Natural Language Engineering **4** (3): 211–228. Printed in the United Kingdom

© 1998 Cambridge University Press

Validating the TEMAA LE evaluation methodology: a case study on Danish spelling checkers†

PATRIZIA PAGGIO
Center for Sprogteknologi, Njalsgade 80, 2300 Copenhagen S, Denmark
Email: patrizia@cst.ku.dk

NANCY L. UNDERWOOD
Center for Sprogteknologi, Njalsgade 80, 2300 Copenhagen S, Denmark
E-mail: nancy@cst.ku.dk

(*Received 18 June 1998*)

Abstract

This paper describes the work carried out at the Center for Sprogteknologi in Copenhagen to validate the LE evaluation methodology developed by the LRE project TEMAA. TEMAA has developed a framework for the evaluation of LE products, implemented in a Parameterisable Testbed (PTB). The framework allows for a modular, formal and flexible description of user requirements and objects of evaluation, it accommodates test methods of various kind and provides a methodology for assessing test results in the light of the requirements expressed by different user types. While the fundamentals of the TEMAA framework are meant to apply to adequacy evaluation of LE products in general, a detailed methodology has been worked out for the evaluation of spelling and grammar checkers, and applied to the concrete evaluation of Danish and Italian spelling checkers. The main focus of this paper is on showing that the general methodology provides a valid model for designing and carrying out a concrete evaluation, as in the case study on Danish spelling checkers.

1 Introduction

Natural Language Engineering (NLE) is a comparatively young technology. Therefore, it is not surprising that well-defined and accepted standards for the evaluation of LE products have not been established yet, and that concrete evaluations of either research prototypes or market products have each had to set their own standards.

† This paper reports on work carried out within the LRE (Language Research and Engineering) project TEMAA (A Testbed Study of Evaluation Methodologies: Authoring Aids). TEMAA was a collaborative project at Center for Sprogteknologi (CST), Copenhagen; Institut Dalle Molle pour les études sémantiques et cognitives (ISSCO), Geneva; Stichting Taaltechnologie (STT), Utrecht; Language Technology Group (LTG), Edinburgh; and Claris Ireland, Dublin. More information about the project can be obtained by contacting the authors at CST.

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

In recent years, however, there has been an increased awareness of the importance of developing a commonly accepted methodology for the evaluation of LE projects and products, as witnessed by the DARPA-funded MUC and TREC conferences (cf. e.g. Sundheim (1991) and Harman (1995)), and by special workshops and publications on evaluation (cf. e.g. Thompson (1992), Falkedal (1994) and Sparck Jones and Galliers (1996)). Thus, although no such methodology exists as yet, there is an emerging consensus on the sort of framework which is desirable for the evaluation of LE systems.

The two EU-sponsored projects on evaluation TEMAA and EAGLES (cf. Manzi, King and Douglas (1996) and EAGLES (to appear)) have developed a framework for the evaluation of LE products based on a decompositional analysis of the system's features and of the requirements posed by the setup in which the system is to be used. Within TEMAA, the framework has been formalised and implemented as a Parameterisable Testbed (PTB), i.e. a workbench whose function is to guide an evaluator in designing and carrying out specific evaluation tasks tailored to the requirements of different user types.

This paper focuses on the way in which the TEMAA framework can be applied to the evaluation of spelling and grammar checkers' linguistic functionality, and in particular discusses a case study on the evaluation of two Danish spelling checkers carried out at CST. After a general introduction to the framework (section 2), the various steps which the study involved are discussed one by one (section 3). They include defining the attributes of the product to be evaluated, setting up the relevant user profiles, designing and carrying out adequate tests, and collecting the results. The overall methodology is then generalised by briefly discussing its application to the evaluation of grammar checkers (section 4). In the conclusion, we claim that the TEMAA methodology provides a valid model for designing and carrying out a concrete evaluation, at the same time allowing for maximal reuse of concepts, methods and actual data from one evaluation to another.

2 The TEMAA evaluation methodology

2.1 What is evaluation in TEMAA?

Market products such as grammar and spelling checkers are typically subjected to *adequacy evaluation*, i.e. evaluating how adequate they are for the tasks which they are designed to perform. However, it is not enough to evaluate a system solely with respect to its design specifications. An evaluation must also take into account the context in which a system is to be used. The system must be evaluated with respect to the requirements of some real (or potential) users. Thus, at the most general level, we consider that to evaluate something is to determine what it is worth to somebody in a particular context. Evaluation can then be characterised as a function from objects and (potential) users of these objects to a value:

$$\text{objects} * \text{users} \rightarrow \text{value}$$

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

For such a function to be useful it is necessary to formalise the notions of *object* and *user*, in such a way as to enable rigorous and consistent evaluation of concrete instances of a particular object type.

2.2 Attributes, measures and methods

Objects of evaluation are decomposed into various attributes which can be independently tested and evaluated against user requirements, and then built up to an evaluation of the whole system. In the absence of standards specifically for LE products, the TEMAA project took as a starting point the ISO Standard 9126 'Quality Characteristics for the evaluation of Software' (ISO 1991), where quality characteristics are those attributes of a piece of software which are considered to be of potential interest to a user. The ISO standard identified six such high level attributes: functionality, reliability, usability, efficiency, maintainability and portability. Given that LE software such as spelling or grammar checkers tends to be generic and usually has facilities for defining user dictionaries and fine tuning grammar rules, we have also added 'customisability' as a top level attribute.

Each of these top level quality characteristics is subdivided into various attributes and sub-attributes. Such attributes will typically differ for different types of LE products, especially with respect to attributes like functionality and customisability, although certain attributes may be the same for different product types.

Attributes and their sub-attributes are organised hierarchically. In addition to the attributes of the top-level quality characteristics, there are a number of what might be called 'hard attributes' which do not easily fall into the ISO quality characteristics, but nevertheless, also occur at the top of this 'quality tree' such as the natural language(s) which the product treats, or its price.

Associated with each attribute is a measure for that attribute and a method for establishing the value for that particular measure. Depending on the attribute in question, measures can be of various types. The measures for certain attributes such as the functionality attributes of positive lexical coverage or error coverage, or the customisability attribute pertaining to the number of user dictionaries which can be loaded at one time, are factual numerical measures. Other (non-numerical) factual measures are, for example, the natural language which the product treats or the usability sub-attribute of which learning resources (on-line help, printed manual) are available. The measures for other usability attributes, however, are more subjective and their values are in terms of rating scales (TEMAA 1996).

Given the wide variety of measures, the methods for eliciting values for the measures (and thus for the attribute with which they are associated) also vary greatly. In our case study on spelling checkers, we have aimed at automating as many methods as possible, and this is quite feasible for functionality attributes as will be described below. However other methods are not so tractable to automation, so for example, in establishing values for some of the usability attributes, complex experiments involving real users may have to be set up.

All these attributes can only be evaluated with respect to user requirements, and

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

in the next section we give an overview of our approach to user profiling based on requirements.

2.3 User profiling

To define a *user* is to define a user of some specific type of object. For the purposes of evaluation, users are defined in terms of their requirements for each of the attributes of the object to be evaluated.

An important question, then, is how to discover these requirements. In the case of LE products detailed sub-attributes of a product may be couched in very specific and technical (linguistic) terms with which the user is unfamiliar. Whilst certain requirements such as the language treated, or the hardware platform(s) the system runs on, can be easily specified by users themselves, the designer of an evaluation (the evaluator) must be able to determine other user requirements without resort to direct questioning of the customer of the evaluation. Thus, we have defined a methodology for specifying user requirements for functionality based on some linguistic characteristics of the user.

Depending both on a user's inherent properties (such as their native language and proficiency in the language treated by the system) and the type of tasks they are performing, including the type of texts being generated and corrected, different attributes will have more or less importance for that user. Thus a very important aspect of user profiling is to define the relative weightings which should be given to particular attributes.

2.4 The Parameterisable Testbed (PTB)

The PTB is a concrete realisation of the formal evaluation framework established in the TEMAA project. In this section we provide a brief sketch of the main elements of the PTB. For a more detailed description see TEMAA (1996). The general functionality of the testbed is as follows. It is fed with descriptions of users and objects (including evaluation measures, and methods for testing the objects) and it outputs an evaluation of the object(s). It is termed a *parameterisable* testbed, since it is possible to add (or modify) new object descriptions, user profiles or measures and methods at any time. Objects can be any type of LE system, spelling or grammar checkers, information retrieval products, machine translation systems, etc.

The PTB itself consists of a number of different modules supporting the various actions involved in the evaluation process.

2.4.1 Defining objects and users

As we have seen, both objects and users are defined in terms of the component attributes and sub-attributes of an object type, and organised hierarchically. A sub-program of the PTB called the Evaluators Tool (ET) provides a graphical editor which supports the definition of both objects and users. In one mode, ET allows the designer of an evaluation to develop a hierarchical featurisation of an object, defining

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

all attributes and sub-attributes and typing their values. The resulting hierarchical featurisation of an object is called a *feature checklist*.

By switching to another mode the graphical editor supports the definition of user types. Basically, extra information pertaining to a user's requirements is added to a feature checklist producing a *user profile*. This information takes the form of constraints and weightings on system features. For non-terminal nodes in the quality tree, the relative importance of an attribute (with respect to its sister attributes) is indicated by differentially weighting attributes. In the case where all sub-attributes of a particular attribute are equally important then they receive the same weighting.

ET also performs the evaluations. Given a user profile and test results for an object, ET computes all terminal values and a root utility value.

2.4.2 Testing product instances

To obtain the values for each of the attributes of a specific product under evaluation, certain tests must be carried out. Given an object type and an instance of that type to be evaluated, the PTB can run all the tests associated with the attributes defined for the object type. All primitive attributes are listed in a file along with the name of the program for establishing their values. Tests can either be automated or partly or wholly carried out by hand. Thus some test measures will involve simply asking the evaluator to supply the value for an attribute, or possibly providing a 'recipe' for determining the value. Others will access another program (e.g. ASCC, described in section 3.3) which will carry out the test.

These tests are not dependent on a specific user type, and once they have been carried out for a specific instance of a product, the results in the form of a full set of attribute value pairs can be stored in libraries for subsequent evaluation against any number of user profiles.

2.4.3 The PTB libraries

An important feature of the PTB is its modularity, containing a number of different libraries which can be continuously updated. Maintaining such libraries helps in the ultimate goal of both standardising the evaluation process and allowing the greatest possible re-use of evaluation experience and materials built up over time, avoiding the costs of starting from scratch for each individual evaluation. Thus, libraries contain feature checklists, user profiles and the results of specific product tests developed for earlier evaluations. However, testing the functionality of LE products in a systematic and reliable way not only requires automation of tests whenever possible but also the construction of large-scale test suites. So tests and test suites are also stored in the libraries. Normally these libraries will be maintained off-line by computer scientists who write and maintain test programs or by experts in evaluation who design and construct the test suites.

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

3 Carrying out an evaluation of spelling checkers

In this section we look at the way in which a concrete evaluation procedure for the adequacy evaluation of spelling checkers can be designed and performed in a systematic way. The various steps involved in the process will be exemplified by referring to the case study on Danish spelling checkers.

Although the project has set up a detailed taxonomy of attributes pertaining to all the quality characteristics of spelling checkers, the case study has focused on the evaluation of functionality, which is the most interesting characteristic from an LE point of view.

3.1 Defining attributes of a spelling checker

The first step in the evaluation procedure is defining the attributes of the object of evaluation, complete with the type of value they take and organising them in a feature checklist. There are three main attributes of the functionality of a spelling checker:

- Positive lexical coverage (recall).
- Error coverage (precision).
- Suggestion adequacy.

The basic function of a spelling checker is to reject non-words. However, the checker should also correctly recognise legal words in the language and not flag them as errors. Therefore we also evaluate positive lexical coverage. Most spelling checkers on the market nowadays also provide suggestions to replace a detected error. Hence, we also evaluate the adequacy of the suggested replacements.

For each attribute, a number of sub-attributes have been specified.

3.1.1 Positive lexical coverage

The following sub-attributes of lexical coverage have been defined:

- Common word coverage.
- Closed sub-vocabularies (e.g. foreign loans).
- Technical sub-languages.
- Productive sub-vocabularies (e.g. numbers, dates).
- Acronyms, abbreviations, symbols.
- Proper names.
- Spelling variants.
- Multi-word units.

For each sub-attribute the measure for assessing a checker's performance, is either a weighted mean score or a simple percentage of words correctly accepted by the checker. Common word coverage and technical sub-languages, are measured in terms of a weighted mean score whilst the measure for the other attributes is a simple percentage.

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

3.1.2 Error coverage

Error coverage is decomposed according to different error types:

- typographical errors,
- spelling errors,
- medium-related errors (e.g. OCR errors).

For each type, the evaluation measure is the percentage of errors correctly detected out of a list.

The distinction between typographical errors (or mis-typings) and spelling errors (or misspellings) is one often made. Mis-typings can be seen as errors of performance, whilst misspellings can be regarded as errors of intention (cf. Hendry and Green 1993). Misspellings are harder to correct for spelling checkers, since they can distort the intended word in a more dramatic way than mis-typings. Medium-related errors such as OCR errors, are of yet a different nature as they create distortions based on the confusion between similar characters that a human writer would not produce.

The fact that these error types differ in their nature and complexity, and that different detecting and correcting techniques perform differently depending on the error type, is noted *inter alia* by Kukich (1992). Since the frequency of the various error types varies depending on the application and user population that a concrete evaluation will be addressing, a definition of the relevant error types must be part of the statement of user requirements.

Typographical as well as spelling errors can be formalised by using the same method, i.e. in terms of string manipulation. Thus, for instance, a typographical error may be:

- addition of a letter X to a letter Y where X and Y are close on the keyboard: (*bicycle* > *biocycle*).

And a spelling error in Danish could be:

- substitution of *n* for *nd*, e.g. *sandsynligvis (probably)* > *sansynligvis*

The TEMAA project has developed formal 'corruption' rules defining misspellings in English, French and Danish. These rules are used to generate errors automatically, and to test spelling checkers' error coverage and suggestion adequacy for the relevant error types. In this paper, we concentrate on the work done on misspellings by native speakers of Danish. However, the methodology this work illustrates should be applicable to spelling errors in general.

A general point to be noted is the fact that spelling errors can result in so-called real-word errors, i.e. strings which although wrong in the context are still legal words of the language[1]. Real-word errors can derive for example from the exchange of different forms of the same word (*speak* instead of *speaks*) or of different prepositions (*for* instead of *of*). Such errors require some knowledge of the context for a checker to be able to detect and cannot, therefore, be treated by a spelling checker alone.

[1] Kukich quotes a study in which it was found that real-word errors amounted to 40% of the errors in a corpus. It is unclear how generalisable this figure is, but in any case real-word errors are not a negligible phenomenon for text-proofing techniques.

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

3.1.3 Suggestion adequacy

The evaluation of suggestion adequacy also relates to the various error types listed above, since distinguishing among different error types makes it possible to tailor an evaluation to a particular user population. Furthermore, the evaluation measure must not only be the percentage of correct suggestions offered for each error type, but also take into account the number of alternatives the user is presented with, and the position of the correct suggestion within the list of alternatives.

Thus, we subdivide suggestion adequacy into the following sub-attributes:

- how often is the first suggestion the correct one?
- how often is a visible suggestion the correct one?
- how often are all visible suggestions wrong?
- how often are no suggestions presented at all?

Note that 'visible' above means visible on the screen without having to scroll down to see hidden suggestions.

3.2 Defining user profiles

As indicated above, a fundamental problem facing the evaluator developing a user profile is how to extract user requirements of a precise enough nature to be translated into constraints and weightings on system attributes. To avoid potential problems arising from users' introspective evaluation of their requirements, for functionality, users are defined in terms of the lexical characteristics of types of texts they write and the sorts of errors they make. In this case study, we have made the simplifying assumption that a spelling checker user is also the original writer of a text. In reality of course the user may be an editor of other people's texts. In such a case the user profile would be more complex since it would have to take into account the possibly different inherent properties (e.g. native language, level of proficiency etc.) of the writer and editor. However, we feel that this would not pose insurmountable problems and that our approach is also valid for those situations since the sorts of errors which the editor must correct must still be objectively defined whether they are correcting their own errors or someone else's.

In defining a user profile for the purposes of our case study, the vocabulary of the text types produced was assumed to be general Danish language whilst the sorts of errors produced were based on existing research into spelling errors made by Danish primary and high school pupils, carried by Andersen *et al* (1992) and Löb (1983). So the user group being profiled was that of Danish native speakers undergoing their basic formal education, writing general language texts. Danish spelling is characterised by a striking lack of correspondence between letters and phonemes, including a large number of silent letters. So it is not surprising that the greatest number of spelling errors are phonetically based. Andersen *et al* produce the following typology of errors:

- r-related errors,
- errors of suffixation (not involving r),

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

- silent letters,
- consonant doubling,
- letter substitution,
- compounding errors,
- errors in loan words,
- syllable omission or repetition,
- others.

From an analysis of the data (cf. TEMAA 1995a), around 30 discrete patterns of spelling errors have been identified and these patterns formed the basis for the corruption rules in our test suites described in section 3.4.

3.3 Methods for measuring spelling checker functionality

To measure the functionality attribute *lexical coverage*, the basic method involves running the spelling checker over a list of words and calculating the percentage of words correctly accepted by the checker. If the word list is structured according to frequency of occurence, a weighted mean score is calculated. Conversely, for *error coverage* the checker is applied to a list of spelling errors and the percentage of words correctly rejected out of the list is calculated. For *suggestion adequacy* the same list of errors paired with their intended correct form are used to determine whether the correct replacement is suggested and if so its position on the list of suggestions.

To test checkers on very large word lists in a consistent way, these methods have been automated. A demonstration tool called the Automated Spelling Checker Checker (ASCC) has been implemented by Stichting Taaltechnologie (cf. TEMAA 1996 for implementation details). ASCC simulates an interactive spelling checking session, running the checker over a word list and saving the results in a separate file and then calculating the relevant performance scores for each type of list.

However the construction of such large-scale representative test suites reflecting the requirements of a particular user type is also a non-trivial exercise as will be described in the next section.

3.4 Creating test materials

The test materials developed in TEMAA consist of word lists, where a word is either a simple word or a compound. Thus, as already discussed, we do not consider the context in which a word (valid or misspelled) occurs in since spelling checkers have no contextual knowledge.

We distinguish two kinds of word lists: lists of valid words ('base lists'), and lists of corrupted ones ('error lists'). Base lists are used to test a spelling checker's positive lexical coverage. They also serve as input for constructing error lists. Error lists are used to test the checker's error coverage. To test suggestion adequacy we use a pair of lists consisting of an error list together with the base list from which the errors have been generated.

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

3.4.1 Creating base lists

Different methods can be envisaged to construct base lists, e.g. using frequency word lists, lemmatised lists taken from existing published dictionaries, manually or automatically constructed samples. TEMAA has opted for frequency word lists where possible, i.e. where such lists were available or where it was possible to construct them within the project. We were interested in lists of inflected words, as words in running text occur as inflected full forms, and not all forms of the same lexeme are equally frequent. Furthermore, we wanted to be able to distinguish between different degrees of frequency. It is in fact very important for a spelling checker to be able to recognise the most frequent words of the domain and not to bother the user with too many false alarms. On the other hand, it may be a disadvantage for the checker to know the most infrequent words, as this may make it too prone to accepting errors which happen to result in infrequent words. Of course, it is difficult to know exactly where to draw the line between words (or more exactly, word forms) that should be recognised and those that should not, but Kukich appears to think that a dictionary of 60,000 domain relevant words is not too large.

Other factors that must be considered when constructing base lists are their representativeness of the domain and their correctness. To take the latter first, base lists must have been proofread to remove systematic typos and spelling errors. As for representativeness, this depends of course on the way in which the corpus from which the word list is derived has been assembled. Criteria to be met are variation on the one hand, and homogeneity on the other. The former is needed to avoid the systematic influence of individual texts or themes on the overall lexical material; the latter ensures that the characteristics of the corpus can be generalised to texts of the same type and domain.

Frequency word lists have been used in the project to test common word coverage in English, Italian and Danish. Their size varies from about 33,000 for Italian to about 6800 for Danish. The lists were divided into frequency classes, and testing results were provided in two different ways, either as simple percentage scores showing the proportion of words recognised by the checker for each class, or as a weighted mean score showing the overall performance for the list as a whole. The weights used correspond to the relative frequency of the various classes[2].

Also for technical sub-languages, we believe the frequency word list approach would be a fruitful one. So, for each technical domain of interest to the user, a separate word list should be used to test the spelling checker's lexical coverage.

In one particular case, namely words belonging to the closed part of speech classes, exhaustive word lists can be constructed. We have provided an example of such a list for Danish. The list was constructed automatically by extracting the relevant word forms from the electronic version of the Danish spelling dictionary (*Retskrivningsordbogen* (1986)).

For other sub-attributes of lexical coverage, including multi-word units, closed sub-vocabularies (e.g. foreign loans, archaisms, slang, obscenities), proper names,

[2] For more details on the individual lists, cf. TEMAA 1995b.

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

acronyms, initialisms, abbreviations, and symbols, the project has foreseen using limited samples, constructed manually or semi-manually. Three different lists were actually constructed to test coverage of loan words and proper names.

Finally, for other sub-attributes such as numbers, units of measure, and dates, the project has opted for the use of domain-related grammars to generate relevant samples.

In conclusion, the compilation of base lists provided ranges over a broad selection of lexical sub-attributes and constitutes a good exemplification of the methods set up by the project.

3.4.2 Creating error lists

Lists of corrupted words, or error lists, are generated automatically by applying so-called corruption rules to our base lists. Corruption rules are intended to simulate typographical or spelling errors. A corruption rule applies to a valid word and produces a corrupted word. This may, in turn, result in another valid word, so the output of corruption rules is checked to filter out real-word errors.

Common practice in most spelling checker evaluations (cf. Green and Hendry (1993)) is to use limited samples of particularly tricky words. Automation of the error generation process and of spelling checker testing, on the other hand, allows for the use of large error samples for the purpose of evaluation.

Note also that generating the errors automatically makes it possible to check the spelling checker's suggestion adequacy automatically. We saw earlier that suggestion adequacy is evaluated in TEMAA by testing whether the spelling checker produces the correct replacement for an error and, if so, in which position among all the possible alternatives the correct suggestion is. However, this presupposes that, given an error, it is always possible when evaluating the behaviour of the spelling checker to know what the correct replacement should be, or in other words what the intended word is. This is by no means always obvious, not even for a human being. Consider thus the following Italian spelling error, and a few possible replacements:

giaca > *gioca, giacca, giace* (plays, jacket, lies)

Clearly, at least in most cases, a human being would be able to disambiguate among the various possibilities when reading the text in which the misspelling occurs. An automatic evaluator would have to be at least more intelligent than the spelling checker it is evaluating in order to know what the correct replacement is. However, if the error is generated by applying a corruption rule to a valid word, the correct replacement can simply be taken to be the original word.

Let us now look more closely at corruption rules. A corruption rule is a collection of substitution statements in Perl style. Such rules are used by a Perl program called Errgen. Taking a word list and a rule file (containing a set of declarations and corruption rules), Errgen outputs a list of words corrupted according to the rules. As an additional facility, Errgen can check the output list against a list of valid words to make sure that the misspellings produced are not real-word errors. Corruption rules were written for English, Italian and Danish. To cover most of

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

the Danish spelling errors listed in section 3.2, for example, a set of 10 rules were written and used to generate error lists. They are listed informally below:

1. rer > re in word-final position, e.g. *revisorer (auditors) > revisore*
2. rer > r in word-final position, e.g. *kontrollører (conductors) > kontrollør*
3. et > ed in word-final position (and excluding the article *et*), e.g. *betragtet (regarded) > betragted*
4. delete r in between vowels, e.g. *Panduro > Panduo* (Panduro is the name of a Danish writer)
5. delete r when preceded by a vowel and followed by a consonant, e.g. *hierarkisk (hierarchical) > hierakisk*
6. nd > n, e.g. *sandsynligvis (probably) > sansynligvis*
7. r > rd, if not preceded by another r, e.g. *samvær (gathering) > samværd*
8. kk > gg, or k > g, e.g. *makke ret (behave) > magge ret*
9. ce > se, if not preceded by a c or an s, e.g. *nyancer (nuances) > nyanser*
10. ch > sj before a vowel if not preceded by an s, e.g. *chance (chance) > sjance*

Rule 2 turned out to generate far too many real-word errors, and was eventually taken out. By applying the remaining rules to our list of common Danish words, we generated a total of 4562 spelling errors that were used to test the error coverage and suggestion adequacy of two different spelling checkers, as described in the next section.

The number of errors is not equally distributed, in the sense that some rules apply more often than others and therefore create more errors than others. This effect could be balanced by weighting each rule differently depending on the frequency of occurence relevant to the actual user profile. However, information on error frequency at least for Danish is very sparse, and although weighting is foreseen in the TEMAA framework, realistic scores could not be set in the case study.

3.5 Testing and evaluating the systems

The testing phase of the case study (cf. TEMAA 1995b) consisted in running ASCC on two spelling checkers for Danish to test their linguistic functionality, i.e. lexical coverage, error coverage and suggestion adequacy. Since our purpose was to evaluate the soundness and usefulness of the methodology we had devised rather than the actual systems, we shall refer to them simply as Product A and Product B.

Lexical coverage was tested against a collection of word lists representative of a number of coverage sub-attributes. The results we show below concern the coverage of the general vocabulary of Danish, which was tested by way of a structured frequency word list. The source used was one of the lists in Maegaard and Ruus (1986) covering the domain of general interest magazines published over five years (1970–1974). The size is approximately 6800 words, covering about 82% of the corpus. The list is divided into three frequency classes, as shown in Table 1 (figures have been rounded up).

The list was chosen because, although not very large, it satisfied our criteria of representativeness and correctness. The corpus it is derived from, in fact, has been

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

Table 1. *The general vocabulary of Danish*

Class interval (number of words)	Relative frequency of each class (%)	Relative cumulative frequency
1–189	48	48
190–1732	21	69
1733–6780	13	82

Table 2. *Common word coverage results*

Product	Class interval (no. of words)	Relative frequency	Total no. of words	No. of words recognised	% of words recognised
A	1–189	48	189	186	98
	190–1732	21	1543	1527	99
	1733–6780	13	5048	4698	93
B	1–189	48	189	189	100
	190–1732	21	1543	1543	99
	1733–6780	13	5048	4783	95

constructed very carefully and the list itself has been proofread (cf. Maegaard and Ruus (1987) for more details). In spite of the relatively small number of words included in the list, the results obtained show a difference in the performance of the two products, as shown in Table 2.

Taking the weighted mean score as the total measure for common word coverage (where the relative frequencies of each class are used as weights), shows clearly and succintly that Product B scores better than Product A with 99% against 97%.

The same base list was used to derive spelling errors by applying the corruption rules listed above. We grouped the results relative to the 4562 errors submitted to the checkers into general error types[3]. From the results shown in Table 3 a remarkable difference can be observed in the behaviour of the two spelling checkers, where Product A, which scored less on common word coverage, does clearly better than Product B on error recognition. The figures indicate the percentage of errors recognised for each group.

For the errors that were recognised by the systems, the adequacy of the suggestion offered by the checker was tested according to the principles described above. Table 4 shows the results of the test. The error types used are the same as for error coverage results.

Again, the results show a noticeable difference between the two spelling checkers, with Product A in general scoring better than Product B. Furthermore, they also disclose the fact that, for *r*-related errors and other suffixation errors, both products

[3] These are actually a subset of the categories used by Andersen *et al* and listed in section 3.2.

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

Table 3. *Error coverage results*

Product	*r* errors[a]	Suffixation[b]	Silent letters[c]	Substitution[d]	Loan words[e]
A	100	100	99	100	100
B	75	100	96	96	100

[a] These errors are generated by rule 1 in the list given in section 3.4.2.
[b] Generated by rule 3.
[c] Generated by rules 4–7.
[d] Generated by rules 8.
[e] Generated by rules 9–10.

Table 4. *Suggestion adequacy results*

Product	Error types	No. of errors (%)	1st sugg. correct (%)	Correct sugg. among 2–4 (%)	Correct sugg. not among first 4 (%)	No sugg. (%)
A	*r* errors	16	19	81	0	0
	Suffixation	235	19	80	1	0
	Silent lett.	3160	81	11	3	5
	Substitution	1061	76	13	5	6
	Loan words	72	51	24	7	18
B	*r* errors	12	17	58	25	0
	Suffixation	235	1	34	65	0
	Silent lett.	3057	76	13	7	4
	Substitution	1025	83	8	6	3
	Loan words	72	81	10	1	8

score rather poorly on first suggestion correct. Thus, a user particularly prone to making spelling errors falling into either of these two categories would not benefit much from these two spelling checkers' suggestion menus.

The test results obtained in the case study have brought forth interesting differences between the two products, and offer, in our opinion, an informative and detailed picture of their linguistic functionality. Hence, they confirm the usefulness of the attributes and sub-attributes defined by the project as benchmarks for evaluating the functionality of spelling checkers.

4 Applying the TEMAA framework to grammar checkers

The TEMAA evaluation methodology is meant to apply not only to spelling checkers, but to LE products in general. In this section, we discuss briefly the attributes, methods and measures that can be used to evaluate a more complex type of system,

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

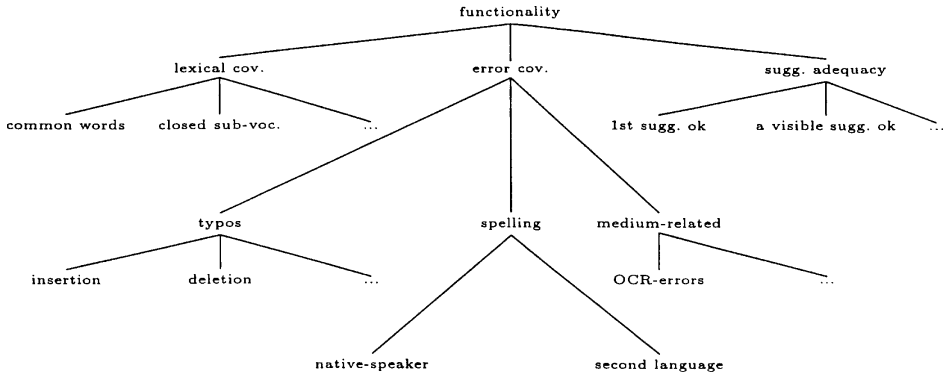


Fig. 1. Spelling checker attribute taxonomy.

namely grammar checkers. As in the case of spelling checkers, we shall focus on the evaluation of functionality.

The factors affecting functionality for grammar checkers are similar to those for spelling checkers. Thus, the functionality of a grammar checker will be assessed by measuring how well the checker treats correct text, detects errors and guides the user in correcting the errors. The results will then be evaluated against user requirements which, as in the case of spelling checkers, will be expressed in terms of language proficiency (native speaker vs. second language learner), text and error types. Due to the more complex nature of grammar errors, the methods that can be used to test the various functionality sub-attributes may not be so easily automated. However, automation of tests for certain error types may be possible.

We distinguish two basic attributes of functionality:

- problem checking; and
- reference lookup.

Whilst most users can be expected to be able to recognise a spelling error once it is flagged, this is not necessarily the case for grammar errors. Thus, under reference lookup we want to check the availability of additional material to support the user in verifying the grammar checker's diagnosis.

Problem checking attributes are very similar to the highest level of functionality attributes for spelling checkers, as can be seen by comparing figures 1 and 2. However, sub-attributes of suggestion adequacy for grammar checkers have not been defined. An evaluation of suggestion adequacy will have to take into the account the fact that many of the suggestions will not involve literal replacement suggestions, but rather a diagnosis of the type of error and possibly some general explanation to help the user to correct the error. Thus, a completely automated procedure to test suggestion adequacy is very hard to design.

To test grammatical coverage and error coverage, two sets of test materials will be necessary, a list of correct sentences covering all the constructions the checker is being tested on, and a list of similar sentences containing grammatical errors. The two test suites can be considered to be the converse of each other as was the case

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

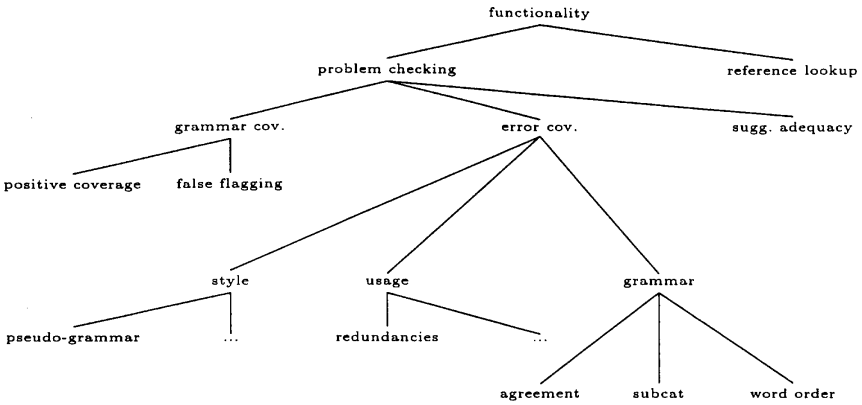


Fig. 2. Grammar checker attribute taxonomy.

of base lists and error lists for spelling checkers. However, generating the one from the other automatically is a lot more problematic, as it would require parsing of the correct sentences as well as a good formal model of grammatical errors. However, generating a list of ungrammatical sentences manually need not be less rigorous (cf. Balkan *et al.* 1995).

The ability of grammar checkers to detect grammatical errors typically deteriorates in long and syntactically complex sentences. Therefore, both test suites should be built in such a way as to reflect different degrees of syntactic complexity. This varying complexity can be reflected in the measures for grammar coverage and error coverage. Thus for grammar coverage, a recall measure of the number of accepted sentences out of the total number of correct sentences can be provided for each context, along with a precision measure of the number of false flaggings out of the total number of flaggings. For error coverage, we are interested not only in the complexity dimension, but also in the dimension of the error types shown in figure 2 under error coverage (for more details, cf. TEMAA 1995c). Thus, an error coverage measure can be provided for each error type and in each syntactic context. This multidimensionality makes it possible to assign weights to either error types or contexts depending on the user requirements of a specific evaluation.

5 Conclusions

In this paper we have sketched the evaluation methodology developed within the TEMAA project, and we have reported on a concrete package developed within the TEMAA framework for the evaluation of Danish spelling checkers. We believe that this concrete case study validates the overall framework in a number of ways. The brief discussion on how to apply the same methodology to the evaluation of grammar checkers helps to confirm its usefulness and flexibility.

The overall TEMAA approach to evaluation of LE products is characterised by being very much user oriented and decompositional in nature. Thus an object of evaluation (in our case a spelling checker) is featurised in terms of its component

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

attributes which may be of interest to a potential user. Such a structured approach has guided our development of a specific (albeit partial) evaluation package for spelling checkers, both in specifying the sub-attributes relevant for spelling checkers and also the definition of tests and test materials necessary for such an evaluation.

By decomposing functionality into very specific positive coverage and error types, and testing spelling checkers' performance on these sub-attributes, we have been able to reveal differences in the performance of different spelling checkers, which otherwise might not have been apparent. Such decomposition thus provides good benchmarks in the evaluation of spelling checker functionality. The framework also allows for weightings on attributes to reflect an attribute's importance to a specific user population. Thus in our case study based on native Danish speakers writing general language texts, we were able to weight different parts of the general vocabulary according to their frequency of occurence.

Automation has made it possible to test systems on large data sets, thus enabling a more rigorous and thorough-going evaluation. However, automation will not be feasible to the same extent when evaluating more complex systems such as grammar checkers. The modularity built into the framework and its implementation in terms of the PTB allows for re-usability, at all levels, ranging from the re-use of product type definitions (feature checklists) and user profiles to the re-use of tests and test suites. Such re-usability opens the way in the future for obtaining consistent evaluations across different product instances and user populations, whilst avoiding the often prohibitive costs of developing an evaluation from scratch for an individual customer.

TEMAA has been a relatively small project and our experience both in the case study and developing the general framework itself, points to a number of areas of work to be done. As pointed out by Manzi *et al.* (1996), the current framework (and the PTB implementation) assumes that evaluation is entirely compositional, which is clearly not the case with LE products which involve such a lot of human-machine interaction, so that e.g. the acceptability of a product involves a trade-off between usability and functionality.

Moreover, the use of weighting offered by the framework has not been fully exploited and validated. In our case study, we have not been able to make full use of the facilities for the weighting of attributes relating to error coverage, due to lack of existing data on error frequency in Danish. The collection of such data, as well as profiling of other user populations, would constitute a project in its own right.

References

Andersen, T., Heltberg, E., Jensen C., Olivarius, P. & Stefansson L. (1992) *Stavefejl - en undersøgelse af stavefejlsmønstre i gymnasiet*. Undervisningsminsteriets Gymnasieafdeling, Denmark.

Balkan, L., Fouvry, F., Arnold, D., Regnier-Prost, S., Dauphin, E. and Lux, V. (1995) *TSNLP User Manual, Volume 1: Test Suite Design, Methodology*. University of Essex, UK.

EAGLES (1998) *EAGLES Evaluation of natural language processing systems*. Final Report. EAGLES Document EAG-EWG-PR2.

Falkedal, K. (ed). (1994) *Proceedings of the evaluators' forum*. ISSCO, University of Geneva.

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press

Green, T. R. C. and Hendry, D. G. (1993) Spelling mistakes: How well do correctors perform? *Adjunct Proceedings of InterCHI '93*.

Harman, D. (1995) Overview of the second text retrieval conference (TREC-2). *Information Processing and Management*, Special Issue on the Second Text Retrieval Conference, pp. 271–289.

Hutchins, W. J. and Somers, H. L. (1992) *An introduction to machine translation*. Technical report no. 291, University of Cambridge Computer Laboratory. Academic Press.

ISO (1991) ISO/IEC9126 *Information technology – software product evaluation – quality characteristics and guidelines for their use*. First Edition 1991-12-15.

Kukich, K. (1992) Techniques for automatically correcting words in text. *ACM Computing Surveys*, **24**(4): 377–438.

Löb, H. (1983) *Bogstavet r - det største staveproblem*. Danmarks Pædagogiske Institut, Copenhagen.

Maegaard, B. and Ruus, H. (1986) *Hyppige ord i danske aviser, ugeblader & fagblader*. Gyldendal, Copenhagen.

Maegaard, B. and Ruus, H. (1987) The composition and use of a text corpus. *Studies in Honour of Roberto Busa S. J.. Linguistica Computazionale*, **4–5**: 103–122. Giardini Editori e Stampatori, Pisa.

Manzi, S., King, M. and Douglas, S. (1996) Working towards user-oriented evaluation. *Proceedings of the International Conference on Natural Language Processing and Industrial Applications (NLP+IA 96)*, pp. 155–160. Moncton, New-Brunswick, Canada.

Retskrivningsordbogen (1986) Dansk Sprognævn. Gyldendal, Copenhagen.

Sparck Jones, K. and Galliers, J. R. (1996) *Evaluating natural language processing systems. Lecture Notes in Artificial Intelligence 1083*. Springer-Verlag.

Sundheim, B. M. (1991) Overview of the third message understanding evaluation conference. *Proceedings of the third message understanding conference (MUC-3)*, pp. 3–24. San Mateo, CA.

TEMAA (1996) *TEMAA Final Report*, LRE-62-070.

TEMAA (1995*a*) *Deliverable 12,* Test Materials, LRE-62-070.

TEMAA (1995*b*) *Deliverable 13: An Experimental Application of the TEMAA Evaluation Framework: Spelling Checkers*, LRE-62-070.

TEMAA (1995*c*) *Deliverable 14: An Evaluation Model for Grammar Checkers*, LRE-62-070.

Thomson, H. (1992) *The strategic role of evaluation in natural language processing and speech technology*. Record of the ESPRIT DANDI/ELSNET/HCRC Workshop. Human Communication Centre, University of Edinburgh.

https://doi.org/10.1017/S1351324998001995 Published online by Cambridge University Press