# Test Suites for NLP

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

## 1 Introduction

In recent years there has been a drive to produce better resources for the evaluation of NLP systems. The EU funded LRE project TSNLP (Test Suites for Natural Language Processing) should be seen in this context.[1] By *test suite* is meant a more or less systematic collection of specially constructed linguistic expressions (*test items*, e.g. sentences), perhaps with associated annotations and descriptions. The project was motivated by the fact that test suites have long been accepted in the NLP community as a useful tool for diagnostic evaluation, and by the fact that there are few publicly available test suites. Moreover, most of the existing test suites have been written for specific systems or simply contain numbers of "interesting examples". This does not meet the demand for large, systematic, well documented and annotated collections of linguistic material, required by a growing number of NLP applications. A further problem for the prospective test suite user is that there are few guidelines for the construction of test suites. The TSNLP project thus has two principle aims: (1) to provide guidelines for test suite construction and (2) to construct substantial test fragments. This paper discusses problems associated with both the design and construction of test material, issues such as the identification of linguistic phenomena and sub-phenomena to be tested, the choice of the vocabulary used in test data, how one can derive ill-formed data and to what extent the interaction between different phenomena should be tested. We also look at some of the methods developed within the project for automating test suite construction, namely a test suite generation tool and an automatic lexicon replacement tool.

[1] We would like to thank our partners in the TSNLP 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.

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 also intended to be multipurpose, in that it aims to be relevant to any strongly syntax-based system, consisting as it does of an inventory of syntactic constructions. Special consideration has been given to the formulation of guidelines for the construction of test material for three syntax-based applications in particular, namely parsers, grammar checkers and controlled language checkers (against which the data will also be tested), but the data construction has concentrated on core, non-application specific data. Finally, 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 should thus support different 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. The TSNLP is thus designed to be maximally reusable. In order to enhance its usability, the TSNLP data is to be mounted on a database for ease of access and manipulation. Work on the database design is described in Oepen *et al.* ([12]).

# 2 Choice of Phenomena

There are two main approaches to test suite construction: the "bottom up" and the "top down" 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 (the EU-sponsored Expert Advisory Group on Language Engineering Standards [6]) which advocates that a system be analysed in terms of its functions. By way of example, the functions of a grammar checker might be (taken from Wojcik *et al.* [13])

1. Assign correct (maybe partial) structure (through parsing or pattern matching) to the input string;
2. Identify errors against the grammar;
3. Report errors against the grammar;
4. Make suggestions for corrections.

To this could be added a 5th, negatively stated task: not to report an error where none exists.

In the bottom up approach test data must be designed to test the particular functionalities of the system. For grammar checkers, for example, one might want to include include so-called 'traps' where patterns which are relatively easy to check in simple constructions (e.g. subject - verb agreement when the verb follows immediately after the head noun of the subject) are deliberately complicated, e.g. by extending the subject with a prepositional complement whose head noun does not agree with the verb in number. For example:

1. *The cake is on the table.*
   (simple test item for subject-verb agreement)

2. *The cake for the children is on the table.*
   (positive trap for subject-verb agreement)

3. *The cakes for the child is on the table.*
   (negative trap for subject-verb agreement)

The bottom-up approach may also tailor data to a particular domain or text types.

The top down approach, by contrast, 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 (Flickinger *et al.* [9]) and the DITO test suite (Nerbonne *et al.* [10]). 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 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.

As far as methodology is concerned, the TSNLP adopts the top down approach. However, the data is 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. Of course, it is a non-trivial problem how to ascertain how frequent a phenomenon is in a corpus. Some work in this direction has been done within the project (Arnold *et al.*[3]). And, although some application-specific data is being constructed, the bulk of the data is core data, that is intended to be both central to general language and relevant for the testing of any syntactic-based system. Particular attention is given to the coverage of closed classes.

# 3 Control over test data

Test suites are not the only type of reference data for testing and evaluating NLP applications. An important, and complementary technique, is the use of test corpora (where by *corpora* we mean a collection of naturally occurring text, increasingly in electronic form). The strength of test corpora is that they represent "real" data, that is naturally occurring data. However, given the complexity of naturally occurring data, it is difficult to isolate the phenomenon or combinations of phenomena one is interested in testing. 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 accidently. This is where test suites are useful, since they allow the construction of focused test data illustrating specific phenomena in isolation or controlled combinations. Thus if one is interested in testing a specific phenomenon (e.g. 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. Note moreover that while well constructed test suites will list every phenomenon only once, corpora are clearly redundant, in that certain phenomena will occur over and over again.

We should of course qualify the remark that test suites can test phenomena in isolation. One cannot say that any sentence illustrates just some phenomenon, since language is inherently a combination of different words standing in specific grammatical relations. Testing a phenomenon in isolation should rather be taken as meaning that as many linguistic parameters as possible are kept under control. This holds in particular for those cases where a phenomenon is illustrated by systematic variation over the parameters making up this phenomenon, while all other parts of the test items are being kept constant. In order to control parameters in TSNLP, a number of general guidelines are laid down, such as "use declarative sentences", "avoid modifiers and adjuncts", and "avoid ambiguity". Annotations are also provided to record the phenomena that occur in a test item but which are not the subject of testing (e.g. subject verb agreement in test items illustrating coordination).

# 4 Lexical and structural phenomena

Vocabulary is one of these parameters that need to be controlled, which is not always possible for all phenomena. For some phenomena the choice of the specific vocabulary (besides the functional words) hardly makes any difference and therefore can be kept stable; this is for example true for coordination phenomena and relative clauses. However, some grammatical phenomena depend on specific lexical items (e.g. subcategorization, passive voice, preposition selection). Note also that for some applications, such as controlled language checkers, the choice of vocabulary may be important since many simplified languages restrict the use of certain words in specific contexts to prevent ambiguous interpretations.

In general, however, for a general purpose test suite such as TSNLP, it is desirable that the vocabulary is as neutral as possible. TSNLP uses simple noun phrases and pronouns, which abstract from any specific semantic properties.

# 5 Combination of phenomena

We noted above that the advantage of test suites over test corpora is that they can present phenomena and combinations of phenomena in isolation. However, combining all different possible phenomena would lead to a combinatorial explosion and in any case, not every combination of phenomena presents an interesting test case. Therefore, some constraints must be imposed on the combinations one chooses to test. Ideally one should concentrate on combinations that are crucial for the system under test, or that are most frequent in one's corpus. (Again, the problem of establishing what combinations of phenomena are most frequent in a corpus is a difficult one). Other, arbitrary combinations of phenomena can of course be tested using the test corpus method.

# 6 Ill-formed test items

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). The problem for the test suite writer is similar to that posed by combinations of phenomena, namely that one can construct ungrammatical examples in many different ways, not all of which may be interesting and not all of which can be listed. Ideally, one would want to concentrate on frequently occurring errors, gathered from corpus studies or empirical studies. The problem is that statistics on frequently occurring errors are not readily available. The relative scarcity of error corpora also means that, unlike arbitrary combinations of phenomena, arbitrary error types cannot be checked using the test corpus method.

An alternative approach to ill-formed data, and that adopted by TSNLP, is to generate them from well-formed items by systematically varying the parameters. For example, for agreement phenomena the relevant features could be systematically varied such that grammatical and ungrammatical examples fall out automatically. In these cases, the parameter variations define a logical space which constrains the set of ungrammatical examples. 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 |

# 7 Annotation scheme

Reusability of existing test suites is severely hampered by lack of annotations about their content. Frequently, only section headings are provided. An excep-

```
essex@feb-95@invented@formal@none@1@362611@NP@much firm@0@2@@0:1
@much@DET@spec@1:2@@@1:2@firm@N_sg@func@0:2@@
```

Figure 1: An annotated test item in machine-readable output. The same information, but in human readable form, is presented in figure 2.

| | | | | | |
|---|---|---|---|---|---|
| author | essex | date | feb-95 | origin | invented |
| format | none | difficulty | 1 | register | formal |
| item-id | 362611 | category | NP | | |
| input | *much firm* | | | | |
| WF | 0 | | | | |
| length | 2 | | | | |
| comment | - | | | | |

| position | instance | category | function | domain | tag | comment |
|---|---|---|---|---|---|---|
| 0:1 | *much* | DET | spec | 1:2 | - | - |
| 1:2 | *firm* | N_sg | func | 0:2 | - | - |

Figure 2: The information in the item annotations, presented in a more readable format.

tion 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 has built upon the DITO annotations. Annotations fall into four different classes:

- item
- phenomenon
- set
- user and application profile

The *item* annotations contain categorial and structural information that is independent of the phenomenon and user parameters. It contains information about the actual string of words, book-keeping information (e.g. author, time of construction, origin), length, category, functional analysis and well-formedness. See figure 2. A common naming system for category and function names as well as high level phenomena names is adopted to ease cross-reference across languages. The *phenomenon* annotations provide a hierarchical classification of the phenomenon as well as information about the "presuppositions" of the phenomena (ie. 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. The *set* annotations allow one to group test items in any combination one wishes (e.g. one can group the positive and ill-formed variants of a particular test item). The *user and application profile* annotations record the centrality of a phenomenon in general language, as well as its relevance 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.

The data input tool (TSCT, described in the following section) produces machine-readable output. The *item* annotations for the determination example given above is shown in figure 1.

# 8 Automating the construction process

The construction of test data is both difficult and time consuming. A number of tools have been developed within TSNLP to aid data construction. Indispensible to the enterprise was the construction of tool (TSCT, which is described in Oepen *et al.* [11]) for inputting data. Here we mention two other tools, the automatic generation tool and the lexical replacement tool, further details of which can be found in Arnold *et al.* [2].

The idea of automatically generating TSs was first proposed by Arnold *et al.*[1], motivated by the fact that, despite the difficulties in test suite construction, some of which have been discussed above, much of the construction process is repetitive and boring. Furthermore, it is easy to make mistakes by, for example , including ungrammatical examples (which are not marked as such), or by missing out examples that are required if a test suite is to be systematic. The repetitive nature of the task, and the need for systematicity suggested that at least some of the processes involved in test suite construction could (and should) be automated.

Thus a small prototype grammar-based *Test Suite Generator* (TSG) was developed within the project, and tested during the data writing. The tool is useful in generating systematic variations of a phenomenon. Its usefulness for test suite construction could be enhanced by getting it to generate more annotations automatically and writing an interface to convert its output into database format. Another area for future investigation would be to get it to generate user-specified error types.

TSNLP concentrates on the production of core phenomena, not specific to any particular domain or text type. In order to use it in a specific evaluation set up, the user may be required to customise the data to his application, text type and or domain. Part of this customisation process includes lexical replacement. TSNLP explored how this process may be automated, by implementing a small emacs-based *Lexical Replacement Tool*. A major problem in applying the tool, however, was the unavailability of suitable lexica with which to hook it up. Other possibilities for lexical (and phrasal) replacement include aligning test items with corpora on the basis of tags. A "tag" slot is provided in the annotation scheme for each constituent to allow such a possibility, but it is left to the user of the test suite to fill in values for this slot. This is again an area for further research.

# 9 Conclusion

We have discussed some theoretical and practical issues in the design and construction of test suites. The results of the project will become public domain and it is hoped that the availability of validated test suites for a range of applications, together with the tools for their manipulation and use will be of great value to the NLP community. Intermediate reports of the project can be obtained from `balka@essex.ac.uk`.

# Bibliography

# References


[1] Arnold, D., Moffat, D., Sadler, L., and Way., A. (1993). *Automatic Test Suite Generation*, in Machine Translation, Vol **8:** 29-38.

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

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

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

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

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

[9] 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.

[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.

[13] Wojcik, R., Harrison, P., and Bremer, J. (1993), *Using Bracketed Parses to Evaluate a Grammar Checking Application*, in: Proceedings of ACL, Boeing Computer Services Research and Technology Division, Seattle.