A Language-Sensitive Text Editor for Dutch

Gerard Kempen
Theo Vosse
University of Nijmegen

Abstract
Modern word processors begin to offer a range of facilities for spelling, grammar and style checking in English. For the Dutch language hardly anything is available as yet. Many commercial word processing packages do include a hyphenation routine and a lexicon-based spelling checker but the practical usefulness of these tools is limited due to certain properties of Dutch orthography, as we will explain below. In this chapter we describe a text editor which incorporates a great deal of lexical, morphological and syntactic knowledge of Dutch and monitors the orthographical quality of Dutch texts. Section 1 deals with those aspects of Dutch orthography which pose problems to human authors as well as to computational language sensitive text editing tools. In section 2 we describe the design and the implementation of the text editor we have built. Section 3 is mainly devoted to a provisional evaluation of the system.

1. Some Dutch spelling problems

The three hardest problems of Dutch orthography are easily explained in terms of a comparison with English, French and German:

1. As in English, there are quite a few *homophonous spelling patterns* whose occurrence in specific words is difficult to remember. For instance, the sound /eI/ is spelled as *ei* in some words but as *ij* in other ones. In addition, proper names often spell the same sound as *y, ey* or even *eij*. And there are four common ways of rendering the diphthong which sounds like the one *in Eng. how: ou, au, ouw and auw*. The consonant /k/ is spelled as either *c, cc, k* or *kk,* and as *ck in* many proper names. And many authors forget when to reduplicate a vowel or a consonant (e.g. double *d* and *l* in *onmiddellijk, Eng. immediately).*

2. As in French, various frequent *inflectional suffixes* are *homophonous.* For example, the verb *gebeuren* (Eng *to happen)* has a third-person singular present-tense form *gebeurt* which sounds exactly the same as the past participle *gebeurd.* Due to word-final devoicing, the endings *-t* and *-d* are both pronounced /t/. If the verb stem ends in *-d* (also pronounced /t/), the third-person *-t* suffix is obligatorily added although it cannot be heard (as in the passive auxiliary *wordt,* Eng *is).* Thus, the third person singular is orthographically distinguished from the homophonous first-person

word (Eng *am),* which has no suffixes. Since the past-participle suffix is not written after verb stems ending in /t/ (e.g. *besteed* instead of **besteedd,* Eng. *spent),* there are three possible spellings of word-final */t/: -d, -t* or *-dt.* In order to select the correct alternative, the author must be able to perform a partial parse of the sentence, e.g. identify word classes, finite verbs, participles, person and number of subject noun phrases.

The following examples further illustrate that Dutch orthography often depends on syntactic context. The verb *besteden* (Eng *to spend)* is homophonous with three other spellings, thereby forcing upon the author a choice between four alternatives:

- *Bestede (inflected* past-participle).
- *Besteden*(infinitive; present-tense plural).
- Besteedde (past-tense singular).
- *Besteedden (past-tense* plural).

Notice that, although the past-participle suffix *-d* is always deleted, the *d* forming part of the past-tense suffix *-de, is* retained. The choice between homophonous forms *besteed* and *besteedt* is dependent on person of the subject and, in case of second person, on word order of subject and finite verb:

Ik besteeD het geld.	I spend the money.
Je besteeDT het geld.	You spend the money.
BesteeD je het geld?	Spend you the money (Do you spend ...).
BesteeDT je broer het geld?	Spends your brother the money. (Does your brother ...).

The last sentence shows that one cannot simply rely on the presence of *je* after the finite verb: *je* is either personal *{you)* or possessive *(your)* pronoun.

The *o* in *auto* (Eng. *car)* is doubled in the diminutive form *(autootje)* but not in the plural *(auto's)* where an apostrophe is mandatory. In all three word forms the *o* is pronounced as the vowel in Eng. *road* rather than *rod.* The plural form of nouns ending in *-ie* is either *-ieën* o*r -iën* depending on whether or not the word's final syllable is accented *(copieën* versus.*poriën;*Eng. *copies, pores).* Again, the pronunciation presents insufficiënt clues.

3. As in German, compound nouns, verbs and adjectives are written as single words (e.g. *wegverkeer, autorijden, druipnat;* Eng. *road traffic, car driving, soaking wet).* The first problem ensuing from this spelling rule is probably caused by interference from English: Dutch compounds are often written as separate words, especially when the compound has low frequency or is a neologism composed by the author.

The third problem derives from two morphological rules which sometimes apply during the compounding process. One rule is the insertion of the shwa phoneme between certain constituent words. A case in point is the second e in *perebloesem,*

the Dutch equivalent of *pear blossom* (Du. *peer/peren* = Eng *pear/pears;* the bracketed *e inpe[e]rebloesem* disappears due to another rule). Shwa insertion only occurs if the first constituent word is singular; if it is a plural noun, one simply writes the plural without an extra shwa, e.g. *perensiroop* (Eng *pear syrup).* However, the difference between *pere-* and *peren-* is not reflected in the pronunciation. The official Dutch-Belgian spelling regulations include an extensive list of composita and simplicia in their correct spelling: the 1954 'Groene Boekje' ('green booklet'). Many words are listed with two legal spellings, one of which designated as the word's *Voorkeurspelling* (preferential spelling). Authors wishing to avoid errors or non-preferred spellings such as *perenboom* or *perestroop* need to consult this list.

Another morphological rule inserts the phoneme /s/ between constituent words, e.g. *regeringscrisis* (Eng. government crisis). In quite a few compounds this *s* is mandatory even though its pronunciation is obscured by a subsequent *s* or *z*. For instance, *regeringssteun* (Eng. *government support)* includes an inaudible *s* in analogy with all other compounds having r*egering*-cum-audible-*s* as initial member *(regeringsgebouw, regeringsprogramma;* Eng. *government building, government program).*

The convention to write compounds as single words has implications for hyphenation. *Between* the word forms that make up a compound, there is always a syllable boundary. *Within* these word forms, the normal phonology-based syllabification rules apply. (Any inserted *e* or *s* is counted as belonging to the preceding word form, e.g. *carnavals-hit* instead of **carnaval-shit.)* However, the existence of a morphology-based hyphenation rule can easily lead authors astray. They are sometimes lured into the assumption that other aspects of morphological structure will be mirrored in hyphenation as well. But in Dutch, derivational and inflectional processes do not interact with hyphenation (except for diminutives). For instance, although the plural noun *openingen* (Eng. *openings)* contains the morphemes *open+ing+en,* standard syllabification rules nevertheless divide it into *o-pe-nin-gen.*

2. Design and implementation of the text editor

2.1. Design goals

The text editor we have built attempts to identify and correct the kinds of errors described in Section 1 as well as two further error types: unintentional *word reduplications* and *punctuation errors.* So it does not pretend to be a fully-fledged spelling, grammar and style checker.

Word reduplications may arise when the author modifies word order in a sentence by copying a word or phrase but forgets to erase the text at the original position. In Dutch, this easily happens in the course of transforming subordinate into main clauses, or vice-versa, because the finite verb occupies different positions in these clause types (roughly: Subject-Object-Verb in subordinate clauses; Verb-Subject-Object in interrogative main clauses; and Subject-Verb-Object in other main clauses).

Other factors are responsible for unintended *contiguous* word reduplications, that is, two adjacent occurrences of the same word—often a function word.

The rules for punctuation in Dutch are very similar to those for English. Typically Dutch are conventions such as the following.

- Never capitalise truncated function words starting with an apostrophe ('s, 'n, *'t);* if they are the first word of a sentence, capitalise the next word instead.
- If a word beginning with the diphthong *ij* is capitalised (e.g. in a proper name or sentence-initially), write *IJ* instead of *Ij*.
- Write a comma between two consecutive finite verbs (the first one usually belonging to an embedded clause, the second one to the embedding clause).

Punctuation problems also arise in the context of abbreviations and proper names (e.g. *KLM* vs. *K.L.M.* vs. *Koninklijke Luchtvaart Maatschappij* vs. *Koninklijke Luchtvaartmaatschappij* vs. *Kon. Luchtv. Mij).* Although there are not many official rules here, authors will at least strive for consistency within a document. We therefore developed a module which helps authors to maintain consistency in their treatment of abbreviations, proper names, numbers, dates, measure phrases, etc.

We emphasise once again that our language-sensitive editor aims at solving *orthographical* problems. Since Dutch orthography is intertwined with syntax in various ways, the system could not function properly without at least a superficial syntactic analysis of the text. This does not imply that the system is a full syntax checker, though.

A final design consideration concerns interaction with users. The system is targetted on texts written by educated authors who are (near-)native speakers of Dutch. Because it makes substantial demands on computational resources and could not run on microcomputers, we have opted for a non-interactive version which processes texts in batch mode[1]. System output consists of a corrected copy of the input text with error diagnostics printed in the margin. A few special editor commands enable the author to inspect corrections and glosses, and to undo or modify any corrections.

2.2. System design

Texts loaded into the editor are processed in four steps:

(1) preprocessing,
(2) word level analysis,
(3) sentence level analysis, and
(4) text resynthesis.

The editor's main linguistic components are depicted in Figure 1.

[1] In earlier publications we have described our work on an *interactive* language-sensitive text editor for Dutch. See Kempen *et al.* (1987) and Vosse (1989).

The *lexicon* is the heart of the system. It contains almost 100,000 lemmas from the authoritative Van Dale Dictionary of Contemporary Dutch. For each lemma, it lists the citation form as well as all current inflected forms ('lexemes'). In total we have about 250,000 word forms. Pointers specify the mutual relationships between each lemma and its lexemes, and between preferred and non-preferred spellings of a word. For the purpose of spelling error correction (triphone analysis; see below), we have computed and stored the pronunciation(s) of every lexeme. The syntactic information in the lexicon also originates from Van Dale and has been enriched in collaboration with the Nijmegen Centre for Lexical Information (CELEX; see Van Gaalen, 1989). Parser and compound analyser need access to the syntactic category and the morphosyntactic features (number, gender, person, case) of every word form. Essential to the parser's functioning is the information about the syntactic valency of lexical entries. Therefore the lexicon includes rudimentary descriptions of the sentence context required by lemmas (in particular the number and phrase type of objects taken by verbs, adjectives and nouns). The lexicon software incorporates interactive tools for semi-automatically adding new words and their inflectional paradigm, and for loading specialised lexical databases, e.g. geographical names or terminology banks.

The *pre-processor* cleans up the input text as follows:

- It removes any characters which do not belong to the text proper, e.g. editing and mark-up commands.
- Characters which do not belong to the set known to the linguistic components, are replaced.
- It corrects punctuation.
- It inserts special symbols marking the beginning and end of abbreviations, proper names, dates, measure phrases, certain frozen idioms which would foul up the parser. These symbols specify syntactic phrase type (e.g. noun phrase, prepositional phrase), thereby simplifying subsequent sentence level analysis.

The pre-processor works on the basis of a string pattern matcher and a dictionary of string patterns to be checked. The pre-processor is an interactive program run by the author. It scans the text for incorrect or undesirable strings. Having discovered a flawed piece of text, it corrects it automatically upon approval by the author. If there are several possible alternatives for corrections, these are shown on the screen. The system then inserts the alternative selected by the author.

Word level analysis comprises a threefold task. First, a list is compiled of all different words ('types') occurring in the text as a whole. This *lexical inventory* obviates the necessity of repeating the full treatment of a word for all its individual occurrences ('tokens'). Second, all words on the list are subjected to a spelling check according to a new technique developed in our institute: triphone analysis. See Van Berkel and De Smedt (1989) for a detailed description.

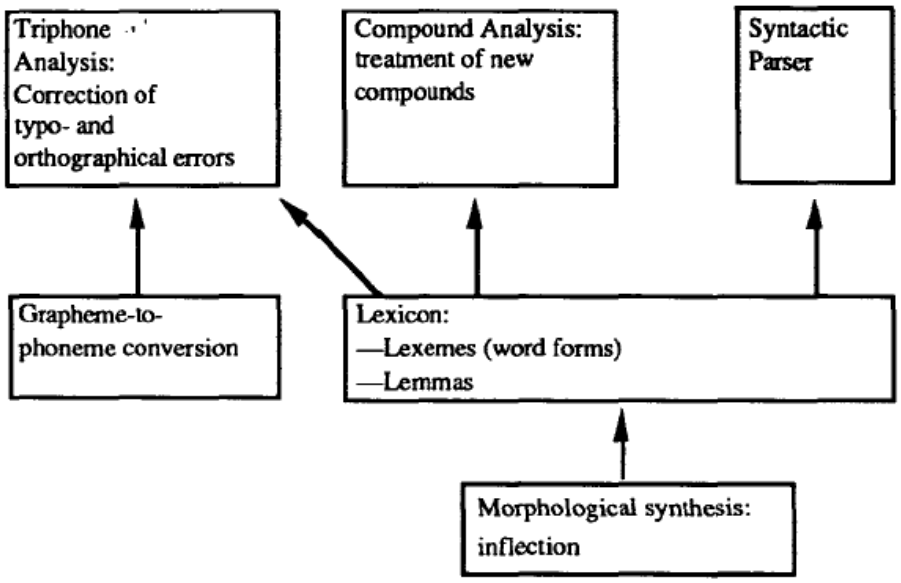

Figure 1. Linguistic components of the text editor

This technique handles both 'look-alikes' (typographical errors, e.g. letter reversals, insertions, omissions) and 'sound-alikes' (incorrect but homophonous spellings). Triphone analysis requires a phonological representation of input words, which is provided by the grapheme-to-phoneme converter (Daelemans 1987). Because a spelling error may very well produce another legal Dutch word (e.g. *zeil =>ziel*; Eng. *sail/soul),* triphone analysis is applied to *all* words listed in the inventory. The result is a small set of correction alternatives (in the order of 1 to 5) for most of the words. If no suggestions for correction are found, the word is probably an unknown compound or proper name. Third, all character strings in the lexical inventory which are illegal (i.e. cannot be looked up in the on-line lexicon) undergo *compound analysis.* The system tries to interpret these non-words as sequences of two or more word forms that do occur in the lexicon. If this attempt succeeds, the result is added to the list of correction alternatives for the string.

The compound analyser deploys some morphological knowledge of Dutch, in particular concerning the combinability of word forms having certain morphosyntactic features, and about their possible change of appearance in a compound (insertion of shwa or /s/; cf. section 1). Parenthetically, there is no need to decompose word forms into morphemes: the elements making up a compound are themselves word forms having 'wordhood' status.

Sentence level analysis. The syntactic parser does not attempt to reconstruct the sentence structure precisely as intended by the author. Since the editor's goal is to monitor orthographical rather than syntactic quality of documents, many aspects of syntactic structure become irrelevant. This applies, for instance, to the attachment

of prepositional phrases and, with the exception of subjects and finite verbs, to the assignment of grammatical function to major clause constituents.

Sentence structures are computed by a Multiple LR(1) parser—a member of the family of shift-reduce parsers—similar in design to the one developed by Tomita (1986). A set of about 500 augmented phrase-structure rules (see next paragraph) was converted to a non-deterministic push-down automaten. Parse trees constructed by this automaton (usually several trees for one sentence) are efficiently represented in memory as 'packed forests'. Spelling errors in the leaves of a parse tree are recognised, diagnosed and corrected by two special mechanisms as follows.

To begin with, the non-terminal symbols of the phrase-structure grammar have been augmented with feature matrices. For example, with symbols S, NP and VP might be associated a matrix specifying number and person features. When the automaton reduces NP and VP to S (cf. the rule S=>NP+VP), it applies a special *unification* algorithm to the NP and VP matrices. Unification succeeds not only in the case when all features match, but also when there are feature violations. In the latter case, a diagnostic flag is attached to the resulting subtree. For instance, if someone types the erroneous sentence *Peter word bedankt* (Eng *'Peter am thanked'),* the NP and VP matrices specify different values for the person feature (NP 3rd, VP lst). Unification nevertheless 'succeeds' and the S node is marked as violating subject-verb agreement. Later on, during text resynthesis, this diagnostic message triggers the replacement of *word* by third-person *wordt* (Eng. *is)* as the desired alternative.

The second technique for dealing with spelling errors utilises special 'error rules' included in the grammar. For instance, there is a rule saying that a clause may consist of two consecutive main clauses: S=>S+S. However, whenever this rule is applied, an error flag is put up at the top S and a warning about incorrect punctuation is put ready.

The parser's diagnostic component employs a system of numerical error weights in order to obtain a quality ranking within a 'forest' (i.e. the set of alternative parse trees computed in response to an input sentence). Any messages, including automatic corrections, are based on the highest-quality tree (least errors).

The parser plays a dual role in the selection of correction alternatives to be communicated to the user. First, if a sentence contains a non-word, the parser is capable of determining which correction alternative(s) fit(s) into the syntactic context. Second, when it fails to produce a fully correct parse tree for a sentence with all words legal, it attempts to construct one by using the correction alternatives.

Text resynthesis. Basing on the results of word and sentence level analyses, the system corrects all words of the document and copies the resulting text to a log file, together with diagnostic messages. The corrections take into consideration certain spelling standards requested by the author in advance, e.g. the preferential spelling as promulgated in the 'Groene Boekje' (cf. section 1), or the punctuation rules in

some style sheet known to the system (cf. abbreviations, capitalisation of names of organisations, etc). The file also contains potential hyphenation points at positions licensed by earlier compound analysis. The author can inspect the log file, evaluate the corrections and undo or modify the ones that are out of place.

3. System evaluation and conclusion

At the time of writing this paper, no detailed evaluation of the language-sensitive editor as a whole is available. However, the individual linguistic modules have been tested extensively over the past few years. We refer to Daelemans (1987) for some quantitative results concerning compound analysis and hyphenation; Van Berkel and De Smedt (1988) present data on triphone analysis. These tests make it amply clear that, although not perfect, the performance of our modules is far better than anything currently available on the market. The parser has not been seriously tested yet because its implementation was finished only recently.

In order to get a first impression of the performance of the integrated system, we have subjected it to an informal Dutch spelling test used to select typists and secretaries. The test consists of 150 sentences, their lengths varying between 3 and 9 words. The total number of words (tokens) was 895 (not counting 10 commas, 15 question marks, 4 exclamation marks and 131 full stops). The text contained 482 different words (types).

In every sentence one word has been underlined, and it is the subject's task to decide whether or not that word contains a spelling error. In fact, 75 sentences are correct. The maximum time to complete the test is 10 minutes. If no more than 5% of the decisions are incorrect, the score is considered 'good'; more than 10% errors means 'failed'.

The 75 to-be-detected spelling errors can be divided into three categories:

- Nonwords (n=59); one of them concerns a punctuation problem (illegal position of apostrophe), three address a diacritics problem (missing or illegal diaeresis), 13 have a verb conjugation problem (illegal but homophonous suffix) and 42 are homophonous misspellings of various other types.
- Legal words which do not fit into the syntactic context (n=14).
- Fossilised multiword idioms spelled according to obsolete orthographic rules (n=2); an example is *te allen tijden* (wrong) instead of *te allen tijde* (correct; Eng *at all times); tijden* is a legal Dutch word which tends to replace the obso.lete *tijde* in this expression).

None of the items address the problem of writing compounds as single words or hyphenation.

The editor program (written in C) was running on a DECstation 3100. It took the spelling test without any words underlined. Whenever it discovered a (real or

putative) spelling error, it replaced it by the single best correction alternative it could find. No other correction alternatives were logged. The text had not been treated by the pre-processor.

Examination of the log file revealed the following results. The editor scored 140 correct decisions. It overlooked 10 misspellings and gave no 'false alarms'. The 'missing hits' were distributed over the above three categories as follows:

- Of 59 non-words, 56 were detected and 55 were corrected as intended by the test.
- Of 14 legal words incompatible with syntactic context, nine were recognised as such and seven were corrected as intended by the test.
- The errors in two fossilised idiomatic expressions were both overlooked.

The editor did not detect any errors in the legal words which were *not* underlined in the original test version.

The 150 sentences took 13.7 minutes of CPU time: 7 for triphone analysis, compound analysis and text resynthesis, and 6.7 for syntactic parsing. Although this is more than the maximum granted to human subjects, one should realise that the editor could not profit from the fact that errors were limited to underlined words, so that it had to pay equal attention to the non-underlined words. Moreover, the human subjects did not correct the misspellings they detected.

The 10 wrong decisions clearly reveal the system's design limitations. Two morphosyntactic errors in the input text were overlooked because they presuppose semantic analysis of the input sentence.

- Ze rende met z'n allen naar huis. They all ran home.
- Vroeger beïnvloeden zijn ouders... Formerly his parents influence...

In (1), the expression *met z'n allen* (Eng. *all*) forces a plural interpretation upon the subject pronoun *ze,* which is ambiguous between Eng. *she* and *they*. This,l in turn, stipulates plural *renden,* homophonous with singular *rende,* as the correct form of the finite verb. In test item (2), the homophonous past-tense form *beïnvloedden* (with two *d's)* is entailed by the adverb *vroeger* (Eng. *formerly).* Since we do not intend to build any semantic knowledge into the editor, such problems will remain beyond the scope of our editor. Two other errors were overlooked because word-level analysis construed them as legal compounds: *erflijk* was interpreted as *erf+lijk* (Eng. *heritage corpse)* rather than as a misspelling for *erfelijk* (Eng. *hereditary)* and the editor did not recognise *eigelijk* as a typo for *eigenlijk* (Eng. *actually)* but as *ei+gelijk* (Eng *egg-equal).* Such errors, too, are unavoidable in the present design.

We were very pleased with the high error correction rate. Of 65 repair attempts 62 were successful at the first try. Of the three mis-hits, one originated from an infelicitous choice by the triphone analyser, and both other ones from a wrong syntactic parse.

Since there is ample room for improving the editor's diagnostic capabilities and for speeding up its performance, we believe we are close to a language-sensitive editor which can substantially enhance the efficiency and quality of typesetting and proof-reading Dutch texts.

Acknowledgements

We thank the present and former members of the Language Technology Project of NICI for their extensive practical and theoretical contributions to design and implementation of the text editing tools described in this paper. We wish to mention in particular Gerben Abbink (pre-processor), Walter Daelemans (compound analysis and grapheme-to-phoneme conversion; presently at ITK, Tilburg), Brigit van Berkel (triphone analysis; now at TNO-ITI, Delft) and Rob Heemels (syntax rules; Océ-Nederland, Venlo).

References

Daelemans, W. (1987) *Studies in Language Technology. An object-oriented computer model of morphophonological aspects of Dutch*. Dissertation, University of Louvain.

Tomita, M. (1986) *Efficiënt Parsing for Natural Language: A Fast Algonthm for Practical Systems*. Kluwer Academie Press, Dordrecht.

Van Berkel, B. and De Smedt, K. (1988) Triphone analysis: a combined method for the correction of orthographical and typographical errors. *Proceedings of the Second Conference on Applied Natural Language Processing.* American Association for Computational Linguistics, Austin TX.

Van Gaaien, M. (1989) *De syntactische valentie van werkwoorden, zelfstandige naamwoorden en adjectieven.* Paper, NICI, University of Nijmegen.