Test suites for Controlled Language Checkers

Frederik Fouvry Lorna Balkan

{fouvry,balka}@essex.ac.uk
CL/MT Group, University of Essex
Wivenhoe Park, Colchester
Essex CO4 3SQ, UK

Abstract

TSNLP produced guidelines and methodology for test suite writing, as well as a substantial amount of annotated test data. To test the guidelines, the created data, and the methodology for controlled language test suites, they were used to test the SECC controlled language checker. We report here on what test suites for controlled language should contain, how we realised the actual test suite, and on the conclusions we could draw from all this.

1 TSNLP

Test Suites for Natural Language Processing (TSNLP – LRE 62-089)[1] aimed at developing guidelines and methodologies for general purpose test suite construction, and constructing a substantial amount of test data for three languages (over 4500 items for each of the languages German, French and English, with documentation). In order for those to be reusable, TSNLP developed an extensive annotation scheme, and a test suite database `tsdb`, that provides flexible and easy access to the test suite.[2]

[1]The project covered the period December 1993–March 1996. We acknowledge our partners: Dominique Estival, Sabine Lehmann, Kirsten Falkedal (ISSCO, Genève), Sylvie Regnier-Prost, Eva Dauphin, Véronika Lux (Aerospatiale, Paris), Klaus Netter, Judith Klein, Stephan Oepen (DFKI, Saarbrücken), and our colleagues at Essex, Evangelia Kordoni and especially Siety Meijer who worked on the first half of the project and was an important contributor for the material we present here. We also thank Lieve Macken and the other members of the SECC-project.

[2]The guidelines and methodology are discussed in [4, 3, 5, 12]. The data construction and the documentation is discussed in [7, 14]. The data base and other tools is the subject of [9, 13]. The annotation scheme is treated in [6, 13] (but it is also discussed in [7, 9, 12, 14], viz. how we used it). The deliverables, user manuals, data and tools are available from http://tsnlp.dfki.uni-sb.de/tsnlp/ or from the project coordinator L. Balkan, CL/MT Group, Dept. Language and Linguistics, University of Essex, Wivenhoe Park, Colchester, Essex CO4 3SQ, UK.

A test suite (in NLP) is a specially designed collection of sentences and sentence fragments used to test an NLP application: language in lab conditions. The difference between test suites and test corpora is that the former are very controlled and systematic. Corpora are real life texts, *i.e.* linguistically convoluted. This is most obvious in syntax and lexicon. Grammatical phenomena interact and occur *en masse*. This makes diagnostics and precise measurement of progress difficult. Test suites seem to be most appropriate to help and perform diagnostic evaluation,[3] because they focus on specific aspects.

The motivation for TSNLP was that although evaluation is steadily becoming more important, there is little test suite material publicly available ([5]). Also, since many test suites cover the same phenomena, effort is often (needlessly) duplicated. It also drew up guidelines and produced customised data for three applications (parsers, grammar checkers and controlled language checkers), to be tested on a real system.

The project developed an annotation scheme on the basis of research into the properties of NLP applications.[4] The test suite database `tsdb` is an implementation of the annotations scheme. It can be extended for system specific information.

At the University of Essex, we customised the English test suite for the Simplified English Controlled Language Checker (SECC, LRE 62-064) ([2, 8]).[5]

2 Test suites for controlled language checkers

2.1 A special case

Controlled language checkers have a somewhat special status as an NLP application from the point of view of testing, for several reasons. They are complex systems, (usually) containing parsers and (grammar) checkers, and bring in their own extra limitations and characteristics, which are a subset of the natural language. They should be able to handle correct input, as any other NLP application, but they should be more robust than other applica-

[3]The EAGLES Evaluation Working Group made a distinction between several different kinds of evaluation. There is diagnostic evaluation ("Where are the errors ?") progress evaluation ("What changed (for the better) ?") and adequacy evaluation ("Is this fit for what we want to use it for ?") ([3, p.25ff.]).

[4]In short, the annotations consist of the test string, wellformedness, length, errors, grammatical phenomena, comments, presuppositions that need to be fulfilled, related strings, etc.

[5]The other applications tested were the grammar checker for French *Le Correcteur 101* (Aerospatiale) and the parser developed by DFKI in their *HPSG system* (with a German grammar). See [10].

tions: they must at least detect the errors, and preferably they should also correct them, or give suggestions for corrections. All these aspects must be taken into account for testing.

In order to assess controlled language checkers it is important to establish which tasks the system performs. We can identify the following:

1. Assignment of correct (partial) parse to the input string
2. Identification of errors against the controlled language, *i.e.* errors against vocabulary, grammar and style restrictions.
3. Identification of errors against grammar.[6]
4. Suggestions for corrections (this is not done by all systems).

Test suites can be written for any phenomenon in the controlled language (from lexical level up to the document level): lexical, morphological, syntactic, semantic stylistic, formatting, document (content) construction. But they are best suited for syntactic phenomena.

We focused on syntax because it was the general focus of the project, but while creating the test suite for SECC, we also added some test items for semantics, lexicon and punctuation where these phenomena interact with syntax.

2.2 Scope of Test Suites

As controlled languages do not aim to deal with all types of input, users, etc., this should be taken into account while writing test suites. Information like this is system specific, so much work will be needed to create the test suite. The most difficult point however remains the controlled language itself: there is no general controlled language, and the grammars will be different as a consequence. So will the test suites.

System Tasks

The scope of test suites for controlled language checkers can be said to be a system's definition of what is ungrammatical and therefore must be corrected.

The description and classification of the errors they have to identify form a vast and far less explored area than the correct language use.

Additionally, users performing an adequacy evaluation will not only want to know whether a controlled language checker identifies errors correctly and consistently, they will also want to assess the adequacy of the corrections proposed and/or the explanation and advice given by the system. This is,

[6]It is not clear whether all controlled language checkers do this.

however, mainly an issue pertaining to the (subjective) interpretation given to the results of applying a test suite and adds no extra requirements on the design of the test suite. Provided the test suite has a reasonable coverage of phenomena, the results of applying it will contain representative examples of the types of corrections and advice suggested by the system.

If some set of ill-formed inputs will thus give the necessary evidence for an assessment of the adequacy of system response, the desire to test the consistency of this response requires the construction of a series of test items where the same error is embedded in different contexts.

2.3 Requirements for Test Suites for Controlled Language Checkers

As previously mentioned (p.2), parsers and grammar checker functionalities form an important part of a controlled language checker. Therefore test suites should also address these parts explicitly, even if it is not possible to isolate the components in use.

For testing the "parser" part, the test suites are easy to write: take a test suite that covers the data your system should cover and adapt the lexicon. This does not hold however for output annotations when you are not able to access the parser results. If some intermediate processing changes the parser output in a unrecognisable way, then the interpretation of the output annotations is not as clear-cut any more. The tests are therefore not without some use. The result will be cluttered with other information, and interpretation of test results must proceed more carefully.

2.3.1 Coverage

Lexical coverage must be based on the system lexicon (and appropriate morphology). Ambiguities in controlled language are more problematic than in "normal language": they must be more strictly avoided, but this is close to impossible. So the test suites should specifically test for ambiguity.

Controlled language checkers should detect errors against the controlled language as well as against the full-size natural language, and preferably correct them, or suggest a correction. Moreover, the should not report controlled language errors, nor any natural language errors, nor error reports where there are no errors in the input. The test suite for a controlled language checker should classify items according to the domain they belong to: if a sentence is correct, but not for the controlled language, then the checker should notice this. All the items belonging together should be linked up.

The ill-formed input (robustness and error detection/correction) preferably has a certain degree of probability: it is more useful for the final assessment. This is an issue where the users want to put the stress, more than developers.

Simple items are needed in the test suite, but for a full evaluation, these necessarily have to be complemented with more complex items: how does the system behave when the phenomena start to interact ? Besides, the controlled language checker will be used on real, *i.e.* linguistically complex, input.

2.3.2 Overview of required Test Data

To start with, we can identify several requirements for test suites for controlled language checkers. Firstly, as we will see below, the number of possible interactions of phenomena in test items is much larger than for a test suite intended for use with other applications like a parser: restrictive rules on vocabulary, style and grammar interact with the usual grammar phenomena. The testing of interaction is necessary, but restrictions need to be imposed. Secondly, a test suite for controlled language checkers requires test items that allow for testing sentence length and style rules on paragraphs. Thirdly, testing of ill-formed input (either grammatically or according to the controlled language) and their correct counterparts is of utmost importance. This is taken into account in the description of test material given below.

As for the different tasks within the controlled language checker, the following list describes the type of test material required.

Controlled Language Vocabulary: The test suite should contain items which will allow us to test the following vocabulary rules:

1. Lexical rules that specify which words are disallowed and which suggest which word to use instead.
2. Lexical rules which include a specification indicating under which circumstances a certain word is allowed, depending on the *category* or *meaning* of the word.

The suite should not attempt to test all words disallowed by the checker, as there is an indefinite number of them and this falls outside the scope of the test suite technique. Test items containing user or company specific terminology must be annotated clearly for this information.

Controlled Language Grammar Rules: These should be tested by the following types of test items:

1. Test items which are grammatically correct and which comply with the controlled language rules (also with regard to vocabulary and style). For this an adapted test suite for parsers can be used.

2. Test items which are grammatically correct but which violate the controlled language grammar rules. As this primarily aims at finding controlled language violations, the important requirement is to include one (or more if interaction is tested) controlled language violation in each test item. If each violation is tested in combination with different grammatical phenomena, a combinatorial explosion would take place. The desired approach is probably to test all violations systematically while ensuring some variation in grammatical phenomena.
3. Test items which are grammatically incorrect but which do not violate the controlled language rules.
4. Test items which are grammatically incorrect and which violate the controlled language grammar rules.

Controlled Language Style Rules: These can only be tested to some extent, depending on the length and amount of test material required (for some rules this may be several paragraphs).

Annotations for Expected Output

The output annotations must indicate how the system under test is expected to perform, when given some input. In the case of a controlled language checker, the output of the system consists in:

- the recognition of the exact error in the test input (or no error flagging, if no error is found in the input);
- suggested correction(s) to the user if an error is found;
- possibly, some advice generalizing over the type of grammatical error contained in the input.

The output annotations should reflect this behaviour in the following way:

- For well-formed input, no output specification should be given, except, maybe some informal statement saying that no system response is expected;
- For ill-formed test items, three kinds of indications should be given as far as possible:
  - the lexical material the system should respond to;
  - a description of, or a name for, the error;

– the correct sentence or construction; where several correct versions are possible, the list of these.

The output annotations should be convertible to the system output. The `tsdb` database brings us closer to this goal.

2.4 Errors

Identifying controlled language Errors

When evaluating the system's capability to identify errors we have to assess the flagging of errors by so-called error critiques. What needs to be established during assessment is:

- whether any critiques are incorrectly reported (these are called *spurious critiques*);
- whether the system failed to identify errors (*missed critiques*).[7]

Another aspect that needs to be taken into account are those error messages which indicate something is wrong with a sentence, but cannot indicate what exactly the problem is. We will call these *vague critiques*.

The errors against controlled language restrictions can be subdivided according to the three different types of restrictions which are generally imposed by controlled languages (see e.g. [11]):

1. Restrictions on vocabulary
2. Restrictions on syntax
3. Restrictions on style

We will look at these restrictions one by one.

Vocabulary The assessment of this task of the controlled language checker is certainly important, but possibly the test suite method is not the appropriate method, as mentioned above.

Syntax To identify errors against the restrictions on syntax the test suite method can usefully be applied. Four types of test items need to be included:

1. Test items which are grammatically correct and which comply with the controlled language rules (also with regard to vocabulary and style).

[7] Both these terms were taken from [15].

2. Test items which are grammatically correct but which violate the controlled language grammar rules.
3. Test items which are grammatically incorrect but which do not violate the controlled language rules (this in fact is the task of the grammar checker).
4. Test items which are grammatically incorrect and which violate the controlled language grammar rules.

Style Although controlled language checkers will attempt to identify errors in style there are some major problems with this subtask. The rules in this task are very diverse. There are rules that can be tested in a very simple way with a few test items (e.g. restrictions on the sentence length). Other rules are difficult to implement, but are easy to write test items for, e.g. *Write only one instruction per test sentence.* (from [1]). Much more difficult to achieve are rules like lay-out rules, e.g. *If you want to show that some steps are closely related, use the tabular layout (vertical layout) for the text.* Some rules require a large piece of text, like ([11]) *Do not use one-sentence paragraphs more than once in every 10 paragraphs.* In these cases, it is probably better to start from a corpus.

Error Correction We can distinguish between three types of controlled language checkers:

1. Checkers which flag mistakes but do not make any suggestions for error correction.
2. Checkers which flag mistakes and make suggestions for error correction.
3. Checkers which flag mistakes, make suggestions for error correction and actually perform some amount of automatic correction in case error detection is fairly straightforward.

Most controlled language checkers will attempt to make suggestions for error correction rather than just flag the mistakes, but the quality of these suggestions often leaves much to be desired. Though a certain number of them may be right, it is not uncommon for systems to come up with suggestions which are too vague or plainly wrong.

Some more advanced checkers attempt a degree of automatic error correction.

If we distinguish between lexical, syntactic and stylistic error correction it is clear that the first type can be achieved quite easily. The second and third type are much more complicated and automatic correction is in many cases not possible. For some of these rules it might be possible to suggest a correction, but more research results are required in order to predict what one could reasonably expect from a controlled language checker which performs correction.

If we want to use test suites for those controlled language checkers that make suggestions for error correction or even do some automatic correction, it is desirable to reuse the test material which we used to test the error identification task of the controlled language- and grammar checker.

3 What has TSNLP done ?

The aim of the TSNLP testing round was twofold. For TSNLP, there was the need to test the annotation scheme and methodology. For SECC, the aim was to use the test suite to improve the system.

We proceeded in the following way:

New items were created. It was clear when we looked at the SECC grammar that TSNLP general purpose data do not cover all their phenomenon description: we did not have the intention to cover non-syntactic phenomena in English,[8] and Cogram, the SECC grammar, has rules of its own that lie outside the scope of TSNLP. These items did not get the full TSNLP annotations, but only output annotations, which were required for the interpretation by the SECC developers.

On the other hand, it was also our intention to customise existing general purpose data, to find out more about the customisation process. The main work consisted of adapting existing data to conform with the input that was required for the system:

a) lexical (it is a controlled language) - this was relatively easy, since the lexicon in the general purpose test suite was kept quite small. More lexical information though in the annotations, would have made it possible to automate this process better.

b) system specific requirements (like ID format)

c) the controlled language-grammar is more limited, and uses a different classification scheme. The tester needs a mapping between these classifications and this will remain a frequently occurring problem, although the classification of TSNLP is rather descriptive and fine-grained. If the grammar of a system tested merges several TSNLP phenomena, or split a phenomenon and add the two parts to other phenomena, it becomes more difficult to extract the data. A study of the “mappings”

[8] Apart from parentheses and abbreviations.

can prove very useful. Furthermore, tsdb, the test suite database developed by TSNLP ([9]), allows one to record this information, so that this information does not get lost for future use.

Additionally, output annotations were added for these data.

As the controlled language restrictions are designed for the *writers* of controlled language-texts, they are more difficult to test than other applications: the test suite data writers also have to comply with the controlled language rules, where they are not testing for them. It took a time to familiarise ourselves with the controlled language.

4 Conclusion

Test suites are a very useful device to evaluate and assess a controlled language checker. They help to point out vague and incomplete rules, wrong output (what is expected in the first place from a testing round), and other errors. Because the TSNLP test suites are so systematic and broad in their coverage, they tell us about the grammar specification and implementation of system (coverage, correctness and coherence) and about the system performance. The TSNLP annotation scheme proved to be fit for customisation and for addition of system specific extensions. The methodology and the guidelines also have shown their value. The results of TSNLP (data, documentation, guidelines, methodologies and tools) are poised to become the basis for a standard benchmark in the NLP community.

The TSNLP tests showed that customisation is a major task, that has to be planned in accordance to what one wants to get out of the tests.[9] The creation of new items is another way to go, but one should take care to try to make them reusable. This mainly means that the items need to be well annotated. Reusability allows one to compare tests and to check the quality of the data, and also avoid duplication of the effort of writing test data. It is an aspect of the test suite that is going to be appreciated more and more as the pool of items grows.

The more extensive the annotations are, the easier the customisation can be, certainly with the help of tools like a lexical replacement tool or a simple parser, to help creating the annotations (semi-)automatically. Such tools will take away the burden from the data writers, but will also improve the consistency in the test suite.

[9] At the time of the tests, the database was not available yet, which posed restrictions on the flexibility and ease with which the data could be accessed.

References

[1] AECMA/AIA Simplified English.

[2] Lorna Balkan, Frederik Fouvry, and Evangelia Kordoni. Report on customisation and testing of test suites for SECC. Report to LRE 62-089 D-WP4.2&3A, University of Essex, UK, August 1995. Confidential document.

[3] Lorna Balkan, Siety Meijer, Kirsten Falkedal *et al.*. Issues in test suite design. Report to LRE 62-089 D-WP2.1, University of Essex, UK, September 1994.

[4] Lorna Balkan, Siety Meijer, Kirsten Falkedal *et al.*. Test suite design. guidelines and methodology. Report to LRE 62-089 D-WP2.1, University of Essex, UK, September 1994.

[5] Dominique Estival, Kirsten Falkedal, Lorna Balkanl *et al.*. Analysis of existing test suites. Report to LRE 62-089 D-WP1, University of Essex, UK, March 1994.

[6] Dominique Estival, Kirsten Falkedal, Sabine Lehmann *et al.*. Test suite design annotation scheme. Report to LRE 62-089 D-WP2.2, University of Essex, UK, September 1994.

[7] Dominique Estival, Sabine Lehmann, Kirsten Falkedal *et al.*. The construction of test material. Report to LRE 62-089 D-WP3.1, University of Essex, UK, December 1995.

[8] Lieve Macken. Testing of the TSNLP test suites. Report to LRE 62-064 WP6A1, University of Leuven BE, August 1995. Confidential document.

[9] Stephan Oepen, Klaus Netter, Judith Baur *et al.*. The TSNLP database. from tsct to tsdb. Report to LRE 62089 D-WP6.1, University of Essex, UK, June 1995.

[10] Sylvie Regnier-Prost, Veronika Lux, Eva Dauphin *et al.*. Testing and customisation of test items. Report to LRE 62-089 D-WP4.2&3, University of Essex, UK, December 1995.

[11] Dirk Schreurs and Geert Adriaens. *Controlled English (CE): From COGRAM to ALCOGRAM*, pages 206–221. Oxford, England – Dordrecht – Boston – London, 1992.

[12] TSNLP. University of Essex, ISSCO, Aerospatiale, DFKI. *User Manual. Volume 1: Background, Methodology and Guidelines*, Fortcoming 1996.

[13] TSNLP. University of Essex, ISSCO, Aerospatiale, DFKI. *User Manual. Volume 2: Test Suite Technology*, Forthcoming 1996.

[14] TSNLP. University of Essex, ISSCO, Aerospatiale, DFKI. *User Manual. Volume 3: Test Data Documentation*, January 1996.

[15] Richard H. Wojcik, Philip Harrison, and John Bremer. Using bracketed parses to evaluate a grammar checking application. In *ACL 1993*. Association for Computational Linguistics, 1993.