Technological evaluation of a controlled language application: precision, recall, and convergence tests for SECC

*Geert Adriaens** and Lieve Macken*[+]

** Siemens Nixdorf Software Centre Liege*
Rue des Fories 2, B-4020 Liege, @csl.sni.be
[+] *University of Leuven Centre for Computational Linguistics*
Maria-Theresiastraat 21, B-3000 Leuven, @ccl.kuleuven.ac.be

Abstract

We report on two performance tests we did with the SECC controlled English checker and corrector. Goal of the tests is a technological evaluation of the tool: current performance, potential optimal performance. The first test is a precision/recall test using a "real-life test suite"; we discuss its results in terms of specific system components that need improvement, and also compare it to a similar test done with the Boeing SE Checker. The second test is a newly devised one; we call it a *convergence test*. Its goal is to check whether the system leaves its proposed automatic corrections untouched in a subsequent run. Results show a degree of non-convergence, and point again at previously unnoticed weaknesses in the system that must be taken care of. The problem types that came out of both tests can be of interest for developers of similar grammar checking (and correcting) applications.

1. Introduction

1.1 SECC in a nutshell

SECC *(A Simplified English Grammar and Style Checker/Corrector)* is a controlled language application in development in the context of an LRE-2[1] project with the same name. Development partners in the project are Siemens Nixdorf (BE), the University of Leuven (BE), Cap Gemini Innovation (FR) and Sietec (DE); intended user is Alcatel Bell (BE)[2]. The project runs from November 93 till May 96, and is currently at a stage where a major deliverable (the sentence-level batch writing tool) is finished. This tool checks technical English documentation in the field of telecommunication (subfield telephony)

[1] LRE stands for *Linguistic Research and Engineering;* it is a framework for projects co-funded by the European Commission.
[2] In more personalised terms, SECC is also: Luc Pauwels (University of Leuven); Anne Derain, Philippe Levisse, Olivier Clavel (Cap Gemini); Frederik Durant (Siemens Nixdorf); Patrick Goyvaerts (Alcatel Bell); and Uus Knops (Sietec).

against a controlled English lexicon (basic vocabulary and technical vocabulary) and a controlled grammar in use at Alcatel in Belgium. It is "sentence-level", because it does not check text units beyond the sentence; it is "batch", because it only handles the submission of completed texts.[3] As described in more detail elsewhere[4], SECC is conceived as a special language pair (English to SE) within the Metal® MT development environment. Its input can be any English sentence; its output is a dual object: the input sentence annotated with error messages, and a proposed correction for the whole sentence.

As to error messages and correction, an important distinction we make for SECC is that between a *weak* and a *strong* diagnosis. A weak diagnosis refers to a *possible* error: the system detects a phenomenon in the input that sometimes is an error, and sometimes is not. Due to its limitations (lack of semantic and contextual information), however, SECC can never be sure it is an error. A case at hand is *Avoid splitting infinitives unless the emphasis is on the adverb.* The system can detect a split infinitive, but it has no indication of emphasis. It will be clear that cases of weak diagnosis (possible error) never trigger automatic correction. A *strong* diagnosis refers to a *certain* error: the system detects a phenomenon in the input that always is an error. An example is *Use a (an) before a word that starts with a consonant (vowel) sound. A+vowel sound* or *an+consonant sound* can be detected and flagged as errors. In cases of strong diagnosis, automatic correction is sometimes possible and done by SECC (as for the *a/an* distinction). In all, SECC then distinguishes between three cases: W-DIAG (weak diagnosis), S-DIAG (strong diagnosis, no correction), and S-CORR (strong diagnosis, correction). As to numbers of rules: the Alcatel grammar (COGRAM) contains 141 rules in all (including a general lexical rule *Use only approved basic or technical SE-words*[5]*)* . SECC currently implements checks for 72 of these: 7 W-DIAG, 10 S-DIAG, and 55 S-CORR (so, in all 65 cases of strong diagnosis). All through this paper, we will give examples of the SECC output as it appears in a batch error report. The proposed correction is given first, then the original input sentence, followed by all syntactic errors grouped according to the part of the sentence they apply to.

---

[3] Current developments address the issues of checking units above the sentence level (from paragraph to full text), and of allowing "interactive" or "on-line" checking of a selected text part within a word processor.

[4] See Adriaens 1994a or 1994b.

[5] A few figures about lexicon sizes: the Metal® general English lexicon we use to date contains over 50000 base forms; as subsets of it, the Alcatel basic SE lexicon (COLEX) contains about 1750 approved words, and the Alcatel technical SE lexicon (COTECH) about 550 (taken from a text corpus of 2500 sentences we use as a testbed for SECC). SECC further contains SE translations for some 1200 non-SE words (basic and technical).

Finally, all errors violating the above general lexical rule are listed; they come in three subtypes:

1. words that are not SE, and have an SE-equivalent are shown as *x --> y*
2. words that are not SE (but are English), and do not have an SE-equivalent are shown as *x --> ?*
3. words that are not SE, nor recognised as English (i.e. not in the dictionary) are shown as *x ??*

To conclude this short introduction to SECC, let us mention that there exist several other controlled English applications comparable to ours: Boeing's BSEC *(Boeing Simplified English Checker)*[6], Carnegie's ClearCheck (reported on elsewhere in this volume), GSI-Erli's AECMA Checker[7], Cap Volmac's AECMA Checker[8], Oracle's CoAuthor AECMA Checker[9]. However interesting a comparison between these systems could be, it falls beyond the scope of this paper. Let us suffice by saying that the number of these applications is growing, that most of them are based on the SE standard used in the aerospace industry (AECMA), and that (currently) only few offer far-reaching automatic correction (SECC, Cap Volmac). We will come back to the Boeing BSEC and to the issue of automatic correction in the tests reported below.

1.2 Goal of this paper

The goal of this paper is to report on a technological evaluation of SECC (as opposed to operational evaluation at the Alcatel user site). For this evaluation, we did two tests: a precision/recall test, and a convergence test. The first type of test is well known from the information retrieval domain, and has also been done for the Boeing BSEC (see section 2). The second type of test is new, and has (to our knowledge) never been done before; it is strongly related to the automatic correction feature of the system (see section 3). As a general note to these tests, we want to stress that they are not about naked figures in the first place (we claim no statistic relevance whatsoever), but about what the figures reveal in terms of strengths and weaknesses (i.e. opportunities for improvement) of the system. The discussion also aims at stressing findings that can be of interest to anybody in the NLP community concerned with building similar applications or devising tests for technological evaluation of NLP applications.

---

[6] See e.g. Wojcik et al 1990, 1993.
[7] We are not aware of publications about GSI-Erli's Checker; it was demonstrated at the First Language Engineering Convention in Paris, July 1994.
[8] See e.g. van der Steen and Dijenborg, 1992.
[9] CoAuthor is available as an off-the-shelf software package.

2. Precision/Recall

In the context of grammar checking, precision and recall refer to the following rates obtained when running the application on a test corpus:

$$\text{Precision} = \frac{\text{Number of correctly flagged errors}}{\text{Total number of errors flagged}}$$

$$\text{Recall} = \frac{\text{Number of correctly flagged errors}}{\text{Total number of errors actually occurring}}$$

Good precision requires a low rate *of spurious* errors, i.e. errors that the system reports, but are not there; good recall requires a low rate of *missed* errors, i.e. errors that are there, but are not reported by the system.

For SECC, the corpus we used for our precision/recall test is a cross between a real-life corpus and a test suite, i.e. a "real-life test suite". To understand its nature, a word is in order about COGRAM, Alcatel's paper grammar that lies at the basis of SECC. Rules in COGRAM look like this:

Do not express an idea in parentheses inside a sentence. Use a separate sentence instead. Use parentheses only to enclose abbreviations.

- The operator can correct the taxation information (add or subtract taxation units).

+ Public-Switched Telephone Network (PSTN) [abbreviation]

Use <u>of</u> in the genitive case when a possessive noun form is inanimate.

- the System 12's novel and attractive architecture

+ the novel and attractive architecture of System 12

Each rule in COGRAM is accompanied by at least one example containing a violation of that rule[10] (the minus-examples); unless the correction is either obvious or too complicated (as for the minus-example in the first rule above), there is also a corresponding plus-example suggesting a correction for the error at hand (as for the second rule above). In cases where a phenomenon is sometimes an error and sometimes correct, a rule can also

[10] Given its real-life nature, the example may contain multiple occurrences of the error, as well as other errors.

contain plus-examples without a minus-example (illustrating the correct usage, as for the first rule: parentheses are only allowed for abbreviations). All the examples are taken from text material produced by the technical writers for whom the grammar is intended (this is the "real-life" aspect). The important thing now is that our test suite contains only the minus-examples, and the plus-examples without minus-counterpart. For the former, it is clear we cannot add the plus-examples, since they contain corrections (S-CORRs) that SECC should (partly) make[11]. The plus-examples without minus-counterpart are important though, because they are a good test for SECC's precision. In all, this test suite contains 253 sentences with a average sentence length of 9.9 words; it consists of 238 minus-examples, and 15 plus-without-minus-examples. Before our precision/recall test, they had not been tested exhaustively with SECC. The most interesting characteristics of this test set lie in its non-artificial nature, and its guaranteed coverage of all implemented rules[12]. All 253 sentences were first hand-annotated for all errors that they contain (giving us the reference material for the test). Next, SECC was run on the corpus, and its output was checked for missed and spurious errors.

In all, the hand-annotated corpus revealed 541 errors. SECC flagged 574 errors, of which 73 were spurious; 40 errors were missed. This gives a total of 501 correctly flagged errors[13]. SECC's precision rate for our test suite is then 501/574, i.e. 87%; its recall rate is 501/541, i.e. 93%. Although both tests are difficult to compare[14], Wojcik et al. (the Boeing BSEC) found a precision rate of 79% and a recall rate of 89% for their test on a statistically representative sample from a large body of real-life text. The only thing we want to point out about these figures is that both systems perform worse for precision than for recall. If

---

[11] The effect of (re)submitting this kind of examples to SECC is exactly the object of the convergence test discussed in section 3.

[12] In the context of a collaboration between the SECC project and another LRE-2 project TSNLP (Test Suites for Natural Language Products), another test suite is under construction using the SECC resources. It will be interesting to compare the results of testing this suite to the results reported here.

[13] Alternatively, (total occurring)-(missed), 541-40, or (total flagged)-(spurious), 574-73, i.e. 501.

[14] A detailed comparison would lead us too far; major differences are: BSEC was tested on a representative real-life test corpus, SECC on a real-life test suite; BSEC's corpus came from writers supposed to apply the SE rules, SECC's material dates from before the introduction of COGRAM at Alcatel; BSEC only tested for the 10 most frequently occurring rules, SECC tested for all rules; BSEC did not start from a corpus annotated by hand for the errors but only for the correct analyses (assuming that a correct analysis guarantees correct error reporting), SECC started from hand-annotation for the errors (not making the BSEC assumption). Finally, BSEC and SECC are based on different grammars (AECMA versus COGRAM), differ in parser and lingware technology (BSEC does not incorporate a recovery strategy when no sentence analysis is found, whereas SECC does - see the discussion of "phrasal analysis"; BSEC only has the error category S-DIAG), and cover different domains (aviation versus telecommunication). If it is hard setting up evaluation tests for NLP applications, it is clear that comparing them is not exactly easier.

we link technological evaluation to operational evaluation (the user appreciation of the system), it is exactly the precision rate that should be improved as much as possible. Spurious errors are at least misleading, often irritating (especially if there are many of them), and in the worst case they lead to a total rejection of the tool (when the user *is sure* that the errors are spurious). Missed errors are not so bad (from a user's point of view): mostly, the user will never know there were missed errors at all. For one thing, if he knew what errors he made, he would not need the tool; for another, missed errors by definition do not show up in the system output (so they cannot be a source of irritation).

We examined the causes of the missed and spurious errors (see table 1 on the next page), in order to see if the system could be improved. Indeed, all cases of lingware problems, monolingual lexicon problems and "bilingual"[15] lexicon problems were easy to solve. Thus, 6 (1+0+5) missed errors and 29 (6+18+5) spurious errors could be eliminated right away. Running SECC after these modifications gave a precision rate of 93% and a recall rate of 94%[16]. Given the nature of our test data (a test suite) and the optimisations made, we estimate that this is close to the *optimal* performance SECC can attain. If some of the remaining hard but solvable problems are solved (wrong and phrasal analysis), the absolute optimum could be 95% for precision and recall[17].

[15] Recall that SECC's bilingual or transfer lexicon contains mappings of non-SE words to SE words; in our approach, the SE words themselves are also identified in the transfer lexicon (not in the source/target lexicon), i.e. as reflexive transfer entries (see Adriaens 1994a/b).

[16] Precision: (501+6)/(574-29), i.e. 507/545, 93%; recall: (501+6)/541, i.e. 507/541, 94%.

[17] Note that both rates (precision and recall) converge to the same (hypothetical) optimum of 95%.

Cause	Type	Number	Total per cause (missed+spurious)
phrasal analysis	missed	15	
	spurious	14	29
wrong analysis	missed	6	
	spurious	26	32
system limitations (lack of semantics and contextual knowledge)	missed	11	
	spurious	3	14
ungrammatical input	missed	0	
	spurious	1	1
deformatting	missed	2	
	spurious	0	2
lingware (transfer-generation procedures)	missed	1	
	spurious	6	7
monolingual lexicon	missed	0	
	spurious	18	18
bilingual lexicon	missed	5	
	spurious	5	10
OVERALL TOTAL	missed	40	
	spurious	73	113

Table 1: causes of missed and/or spurious errors

As to other figures, it is interesting to mention that of the 72 rules implemented (and addressed by the test suite), only 21 gave rise to missed and/or spurious errors (see table 2 on the next page; the rules are ordered by descending frequency of flagged errors). To get an idea of the relative weight of the rules, we composed a top-10 of rules generating the • most error messages. Of course, this top-10 was not derived from the test suite results; we derived it from the real-life corpus we used for the convergence test discussed below. Table 3 shows the top-10 of these rules[18]. (The error frequencies are also those found in that corpus; they are added here without further comments). Crossing this list with the list of 21 rules yielding missed/spurious errors, we find 8 of the 10 most frequent errors among these 21 (they are marked with a asterisk). This is again a good indication of what rules we should concentrate on when trying to improve system performance. In any case, it was a good sign that all of the spurious errors that can be corrected (the 29 mentioned above) are generated by rules from the top-10.

[18] Four of these phenomena are also in the BSEC top-10 of most frequent errors: usage of non-SE words (1st), missing articles (2nd), bad use of noun clusters (3rd), and passives (4th). For both applications, errors against SE word usage are by far the most frequent.

Cogram rule (abridged wording)	Errors actually occurring	Flagged errors	Missed errors	Spurious errors
Non-SE word (*)	156	185	4	33
Do not use the passive voice (*)	78	75	4	1
Missing article (*)	12	28	3	19
Only capitalise proper nouns and technical terms (*)	19	23	4	8
Express action with verbs, not with verb-derived nouns (*)	22	17	5	0
Avoid complex multi-word compounds (*)	18	14	6	2
Repeat articles in co-ordinated word groups	6	8	0	2
It must refer to a preceding noun	6	8	0	2
No nominal subclauses as subject	6	8	0	2
Use parentheses only to enclose abbreviations (*)	6	7	1	2
Avoid *there* + verb	8	6	2	0
Preposition + *whom* for persons in relative clauses	7	5	2	0
That in restrictive relative clauses, or commas in non-restrictive ones	6	5	1	0
Hyphen in certain multi-word compounds (*)	6	5	1	0
Not more than three verb groups per sentence	5	5	1	1
Because for reason	4	3	1	0
Comma + *and/or* before last element in long enumerations	2	3	0	1
Illegal co-ordinating conjunction	3	2	1	0
Number as first word in full	3	1	2	0
Avoid splitting infinitives	2	1	1	0
Adverb of manner after intransitive verb	2	1	1	0

Table 2: missed and spurious errors per Cogram rule

Cogram rule (abridged wording)	Number of detected errors	Number of corrected errors
Non-SE word	416	217
Do not use the passive voice	105	6
Missing article	86	/
Only capitalise proper nouns and technical terms	85	85
Write short sentences *(not in table 2)*	54	/
Use parentheses only to enclose abbreviations	37	/
Express action with verbs, not with verb-derived nouns	37	/
Hyphen in certain multi-word compounds	26	26
Avoid complex multi-word compounds	25	/
One-word numbers in full when surrounded by text *(not in table 2)*	18	18

Table 3: top-10 most frequent errors detected/corrected by SECC (source: convergence corpus)

One rule that jumps to the eye in table 2 because of its bad performance is the rule *"Missing article"* about the correct usage of articles: *Use **the** before a noun for specific reference; use **a/an** before a singular noun for non-specific reference.* It is a top-10 rule (also for Boeing's BSEC), and scores only 32% for precision (75% for recall). This rule relies heavily on exhaustive and correct coding of the *mass-count* distinction in the monolingual lexicon. In the implementation, singular countable nouns cannot occur without a determiner. Spurious errors are caused by incorrectly coding words as *count* (when they should either

be *mass* or *count-and-mass).* Missed errors are caused by either coding *mass-nouns* as *count-nouns,* or in cases where words are coded (correctly or wrongly) as *count-and-mass.* In this last case, SECC has no way of disambiguating, and remains silent; it does not issue a W-DIAG either, because this would lead to too many spurious (i.e. irrelevant) messages. Again, the bottom Line is that the test results point at a very specific phenomenon in the system that should be improved (lexicon coding).

A final word is in order about the effect of wrong and phrasal analysis. With "wrong analysis", we refer to cases where SECC finds an overall sentence analysis, but not the right one (e.g. with wrong attachments); with "phrasal analysis", we refer to the fail-soft mechanism that always returns the most interesting constituent chunks identified in the input in case no overall sentence analysis is found (also called "fitted parse"[19]). Given earlier statements about the effect of bad and fitted parses, we were interested in their effect on system performance. The general feeling[20] is that bad parses (fitted or not) still yield fairly reliable error messages. As table 1 shows, we have to mitigate this. For the cases of bad analysis, 35% of the spurious errors (26 out of 73) occur in badly analysed sentences. As to phrasal analysis, it yields about as many missed as spurious errors, and is globally taken responsible for 25% (29 out of 113) of all missed and spurious errors. That it gives rise to a lot of missed errors (37% of all missed errors) is not surprising: the analysis is incomplete, and hence certain rules never get applied to it. In our test suite, we have too few cases of phrasal analysis to make claims for or against fitted parsing. Some phrasals do yield correct error reports (see also below for a discussion of phrasal analysis and convergence). In any case, the lesson to be drawn is that the analysis component of the system is critical for the performance of SECC (or any such application, for that matter). In the SECC project, an additional workpackage has been defined that is precisely meant to reduce cases of bad and phrasal analysis. As soon as a new version of the analysis component is available, we can check the effect by redoing the precision/recall test.

[19] See Jensen and Heidorn 1993, in particular the contributions by Ravin, and Richardson and Braden-Harder (the IBM PLNLP approach and Critique system).
[20] See Wojcik et al. 1993, Richardson and Braden-Harder 1993.

3. Convergence

3.1 Definitions

SECC is currently still one of the rare systems that offer proposals for correction in the form of completely (re)-generated "target" SE sentences. Given this feature, we were intrigued by the effect of resubmitting proposed corrections to the system. The intuition is that SECC should not flag its own corrections; if it does, we should again be able to pinpoint flaws in the system and find room for improvement.

We define the notion of *convergence* as the following rate:

$$\text{Convergence} = \frac{\text{number of automatically corrected sentences (runl) that remain untouched (run2)}}{\text{number of automatically corrected sentences (runl)}}$$

A sentence is considered to be corrected automatically as soon as *one* correction is made (one case of S-CORR suffices).

The notion "untouched" goes in two directions:

- First, no cases of weak diagnosis and of strong diagnosis without correction (W-DIAG and S-DIAG) from runl should disappear in run2 (cp. missed errors in recall). As a "benign" exception, however, we accept the logical disappearance of a W-DIAG or S-DIAG as a side-effect of an S-CORR (where both relate to the same input segment).

  An example is the following:

  RUN1

  CORRECTION:
  answers

  DIAGNOSIS:
  responses

  responses
  COGRAM 63: Express action with verbs, not with verb-derived nouns. [W-DIAG]

  responses (N) --> answer, reaction [S-CORR]

  RUN2

  <the input *answers* does not generate any messages>

  In the first run, *responses* generates two error messages; it is lexically corrected to *answers* (S-CORR). As a consequence, the other error message (W-DIAG) that was triggered by *responses* (rightly) disappears.

- Second, no new cases of strong diagnosis, with or without correction (S-DIAG and S-CORR) should be introduced in run2 (cp. spurious errors in precision). Although a rigid interpretation of "untouched" might exclude the acceptance of new weak diagnosis messages in the second run, we do accept new W-DIAGs in the second run.

  A relatively innocent example is the following: SECC corrects *provided that* to *if* (runl); if triggers a W-DIAG, warning against the potential wrong usage of the conditional conjunction as a temporal one (where *when* is intended). A more controversial example is the following:

RUN1

CORRECTION:
this concentrator is designed on the same technology as A1000 S12.

DIAGNOSIS:
this is a concentrator designed on the same technology as A1000 S12.

this is a concentrator designed on the same technology as A1000 S12
COGRAM 102: Avoid "this is", "these are", and so on. [S-CORR]

RUN2

CORRECTION:
this concentrator is designed on the same technology as A1000 S12.

DIAGNOSIS:
this concentrator is designed on the same technology as A1000 S12.

is designed
COGRAM 81: Do not use the passive voice. [W-DIAG]

The first run flags and corrects *this is..*; this results in a passive sentence, and some cases of passive are only weakly diagnosed (other cases are strongly diagnosed and even corrected) [21].

[21] Although we cannot make a thorough analysis of the value of W-DIAG (value one might have doubts about), we can briefly say a few things about it. The source of its existence is to a large extent pedagogical: point users to the potential difficulties associated with certain constructions. On the other hand, it is clear that W-DIAGs are not always relevant *for the sentence that triggers them* (if they do not apply, one could consider them as spurious errors). To get a better idea of their usefulness, we checked the relevance of all W-DIAGs in the convergence corpus (i.e. is the possible error actually an error in the case at hand?). We found that 40% of the W-DIAGs were indeed relevant. Rather than leaving them out completely, we will make sure that users can switch off the W-DIAGs if they find them too irritating.

To complete the introduction of the convergence notion, we give one more example that converges completely (examples of non-convergence are discussed further below).
In the first run five SE-errors (four grammatical and one lexical) are corrected (S-CORR), two more certain errors are signalled (S-DIAG), and for one a warning is given (W-DIAG).
In the second run, the two strong error messages and the weak one are reported again. No new error messages are generated.

RUN1

CORRECTION:
Each call that waits for connection to an operator is considered to he in a queue, regardless of the shape that such a queue can take.

DIAGNOSIS:

Each call which is waiting for connection to an operator is considered to be in a queue, regardless of the form such a queue may take.
COGRAM 29: Write short sentences. [S-DIAG]

which is waiting for connection to an operator
COGRAM 43: "That" for non-persons in restrictive relative clauses. [S-CORR]

is waiting
COGRAM 79: Avoid present participles in verb groups. [S-CORR]

connection
COGRAM 73 74: "a"/ "an" for non-specific reference; "the" for specific reference. [S-DIAG]

is considered
COGRAM 81: Do not use the passive voice. [W-DIAG]

such a queue may take
COGRAM 44: Relative clause must start with relative pronoun. [S-CORR]

may
COGRAM 86: Only "can" for ability/possibility in the present. [S-CORR]

form (N) —> shape , document, page [S-CORR]

RUN2

CORRECTION:
Each call that waits for connection to an operator is considered to be in a queue, regardless of the shape that such a queue can take.

DIAGNOSIS:

Each call that waits for connection to an operator is considered to be in a queue, regardless of the shape that such a queue can take.
COGRAM 29: Write short sentences. [S-DIAG]

connection
COGRAM 73 74: "a"/ "an" for non-specific reference; "the" for specific reference. [S-DIAG]

is considered
COGRAM 81: Do not use the passive voice. [W-DIAG]

3.2 General test results

For our convergence test itself then, we collected 503 sentences from a corpus of Alcatel telecommunication text of over 3000 sentences (by picking every sixth sentence). We ran them through SECC, and resubmitted all sentences that gave rise to at least one S-CORR[22]. Here are the results:

RUN1

Number of correct sentences	164	
Number of automatically corrected sentences	238	
Number of sentences with only W-DIAG and S-DIAG	101	
Total number of sentences	503	

RUN2

Number of correct sentences	225	(up 12% from 164)
Number of automatically corrected sentences	31	
Number of sentences with only W-DIAG and S-DIAG	247	
Total number of sentences	503	

From the bare figures, it is clear that 31 sentences did not converge in the second run (new S-CORR occurrences). Examination of the second run in more detail revealed another 24 non-converging sentences (introducing new S-DIAGs, or suppressing original W-DIAGs/S-DIAGs), which brought the total of non-converging sentences to 55. Hence, the convergence rate is (238-55)/238 = 183/238, i.e. 77%. In other words, a little more than 2 sentences out of 10 did not converge.

As for our precision/recall test, we examined the causes of the divergence to see if some problems could easily be fixed (see table 4 on the next page). All 17 cases of lingware problems could indeed be fixed, whereas the cases of different analysis due to homography, and the cases of bad or phrasal analysis require more work. With the lingware problems fixed, the convergence rate goes up slightly from 77% to 84% (183+17/238 = 200/238), leaving us with a little less than 2 non-converging sentences out of 10.

[22] Note that for the convergence test, we were not concerned about precision and recall in the first run; convergence is tested here as a purely application-internal (SECC-internal) matter.

Cause	Number of sentences
Lingware (transfer-generation procedures)	17
Different analysis (first run and second run) due to homography	10
Miscellaneous cases (due to a bad analysis or a phrasal analysis)	28
TOTAL	55

Table 4: Causes of non-convergence

3.3 Lingware problems

Refining the category "lingware problems", we saw that some cases of non-convergence were simply caused by bugs that led to badly executed correction (generation) of the target SE sentence. A simple example is the case where the first run replaced the illegal "..." (in enumerations) at the end of a sentence by "and so on", without adding the full stop. The second run then does an S-CORR for the rule *End a sentence with a full stop,* adding the final punctuation. Note again that without the convergence test, some of these errors might never have been found, or only during productive use of the tool.

Beside these SECC-internal errors, there are three potential sources of divergence with which any system might be confronted:

1. *lexicon-lexicon* feeding relationships (with possible circularity): replace X by Y, and replace Y by Z (or even worse: by X)
2. *lexicon-grammar* or *grammar-lexicon* feeding relationships (with possible circularity): the lexicon replaces X by Y, and the grammar corrects Y to be Z (or even worse: to X), and vice versa
3. *grammar-grammar* feeding relationships (with possible circularity): one grammar rule corrects X to be Y, another corrects Y to be Z (or even worse: to X)

We found no cases of lexicon-lexicon feeding relationships. This is not surprising, because we run consistency checks on our monolingual and bilingual lexicons to prevent these feeding relationships from occurring. We regularly check if

- all source entries in the English-SE bilingual lexicon exist in the English lexicon
- all target entries in the bilingual lexicon exist in the SE lexicon, i.e. are SE-accepted (by verifying if there is a reflexive entry for them in the bilingual lexicon — this is our way of marking an entry as SE-approved)
- no circularities occur in the bilingual lexicon

We did find cases of the other possible feeding relationships, though, that we were not aware of before the convergence test. A lexicon-grammar feeding relationship existed

between the dictionary entry *therefore --> thus* (S-CORR), and the grammar rule *Avoid connecting adverbs such as "hence", "**thus**", "as such" or "so"* (S-DIAG). A grammar-grammar feeding relationship is illustrated by the following example:

RUN1

CORRECTION:
25 ISDN exchanges were active.

DIAGNOSIS:
A total of 25 ISDN exchanges were operational.

A total of
COGRAM 62: Avoid wordiness. [S-CORR]

operational (A) --> active , working [S-CORR]

RUN2

CORRECTION:
25 ISDN exchanges were active.

DIAGNOSIS:
25 ISDN exchanges were active.

25
COGRAM 121: Number as first word in full. [S-DIAG]

In the first run, *a total of is* deleted (wordy expression). If, however, this deletion happens in the beginning of a sentence, the possibility exists that a number becomes the first word. If so, it gets flagged because it should be written in full.

3.4 Different analysis

As table 4 showed, there are 10 cases (18%) in which a particular phenomenon causes divergence: a different analysis for a slightly changed sentence, due to homography in one of its words. In these cases, the effects can be quite unpredictable (potential strong divergence). An example:

RUN1

CORRECTION:
Exchange auxiliary control

DIAGNOSIS:
Exchange Auxiliary Control ;;***Exchange*** *analysed as NOUN, the whole as an NP*

Auxiliary
COGRAM 70: Only capitalise proper nouns and technical terms. [S-CORR]

Control
COGRAM 70: Only capitalise proper nouns and technical terms. [S-CORR]

Auxiliary (N) --> ? [S-DIAG]

RUN2

CORRECTION:
Interchange auxiliary control.

DIAGNOSIS:
Exchange auxiliary control ;;***Exchange*** *analysed as VERB, the whole as an imperative S*

Exchange auxiliary control
COGRAM 125: Full stop at end of sentence. [S-CORR]

Exchange (V) --> interchange [S-CORR]

In the first run, capitalisation is undone for two words *(auxiliary* and *control)*. In the second run, it shows that the different spelling results in a different analysis: *exchange* is now analysed as a verb, and corrected to *interchange* given the bilingual dictionary entry for it; moreover, a full stop is added to the sentence.

3.5 Bad/phrasal analysis

Most cases of divergence (28 out of 55, i.e. 50%) are caused by a bad or phrasal analysis. Here again, the effects are unpredictable, and we even found cases of circular correction:

RUN1

CORRECTION:
Phase one:

DIAGNOSIS:
Phase 1: ;;*wrongly analysed as an imperative sentence*

1
COGRAM 120: One-word numbers in full when surrounded by text. [S-CORR]

Phase (V)->? [S-DIAG]

RUN2

CORRECTION:
Phase 1:

DIAGNOSIS:
Phase one: *;;now analysed correctly as an NP*

one
COGRAM 122: Use numbers for part numbers. [S-CORR]

The first run corrects *Phase 1* (wrongly analysed) to *Phase one;* the second run corrects *Phase one* (now analysed correctly) to *Phase 1.*

As to phrasal analysis, some final remarks are in order. First, non-convergence is not always bad, as is witnessed by the following case of phrasal analysis: the first run did not yield an overall sentence analysis, but still a correction was done (replacing "..." by *and so on).* The second run did find an analysis for the sentence, and yielded two useful error messages.

RUN1

CORRECTION:
Measurements (number of accepted calls by an operator, and so on)

PARTIAL DIAGNOSIS:
Measurements (number of accepted calls by an operator,...)

COGRAM 127: Use "and so on", not "...". [S-CORR]

RUN2

CORRECTION:
Measurements (number of accepted calls by an operator, and so on)

DIAGNOSIS:
Measurements (number of accepted calls by an operator, and so on)

(number of accepted calls by an operator, and so on)
COGRAM 31: Use parentheses only to enclose abbreviations. [S-DIAG]

number
COGRAM 73 74: "a"/ "an" for non-specific reference; "the" for specific reference.
[S-DIAG]

Of the 283 sentences with correction, 49 were cases of phrasal analysis. Of these 49, 29 remained untouched in the second run (i.e. the convergence rate for phrasals is almost 60%). Although this rate is lower than the overall convergence rate of 84%, we consider it

still high enough to do automatic correction even in cases where no overall sentence analysis is found.

4. Conclusions and further research

We have run two tests on SECC, our controlled English application intended to help technical writers comply with a controlled grammar and lexicon. The first test was a precision/recall test with a real-life test suite, somehow similar to the one done by Wojcik et al. for the Boeing BSEC (a comparable application). Goals of this test were: to find weak spots in the application (critical components that possibly degrade system performance), and to get an idea of the optimal performance the system can attain (viz. 95% precision and recall). The second test was a newly devised convergence test checking whether SECC leaves its proposed sentence corrections untouched upon resubmission. The fact that we did not find 100% convergence (we currently attain around 80%) shows that this kind of test is certainly useful. As with the precision/recall test, the convergence test pointed out several aspects of the system that can further be improved (grammar and lexicon components, their interactions, and so on). From both tests it emerged that an accurate, rich and robust English analysis component is the key to a maximally successful grammar checking and correcting application. For SECC, an improved version of the Metal® MT English analysis is currently being worked on. It will be interesting to redo our tests with this improved version.

With this paper we have tried to show the value of elaborate technological evaluation tests for controlled language checkers/correctors like SECC. We hope that comparable systems will run similar tests (in particular the convergence test), so that new lessons can be learnt about ways to improve the quality of the applications in this promising subarea of NLP.

References

Adriaens, G. (1994a) - The LRE SECC Project: Simplified English Grammar and Style Correction in an MT Framework. *In Proceedings of the 1st Language Engineering Convention (Paris),* 1-8.

Adriaens, G. (1994b) - Simplified English Grammar and Style Correction in an MT Framework: The LRE SECC Project. In *Proceedings of the 16th Conference on Translating and the Computer (London),* 78-88. Also in *Aslib Proceedings,* 47 (3), March 1995, 73-82.

Ravin, Y. (1993) - Grammar Errors and Style Weaknesses in a Text-Critiquing System. In K. Jensen, J.E. Heidorn and S.D. Richardson (eds), *Natural Language Processing: The PLNLP Approach.* Kluwer Academic Publishers, Boston, 65-76.

Richardson, S.D. and L. Braden-Harder (1993) - The Experience of Developing a Large-Scale Natural Language Processing System: Critique. In K. Jensen, J.E. Heidorn and S.D. Richardson (eds), *Natural Language Processing: The PLNLP Approach.* Kluwer Academic Publishers, Boston, 77-89.

Steen, G.J. van der, and Dijenborg, B. (1992) - Linguistic Engineering: Tools and Products. In *Proceedings of the Twente Workshop on Language Technology 2,* University of Twente.

Wojcik, R.H., Hoard, J.E. and Holzhauser, K.C. (1990) - The Boeing Simplified English Checker. In *Proceedings of the International Conference on Human-Machine Interaction and Artificial Intelligence in Aeronautics and Space.* Toulouse, Centre d'Etudes et de Recherches de Toulouse, 43-57.

Wojcik, R. H., Harrison, P. and Bremer, J. (1993) - Using Bracketed Parses to Evaluate a Grammar Checking Application. In *Proceedings of ACL93,* 38-45.