Falkedal (1990) for example discuss how MT systems require "bilingual" test suites that probe the capacity of systems to deal with particular translation problems. Translation problems include for example the problem of lexical and structural mismatch, e.g. the classical *like– plaire* case, where the arguments of the verb are reversed in translation: *John likes Mary* translates as *Mary plaît à John*. In general, the construction of "bilingual" test suites requires some rather detailed insight into the nature of translation problems. Other "monolingual" test suites may be required for an MT system to test the monolingual components of an MT system. King and Falkedal (ibid) point out that in contrast to the "bilingual" test suites monolingual test suites should be translationally unproblematic, so that they do not introduce irrelevant difficulties.

For the construction of system-specific test suites, various options are available, depending on the type of evaluation that one is performing. Different evaluation scenarios can be distinguished: the so-called "black box" scenario, where the evaluator does not have access to the internal workings of the system, and the "glass box" scenario, where the evaluator does have access to the system rules. The former scenario is typically associated with the system user while the system developer is obviously associated with the latter. The test suite writer who is in a glass box situation has the option of tuning his test suite to the rules of his system, and the purpose of the evaluation will often be to test the reliability of these rules. The evaluator will thus be performing a diagnostic evaluation, to locate the source of errors of his system. Examples of test suites of this kind written for MT systems include, eg. Gambäck (1992), who describes the test suite he wrote to test the compositionality of a transfer-based MT system. The user, of course, might equally be interested in performing a diagnostic evaluation, despite having little or no access to the rules.

Even without direct access to the rules of a system, test suites can be written that try to "second guess" the rules, in order to test a hypothesis about the internal workings of the system. Douglas (1990) for example describes a test suite of this kind that she wrote for grammar checkers. To each error kind, that formed the basis of the test suite, she associated an indication of the type of underlying technology that she deemed necessary to deal with it. For example, some error types require full blown syntactic analysis rather than simple pattern matching. Conjoined NPs is an example, since a system that merely checked subject/verb agreement against the preceding NP in the sentence *John and Mary are* would mistakenly flag the sentence as ungrammatical, failing to realise that *John and Mary* is in fact the grammatical subject of the sentence and plural in number.

The user of an MT system may not be in an entirely black box scenario, having access to a system's lexical rules and possibly intermediate representations. An example of a test suite of this kind is described by Heid and Hildenbrand (1991), who wrote a test suite for the French to German module of the SYSTRAN MT system, given some informal information about the contents of the lexicon. The idea was to test whether verbs that were similarly categorised by the developer displayed the same behaviour when handled by the system. Knowledge about the system's lexical representations were thus used to guide test suite construction. In general, the more access an evaluator has to a system's internal workings, the more error diagnosis he can perform.

In addition to choosing whether or not to tune a test suite to a specific application or system, the evaluator also faces the choice of whether to tune it to a particular domain. This is an attractive option for the user in particular, who is interested to know how a system or systems perform(s) on his text types. An extreme example of this approach is represented by Lehrberger and Bourbeau (1988) who, in their guidelines for constructing test suites for MT systems from the user's point of view, propose that the test suite should consist of sentences derived from real corpora, and not be artificially derived. They add that these test sentences should be representative of the given domain, but offer no concrete definition of what "representative" means.

3.2 Top down approach

The top down approach to test suite construction starts from a list of linguistic phenomena, abstracted away from any particular application. Examples of this kind of test suite include the Hewlett Packard (HP) test suite (see Flickinger et al. 1987) and the DITO test suite (see Nerbonne et al. 1992). The HP test suite aims to provide coverage of a wide variety of syntactic phenomena, with some coverage of semantic phenomena and discourse phenomena as well. The DITO test suite concentrates on a smaller number of syntactic phenomena but covers them in greater depth. Both test suites are intended to be general purpose.

A feature of test suites of this kind seems to be a desire to draw the vocabulary from a general domain. DITO, for example, takes its vocabulary from personnel management since this domain is popular in natural language processing. This is in contrast to the bottom up approach, where the user at any rate is likely to tune his test suite to a more specialised domain.

A problem for this type of test suite is how to relate the phenomena they contain to the system or application that one is interested in testing. In addition, the phenomena, at least for adequacy testing, need to be related to frequency of occurrence in the text types one is interested in. Frequency information is provided in the HP test suite through the use of the markers "core" and "periphery", but these are likely to vary with text type. Note that weightings relating test suite input with frequency of phenomena in some text type are necessary for any test suite in order to make it a really useful evaluation tool.

3.3 The mixed approach

Regnier and Dauphin (1994) describe a methodology for test suite construction for MT systems that combines both the bottom up and the top down approach. The test suite is written from the user's point of view in a black box situation. A first round of testing identifies test sentences that are problematic for translation. These are then used as the basis for producing generic test sentences. The vocabulary in the generic test sentences is simplified and reduced. Individual phenomena are isolated and for each phenomenon a simple sentence is produced to illustrate the phenomenon. A whole series of test sentences is then written to illustrate all the different variations of the phenomenon.

This approach combines the advantages of the bottom up and the top down approach, in that the phenomena in the test suite are known to be problematic for a particular application, but are dealt with on a general level, that could potentially be used for a range of applications.

4 The TSNLP Project

TSNLP started in December 1993 and has a duration of 20 months. The partners involved are The University of Essex, UK who are the coordinators, plus Aerospatiale, France, Deutsches Forschungszentrum fuer Kuenstliche Intelligenz GmbH. (DFKI), Saarbruecken, Germany, and Istituto per gli Studii Semantici e Cognitivi (ISSCO), Geneva, Switzerland. Motivation for the project was the perceived lack of general guidelines for test suite construction, and adequate test material, with the consequent duplication of effort amongst many test suite writers and hence waste of time and resources.

The aims of the project are as follows:

- To define a set of guidelines for the construction of test suites for a range of NL products, including machine translation systems, concentrating on grammar checkers, parsers and controlled language checkers.
- To produce substantial test suite fragments covering core syntactic phenomena in three languages (English, French and German).

The project includes a testing phase for each of the three applications and revisions to the guidelines are foreseen in the light of test results. A subsidiary aim is to identify and develop a number of tools which will facilitate the construction and use of test suites, namely:

- A database in which the test suite will be stored which will allow easy access and manipulation of the data.
  We take as our starting point the database used by DITO (see Nerbonne et al. (ibid).
- An automatic test suite generation tool.
  Little previous work has been done on the automatic generation of test suites, but the endeavour seems worthwhile, given the labour-intensive and error-prone business of constructing test suites by hand. The project will take as a starting point work by Arnold et al. (1993b) on test suite generation.
- A lexical replacement tool.
  This will be helpful in the customisation that will be necessary to test system performance against a user's own corpora.

The project concentrates on producing test suites for parsers, grammar checkers and controlled language checkers. These applications were chosen, because TSNLP will concentrate on syntactic phenomena, and these applications have a large syntactic component. It is hoped, however, that the data will be useful to other application types which include a large syntactic component, which includes many MT systems. The fact that parallel data is being produced in three languages (English, French and German) means that it will be of interest to multilingual applications, including of course, MT. The results of the project, both scientific reports and actual test suites, will be in the public domain.

4.1 TSNLP methodology

As far as methodology is concerned, the TSNLP adopts the top down approach. However, the data will be annotated in such a way that information about centrality of the phenomena to a particular application, and frequency of phenomena in a particular text type can be provided. Thus subsets of data will be able to be extracted that are appropriate for certain applications and/or domains.

TSNLP is above all interested in producing a test suite that is flexible and reusable. TSNLP began by reviewing publicly available test suites, to see in what ways test suite design could be improved. The review revealed that the following characteristics are important for flexibility and reusability:

1. Systematic annotation scheme:
   An explicit characterisation of the test data, not merely section headings.
2. Support tools:
   Software tools to assist in the creation or use of test suites

3. Documentation:
   Documentation is useful on both the design and content of the test suite.

Few of the test suites we examined or which are reported on in the literature contain any or all of these characteristics. They are however, a key focus of TSNLP. The development of support tools (database, generation tool, and lexical replacement tool) have been discussed above. We say more about (1) and (3) below.

Systematic annotation scheme

Reusability of existing test suite is severely hampered by lack of annotations about their content. Frequently, only section headings are provided. An exception is the DITO test suite, that uses a very explicit annotation scheme, specifying, amongst other things, length of sentence, syntactic category and position of constituents, grammaticality status, and type of error. TSNLP will build upon the DITO annotations.

As mentioned above, systematicity is a useful property of test suites, yet is not always present in the test suites we examined. Several suggestions of how to achieve systematicity are discussed in the literature on test suites. The Neal Montgomery Method for example(see Neal et al. 1992) proposes that the evaluator progresses from very elementary sentence types containing simple constituents to more complex sentence (or paragraph) types. The idea is that each time a test sentence (or paragraph) is presented to the NLP system being evaluated, the sentence or paragraph should contain only one new (untested) linguistic capability or one new untested combination of tested capabilities. Gambäck (1992) is interested in the systematic production of combinations of phenomena, and proposes so called "compositionality" tables for this purpose. In DITO (ibid) systematicity extends to the creation of ill-formed examples. For verb subcategorisation, for example, three types of ill-formed data are derived for each example: (a) an obligatory argument is missing (b) there is an argument too many (c) one of the arguments has the wrong form. In TSNLP, as in DITO, the expectation is that systematicity can be improved by the use of a proper annotation scheme.

Documentation

The provision of detailed documentation about a test suite is also vital for its reusability. The user typically wants to know such things as the source of the data, the description of the phenomena (possibly with references), the methodology used (eg. how ill-formed data was derived and selected), etc. Few test suite builders have provided documentation in sufficient detail, but again there are exceptions, amongst them the Neal Montgomery Method (ibid) and DITO (ibid).

5 Conclusion

We have discussed the importance of test suites in evaluation and looked at the present state of test suite construction. We then discussed what improvements TSNLP is expected to bring to test suite design and construction. Additionally, it is expected that the availability of validated test data that is fully annotated and accessible, by means of the database, will be of benefit to developers and users of NLP products, even outside the applications for which the data is principally designed (i.e. grammar checkers, controlled language checkers and parsers). The multilingual nature of the project makes it of particular interest to MT, since it will allow the extraction of parallel data across different languages, which might be a good starting point for the construction of a true MT test suite.

REFERENCES

Arnold, D., Balkan, L., Humphreys, R. Lee, Meijer, S., and Sadler, L. (1993a): Machine Translation: An Introductory Guide, NCC Blackwell, Manchester, Oxford.

Arnold, D., Moffat, D., Sadler, L., Way, A., (1993b): "Automatic Test Suite Generation", in *Machine Translation*, Volume 8, nos 1-2, pp29-38.

Douglas, S. (1990): Intelligent Text Processing: A Survey of the Available Products, University of Edinburgh.

Flickinger, D., Nerbonne, J., Sag, I.A. and Wasow, T., (1987): "Toward Evaluation of NLP Systems", Hewlett-Packard Laboratories, distr. at the 24th Annual Meeting of the Association for Computational Linguistics (ACL), Stanford.

Gambäck, B., (1992): "Developer Oriented Evaluation: Two Experiments", talk presented at the *Workshop on the Strategic Role of Evaluation in Natural Language Processing and Speech Technology*, Record of the ESPRIT DANDI-ELSNET-HCRC Workshop, Edinburgh, 1992.

Heid, U., and Hildenbrand, E., (1991): "Some practical experience with the use of test suites for the evaluation of SYSTRAN" in the *Proceedings of the Evaluators' Forum*, Les Rasses, April 21-24 1991, available from ISSCO, University of Geneva, Switzerland.

Hoard, J., (1991): "Preliminaries to the Development of Evaluation Metrics for Natural Language Semantic and Pragmatic Analysis Systems", in Neal, J.G. and Walter, S.M. (eds), (1991): *Natural Language Processing Systems Evaluation Workshop*, Report RL-TR-91-362, Rome Laboratory.

King, M. and Falkedal, K., (1990): "Using Test Suites in the Evaluation of Machine Translation Systems", in *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, Helsinki.

Lehrberger, J. and Bourbeau, L., (1988): Machine Translation: Linguistic characteristics of MT systems and general methodology of evaluation, John Betjamins.

Nerbonne, J., Netter, K., Kader Diagne, A., Klein, J., and Dickman, L., (1992): "A Diagnostic Tool for German Syntax" Report DFKI D-92-03, Saarbrücken. Also in: Neal, J. and Walter, S. eds., 1991, *Natural Language Processing Systems Evaluation Workshop*, Berkeley, Rome Laboratory, Report RL-TR-91-362, New York.

Neal et al. (1992): An Evaluation Methodology for Natural Language Processing Systems, RL-TR-92-308, Final Technical Report, Rome Laboratory, Air Force Material Command, Griffiss Air Force Base, New York.

Pierce, J. R., and Carroll, J.B., (1966): Language and Machines – Computers in Translation and Linguistics (ALPAC Report), ALPAC, Washington D.C., 1966.

Regnier, S. and Dauphin, E. (1994): *Test Suite Design at Aerospatiale*, Input Paper for WP2, TSNLP.

Thompson, Henry, S. (1991): "Automatic Evaluation of Translation Quality: Outline of Methodology and Report on Pilot Experiment", *Proceedings of the Evaluators' Forum*, April 21-4, Less Rasses, Vaud, Switzerland, 215-224.

Thompson, H., (1992): "Parseval Workshop", ELSNews, Volume 1(2).