A language technology test bench – automatized testing in the Divvun project

Sjur Nørstebø Moshagen
Norwegian Sámi Parliament
Norway

sjur.moshagen@samediggi.no

Abstract

The presentation describes a language independent test bench for testing proofing tools, and more generally language technology tools, where the testing is fully automatized. The test results are transformed into xml, and further to HTML. The test bench is freely available as part of the language technology resources in the Divvun project[1] and Centre for Sámi Language Technology at the University of Tromsø[2].

1 Introduction

The development of basic language technology tools for regular end users, such as good spellers and accurate hyphenators, has in practice not progressed since the 1980s, especially within the open source domain. All open source speller engines of today are still list-based as they were in the 70s — they all claim some sort of inheritance from iSpell[3], the (in)famous Unix speller developed originally for English. In the 80-ies the two-level model was developed (Koskenniemi 1983), and further commercialised in proofing tools by Lingsoft[4]. What happened in the 90s was of course the development of grammar checkers based on linguistic analysis, cf. the SCARRIE project (de Smedt & Rosén 2000) and the Constraint Grammar-based grammar checkers from Lingsoft (Birn 2000), but these are not basic tools anymore, and we'll keep them out of the discussion in this article.

There are many reasons for this lack of development, here we will present one cause: the lack of systematic and comparable testing across languages and speller engines to enable easy and automatic comparison of the qualities of available language technologies.

This has led to roughly four tiers in the proofing tools market: 1) good, commercial tools for the big languages – but based on closed source, and with no independent and neutral quality evaluation; 2) reasonably good tools for smaller but rich language societies – still based on closed source and no independent quality assessment; 3) more or less bad tools for many languages, based on open source; and 4) no tools for very many languages.

To help solve this situation, one would need an open, vendor-neutral test bench for proofing tools, together with standardised measures for the quality of these tools. That is what this paper is all about.

2 The Divvun project and automatized testing

The Divvun project develops proofing tools for the Sámi languages, and has so far released spell checkers and hyphenators for North and Julev Sámi. An important secondary goal has been to set up a good, language independent infrastructure to make it easy to add new languages, and an important part of this infrastructure is a good test bench for the tools we make.

In the following we will concentrate on the testing of spell checkers, but we also support testing of hyphenators, and the modular structure of the test bench makes it easy to add support for other tools as well.

The test bench takes three types of input: XML formatted, as tab-separated lines of text, or

[1] http://www.divvun.no/
[2] http://giellatekno.uit.no/
[3] http://fmg-www.cs.ucla.edu/geoff/ispell.html
[4] http://www.lingsoft.fi/

as generated or extracted data from our transducer lexicons. The XML format is used in corpus files for correct-marked documents, and is automatically added from a very simple mark-up system[4] in a copy of the original document. An example of this markup is shown in 1), and the resulting XML is shown in 2).

```
1) Her er ein fiel§(feil).
2) Her er ein <error
correct="feil">fiel<error>.
```

The tab-separated data is used for regression tests, typo tests, and word construction tests. Finally we have a couple of specialised tests to test the conversion from our Xerox-format-based source code to the final proofing tools: baseform tests and paradigm tests.

2.1 Data flow in the test bench

A simple diagram over how the data flows in the test bench is shown in Figure 1.

2.2 How the test results are presented

The test output is read and parsed by a Perl script, and transformed to a standard XML format. From the XML test reports, it is possible to generate all sorts of reports – presently the only supported output is a relatively simple HTML page. In the future we hope to be able to generate overview reports, cross-lingual comparisons, etc.

Although the HTML report is simple, it contains all the relevant statistics for that test run, as well as colour highlighting of essential features. After an introduction with important metadata, the statistics follow, and then the body of the test output from the speller. An example of such a test report can be found on our web site[5].

3 Different speller engines

As seen in Figure 1, we support different speller engines, and it is straightforward to add support for new ones. It is easiest if the speller engine has a command line interface (the test bench is meant to be run from a Unix-like prompt), but it is also possible to script a GUI host application. This is how we run the MS spellers in Word — by using an AppleScript (which can be started from the command line) to script Word, we can run the speller test suit through all languages with built-in speller support in MS Word.

By supporting different spell checkers and spell checker engines, it becomes easy to compare both lexicons and speller engines. We have developed two different versions of our North Sámi Speller, one for MS Office using a speller engine from Polderland[6], and another for OpenOffice, using Hunspell[7] as the speller engine. Table 1 gives Precision and Recall for the two spellers, using a gold standard document as the test data[8].

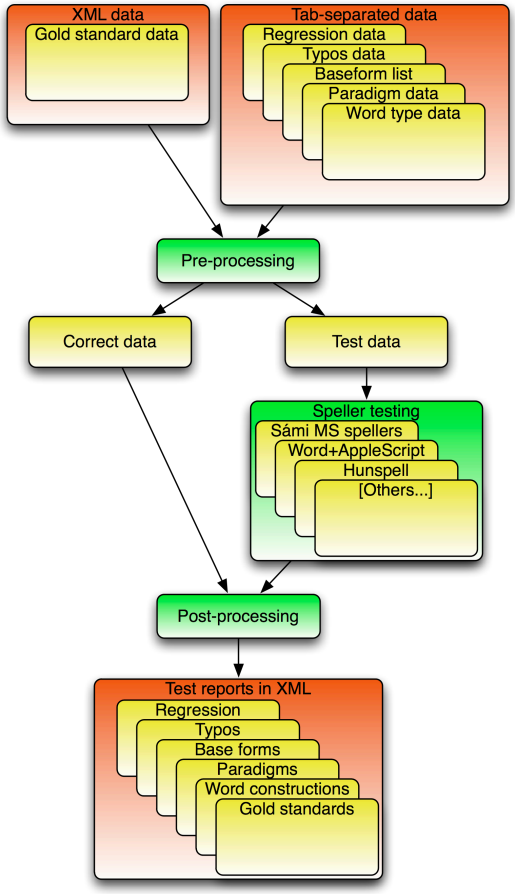

Figure 1: Data flow in the test bench

	Polderland	Hunspell
Precision	89.57	84.07
Recall	98.10	90.48

Table 1: Precision and recall for two North Sámi spell checkers

These figures are of course mainly a measure of how well we have been able to formulate the North Sámi grammar within the limits of the formalism for each spell checker engine. The point in this paper is rather that until now it has been very hard to do such comparisons, while our test bench has turned the task into one simple command on the command line.

[5] http://www.divvun.no/doc/proof/spelling/testing/error-markup.html

[6] http://www.polderland.nl/

[7] http://hunspell.sourceforge.net/

[8] http://www.divvun.no/doc/proof/spelling/testing/Markansluska-pl-forrest-sme-20081013.html

There are other measures of the quality of a spell checker. Table 2 shows the percentage of all spelling errors with: correct suggestion (first row); correct suggestion among the top five suggestions (second row); only incorrect suggestions (third row); and no suggestions (last row). That is, these figures measure the ability to provide relevant suggestions.

	Polderland	Hunspell
Corr sug/all errs	85.44	74.73
Corr sug in top 5	82.52	74.73
Only incorr sug	13.59	25.27
No suggestions	0.97	0.0

Table 2: Suggestion quality for our two North Sámi spellers, gold standard test[7]

In Table 3 the same type of figures is given for another type of test data, a collection of known spelling errors and their corrections[9].

	Polderland	Hunspell
Corr sug/all errs	81.22	72.72
Corr sug in top 5	78.84	72.40
Only incorr sug	15.67	27.28
No suggestions	3.11	0.0

Table 3: Suggestion quality for our two North Sámi spellers when tested on a collection of known typos.

The figures in Table 2 & 3 show that there is a significant difference between the two engines in their ability to provide relevant suggestions. The difference corresponds relatively well to the subjective impression, although I had expected an even bigger difference.

In normal usage the Polderland-based North Sámi speller has a correct suggestion more often than the Hunspell-based one, roughly for 10% more of the spelling errors. Another noticeable difference is that Hunspell never returns nothing – you always get one suggestion or another. In Hunspell's case this means that in regular use as modelled by the gold standard test the speller will suggest just noise in one out of 4 spelling errors. The other speller does the same only in 1 out of 7.

This is a very noticeable difference for the end users. The suggestions are so to speak the user interface of the speller, and the perceived overall quality of the speller will be influenced by the quality of the suggestions. And for minority language writers, the suggestions tend to be more important than for majority language users, since you can expect to find more insecure writers in the minority language community.

[9] http://www.divvun.no/doc/proof/spelling/testing/typos-pl-forrest-sme-20081113.html

4 Further development

The open-source[10] test bench is a work in progress. Among the things we would like to add is support for more speller engines, and other types of proofing tools like grammar checkers. Also, there is much that can be done to extract more statistics and create better reports, as well as to add precision and recall metrics on the suggestions (cf Bick 2006). We would as well like to be able to test more languages.

5 Conclusion

Having access to an open and modular test bench for proofing tools will hopefully be a valuable asset to further develop and improve the most common and important writing aid, the spelling checker. And the possibility to compare different technologies could increase the interest in improving existing tools, and in the best of cases develop new ones. Basic proofing tools are a requirement for supporting small language communities, and the communities deserve better tools than what they are served now. We hope the test bench can be a small contribution in that endeavour.

References

Bick, Eckhard (2006): "A Constraint Grammar Based Spellchecker for Danish with a Special Focus on Dyslexics" in *A Man of Measure – Festschrift in Honour of Fred Karlsson*, pp. 387–396

Birn, Jussi (2000): "Detecting grammar errors with Lingsoft's Swedish grammar checker". In Torbjørn Nordgård (ed.) *NODALIDA '99 Proceedings from the 12th Nordiske datalingvistikkdager*, pp. 28–40. Trondheim: Department of Linguistics, University of Trondheim.

Koskenniemi, Kimmo. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. [PhD dissertation]. Publications of the Department of General Linguistics, University of Helsinki, No. 11.

de Smedt, Koenraad & Victoria Rosén (2000): "Automatic proofreading for Norwegian: The challenges of lexical and grammatical variation". Published in: Nordgård, T. (ed.) *NODALIDA '99: Proceedings from the 12th "Nordiske datalingvistikkdager"*, Trondheim, 9-10 December, 1999 (pp. 206-215). Trondheim: NTNU.

[10] Access to our Subversion repository is protected, but a user name and a password will be given by sending an e-mail to divvun@samediggi.no. Further instructions on our home page, see Footnote 1.