Test Suites for Evaluation in Natural Language Engineering

Lorna Balkan, Douglas Arnold and Frederik Fouvry
Dept. of Language & Linguistics,
University of Essex,
Wivenhoe Park,
Colchester, CO4 3SQ, UK
{balka,doug,fouvry}@essex.ac.uk

1 Introduction

In the Natural Language Engineering (NLE) context, a *test suite* is a more or less systematic collection of specially constructed linguistic examples (e.g. sentences) with annotations and other information. The idea is that such examples can be used for testing and evaluating various kinds of Natural Language Processing (NLP) systems. They are particularly useful for *progressive* evaluation (where the idea is to make sure that changes to a system under development represent improvements, rather than the reverse), and for *diagnostic* evaluation (where the idea is to locate and isolate particular errors). This means they are particularly useful for system developers. But they also have wider application, and have long been accepted as useful tools for evaluation in NLE and NLP generally. Unfortunately, most existing test suites tend to be relatively unsystematic and lack generality (having been constructed with particular systems in mind). Moreover, there is no established methodology which a system developer or other evaluator can follow in constructing a test suite of their own.

The TSNLP project[1] (Test Suites for Natural Language Processing) seeks to address these problems. In particular, the project has tried to produce realistic and general *guidelines for test suite construction*, and to construct *substantial amounts of test data* in three languages (English, French and German). The bulk of the data (several thousand test items) covers "core" syntactic phenomena and is 'general purpose' in the sense of being suitable for testing any system where syntactic processing plays a non-trivial role. In addition, some effort has been invested in producing application specific data (for parsers, grammar checkers and controlled language checkers), and in validating both the data and the general approach by performing experimental tests on a number of NLP products. The data has been mounted onto a

[1]The TSNLP project is funded by the CEC and started on 1 December 1993 with a duration of 23 months. We would like to thank our partners in the project: Dominique Estival, Kirsten Falkedal and Sabine Lehmann at ISSCO, Geneva, Eva Dauphin and Veronika Lux and Sylvie Regnier-Prost at Aerospatiale, Paris, and Judith Klein, Klaus Netter and Stephan Oepen at DFKI in Saarbrücken, Germany.

database system, for ease of access and manipulation (for further details see [12]).

All the results of the project, including actual test suites are, or will be, in the public domain. Details of how they can be obtained are given below (section 4).

The following section looks at the role of test suites in evaluation, and is followed by a discussion of the design issues that have informed the TSNLP project.

2 Evaluation

Traditionally there are two main ways of evaluating NLP systems, either by the use of test suites or by the use of test corpora (i.e. collections of naturally occurring texts). Test suites and test corpora have different roles to play in evaluation and should be seen as complementary rather than competing techniques.

The strength of test corpora is that they represent "real" data, that is naturally occurring data, so if one is interested in seeing how one's system performs on real life text, the test corpus method is preferable. However, for diagnostic purposes, it is important to isolate the exact phenomena or combinations of phenomena that are problematic, and this is difficult to do with corpora, due to the complexity of naturally occurring data, and the lack of tools for locating syntactic constructions in text. Furthermore, it is unlikely that one will be able to find all the systematic variations of a particular phenomenon or phenomena in a particular corpus, given that they occur only accidentally. This is true not only of positive (well-formed) data, but also of negative (ill-formed) data. The inclusion of ill-formed test items is crucial for some applications (e.g. grammar checkers and controlled language checkers which are designed to recognise and mark ill-formed constructions). However, unlike arbitrary combinations of phenomena, which can be tested using the test corpus method, arbitrary error types cannot be so checked, due to the relative scarcity of error corpora.

Test suites, by contrast, allow one to test phenomena and combinations of phenomena in a controlled fashion. One can systematically generate all the parametric variations of a phenomenon and any combination of phenomena one is interested in. Equally, one can generate ill-formed data by systematically varying the parameters or constraints associated with the well-formed items, such that the ill-formed examples fall out automatically. As an example, consider NP agreement for determiners and nouns in English, which show the following paradigm:

* much firm	Det + Singular count noun
* much firms	Det + Plural count noun
much money	Det + Non-count noun

Note that in test suites the vocabulary, as well as the sort of construction being tested, can be controlled. This allows the evaluator to focus on the way the system deals with the construction without the distraction of problems relating to lexical coverage. And an added advantage of the test suite method is that they avoid the problem of redundancy inherent in corpora, where multiple instances of particular phenomena occur.

The test suite method is particularly useful for testing syntactic phenomena where the range of phenomena is relatively well-understood and well-documented. Semantic and pragmatic phenomena are less accessible to the test suite method, since the phenomena are less easy to characterise, and are frequently context-dependent. This means that many phenomena, such as anaphora resolution need 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. The TSNLP project concentrates on the production of syntactically based test items.[2]

It is also the case that some applications are less well suited to being tested by the test suite method than by test corpora. Message understanding systems, for example, need whole sequences of sentences as input, so are better suited to the test corpora method. Test suites are useful for any system which has a large syntactic analysis component. Furthermore, they are best suited to applications where it is possible to specify not just the nature of the input, but also the nature of the output, as in grammar checkers for example. (Generation systems, on the other hand, are less well-suited to this method, since it is difficult in general to specify not only what the input to a generation system should be, but also what constitutes an appropriate output). As noted above, the bulk of TSNLP data is intended for any syntactically-based system. Some effort has also gone into the design and construction of system- specific data for parsers, grammar checkers and the controlled language checker SECC.

We noted above that test suites are particularly useful for *diagnostic* and *progressive* evaluation, as defined above, following EAGLES (see [7]) terminology. They can also be used for *adequacy* evaluation, which, again using EAGLES terminology, aims to determine whether and to what extent a particular system meets some pre-specified requirements. Developers are chiefly interested in diagnostic and progress evaluation, while users are mainly interested in adequacy evaluation. However, if developers aim eventually to market their products then adequacy evaluation is an issue for them too.

[2] An exception is the test data written for the controlled language checker SECC [1], where many of the phenomena are lexically based.

Likewise, if a user wants to know not just how a product behaves today, but in its future potential, then they will be interested in performing a diagnostic evaluation. A diagnostic evaluation for the developer will, however, differ from that of the user in that the user is typically in a black box situation with respect to the system (i.e. he does not have access to its internal workings), while a developer will be in a glass box situation, where he will have access to the system rules.

In summary, test suites are a useful tool for anyone who wishes to test, benchmark or evaluate systems with grammar components.

3 Methodology: How to construct and use a test suite

Given how useful test suites can be, one might expect there would be a well tried and widely understood methodology for their use and construction. However, this is not the case. Attempting to supply such a methodology was accordingly a key aim of TSNLP. Space does not permit a detailed discussion (cf. [4]), but a number of points are worth making.

First, there are design issues relevant to the construction of any test suite. As noted above, the strength of test suites is that they allow some control over what is being tested. In TSNLP, we attempt to use a limited vocabulary, and avoid categorial and semantic ambiguity where possible. This is to avoid lexical interference where lexical phenomena are not being tested. We have also made a conscious effort to expand closed class items where appropriate (e.g. expand the set of modal auxiliaries in our modality tense and aspect data).

We said above that test suites allow one to isolate phenomena for testing, but due to the nature of language itself, sentences seldom if ever contain only one phenomenon. What one can do is try to keep constant the parts of the sentence not being tested for, by for example, ensuring that there is correct subject verb agreement in sentences not specifically testing subject verb agreement. Sentences are also kept as short and simple as possible, by, for example, using declarative sentences in the present tense and avoiding modifiers and adjuncts.

Where possible, the parameters of a phenomenon are identified and varied systematically to produce the space of well-formed and ill-formed items for that phenomenon. We illustrated this with the NP agreement example above, where the relevant parameters are the number and semantic type (count or non-count) of the noun the determiner cooccurs with. An automatic test suite generation tool was developed in the course of the project ([2]) which was used to increase systematicity in this respect. The tool is based on a simple DCG grammar, and in its current form, produces only the sentences or sentence fragments plus an output (a parse tree) and simple annotations, but the tool could be enhanced to generate more annotations automatically and an interface written to convert its output into database format.

Other ways of automating the construction of test suites have been explored within the project, including a tool (TSCT, which is described in Oepen *et al.* [11]) for inputting data, and a lexical replacement tool ([2]).

The second set of design issues is associated with the nature of TSNLP itself. One of the main characteristics of TSNLP is that it is a multilingual project that aims to produce parallel test data in three languages: English, French and German. By *parallel* we mean that the data will cover the same set of phenomena, and use the same annotation scheme, with language-specific variations where appropriate. The annotation scheme will be described in greater detail below. The data is intended to be multi-user. By that we mean that, although, as we noted above, test suites are primarily aimed at system developers, we hope that the extensive annotations of the data will render them useful for system users too. The data is also supported by extensive documentation which is meant to be accessible to the non-specialist.

From the above, it should be clear that a key point in our methodology was the devising of a set of annotations that serve not only to clarify for the user what is being tested by a test item, but also serve to ensure a systematic approach to test suite writing. A review of existing publicly available test suites carried out within TSNLP ([8]) revealed that there was a tendency for test suites in general to be under-annotated (with the exception of DITO [10]), which causes problems for their reusability. One aim of the TSNLP data is that it should be maximally reusable. TSNLP has built upon the DITO annotations[3]. TSNLP annotations include information about the actual string of words, its length, category, functional analysis and well-formedness, as well as giving a hierarchical classification of the phenomenon and a list of the phenomena presupposed to be correct for that test item, which are not explicitly being tested for, and information about the source and type of the error for ill-formed items (see Figure 1). A common naming system for category and function names as well as high level phenomena names is adopted to ease cross-reference across languages. Annotations are also provided for the relevance of a test item for a particular application type and its frequency and weighting for a particular system and text type. The determination of these last values falls outside the scope of the current project and is an area for further research.

Of course, devising methodologies and guidelines in the abstract is not enough: the proof of a methodology is in its application in realistic situations. For this reason, the project includes a "validation" phase, now in progress, where

[3]Full details of the TSNLP annotation scheme can be found in [9].

Phenomenon supertype	NP_agreement				
Phenomenon name	Postdeterminer+Noun				
Presuppositions	Word order is correct				
Id	362611				
Category	NP				
Input	*much firm*				
Length	2				
WF	0				
Error type	*much* cooccurs with mass nouns				
Analysis	position	instance	category	function	functional domain
	0:1	*much*	DET	spec	1:2
	1:2	*firm*	N_sg	func	0:2

Figure 1: Some TSNLP annotations, presented in readable form. Note that the phenomenon name is simplified.

a number of NLP systems are being tested (the issue being, of course, not how well the system behaves, but how easy it is to test them).

4 Results

The results of the project, including test data and reports, will become public domain on completion of the project in October 1995. Intermediate reports to date can be obtained from `http://clwww.essex.ac.uk/Group/Projects/tsnlp/` and from the coordinator at `balka@essex.ac.uk`, and are as follows:

- D-WP1 Analysis of Existing Test Suites [8]
- D-WP2.1 (part1) Test Suite Design – Guidelines and Methodology [4]
- D-WP2.1 (part2) Issues in Test Suite Design [5]
- D-WP2.2. Test Suite Design – Annotation Scheme [9]
- D-WP5.1 Design and Implementation of Test Suite Tools [2]
- D-WP5.2 Corpus-based test suite generation [6]

We said at the outset that the current situation as regards availability of test suites for NLE was not good. The TSNLP project has gone some way to remedying the situation, providing substantial test fragments in three languages, with supporting guidelines and documentation. However, there are a number of needs that should still be addressed. As noted above, the assignment of weightings to test items requires corpus analysis and is a topic for further research. The type of ill-formed item found in the test suite is limited at present to examples derived from well-formed items where normal grammatical feature restrictions have been violated. A topic for future research is the inclusion of "performance" errors, ie. errors that occur as a result of text manipulation, treatment of formatted structures, punctuation, etc. The coverage of performance errors presupposes the availability of some type of error corpora, which are not easily obtainable.

References

[1] Adriaens, G. (1994). *SECC: Simplified English Grammar and Style Correction in an MT Framework*, Proceedings of the Linguistic Engineering Conference, Paris 1994, ELSNET, Edinburgh.

[2] Arnold, D., Rondell, M., and Fouvry, F. (1994). *Design and Implementation of Test Suite Tools*, Report to LRE 62-089 (D-WP5.1), University of Essex.

[3] Arnold, D., Balkan, L., Fouvry, F., Dauphin, E., Lux, V., Regnier-Prost, S., Klein, J., Netter, K., Oepen, S., Estival, D., Falkedal, K., and Lehmann, S. (1995). *Checking against Corpora*, Report to LRE 62-089 (D-WP3.2).

[4] Balkan, L., Meijer, S. *et al.* (1994). *Test Suite Design — Guidelines and Methodology*, Report to LRE 62-089 (D-WP2.1a), University of Essex.

[5] Balkan, L., Meijer, S. *et al.* (1994). *Issues in Test Suite Design*, Report to LRE 62-089 (D-WP2.1b), University of Essex.

[6] Balkan, L., Fouvry, F.*et al.* (1995). *Corpus-based test suite generation*, Report to LRE 62-089 (D-WP5.2), University of Essex.

[7] EAGLES Evaluation Subgroup. (1994): *Evaluation of Natural Language Processing Systems (Draft)*, EAG-EWG-PR.2, Centre for Language Technology, Copenhagen, Denmark.

[8] Estival, D., Falkedal, K. *et al.* (1994). *Analysis of Existing Test Suites*, Report to LRE 62-089 (D-WP1), University of Essex.

[9] Estival, D., Falkedal, K. *et al.* (1994). *Test Suite Design — Annotation Scheme*, Report to LRE 62-089 (D-WP2.2), University of Essex.

[10] Nerbonne, J., Netter, K., Kader Diagne, A. Klein, J, and Dickman, L. *A Diagnostic Tool for German Syntax.* DFKI D-92-03, Saarbrücken. Also in: Neal, J. and Walter, S. eds. (1991): *Natural Language Processing*

Systems Evaluation Workshop, Berkeley. Rome Laboratory, RL-TR-91-362, New York

[11] Oepen, S. and Netter, K. *TSNLP – Test Suites for Natural Language Processing*, to appear in the proceedings of the Linguistic Databases workshop, Groningen 1995.

[12] Oepen, S., Netter, K., Baur, J., Fettig, T., Klein, J., and Oberhauser, F. *The TSNLP Database – From tsct to tsdb –* , Report to LRE 62-089 (D-WP6.1), Deutsches Forschungszentrum fuer Kuenstliche Intelligenz GmbH.