Learning from Errors. Systematic Analysis of Complex Writing Errors for Improving Writing Technology

Cerstin Mahlow

Abstract In this paper, we describe ongoing research on writing errors with the ultimate goal to develop error-preventing editing functions in word-processors. Drawing from the state-of-the-art research in errors carried out in various fields, we propose the application of a general concept for action-slips as introduced by Norman [39]. We demonstrate the feasibility of this approach by using a large corpus of writing errors in published texts. The concept of slips considers both the process and the product: some failure in a procedure results in an error in the product, i.e., is visible in the written text. In order to develop preventing functions, we need to determine causes of such visible errors.

1 Introduction

Published texts by native-language, expert writers—or more generally: finished texts, i.e., when the author has no intention to carry out further revisions, but submits the text to the reader or publisher, even if carefully proofread by the author—still contain numerous errors. Examples are sentences without finite verbs (*Dabei es vielfältige Arbeitsbereiche.* [missing *gibt* as second word], *Sie sich viel Zeit für den Bettenkauf.* [missing *Nehmen* as first word]), duplicate words (*Wir fliegen wie geplant am 30. Oktober nach Kairo fliegen* [last *fliegen* should be removed], *This was former camel trading venue was home to the 1982 stock market crash in Kuwait, which wiped out many billions in regional wealth at the time.* [first *was* should be removed]), superfluous words (*like when you're sitting at a someone else's computer*), or agreement errors (*Von der grauhaarigen Frau sieht Christin nur zuckenden den Rücken.*) [should be *den zuckenden Rücken*]).

In a US-wide study on error types in native-language college-student writing (using final drafts of essays), Lunsford and Lunsford [32] in 2008 found similar errors

Cerstin Mahlow
University of Stuttgart Germany e-mail: cerstin.mahlow@ims.uni-stuttgart.de

as Connors and Lunsford [3] had identified in a comparable study 20 years before. In the late 1980s, students still wrote the essays by hand (or with a type-writer), and thus could not use any automatic checkers. Twenty years later, students used word processors with spelling and grammar checkers. The number of spelling errors had thus decreased dramatically—from over 30 % to only 6.5 % of all errors. However, apart from punctuation issues, Lunsford and Lunsford [32] still found a similar amount of "wrong words," "vague pronoun references," "subject-verb agreement errors," "missing words," "unnecessary shifts in verb tense," "fused sentences," or "sentence fragments" [32, p. 795], a clear indication that these errors cannot be detected and corrected by automatic checkers. Additionally, the error rate in 2008 with 2.29 errors per 100 words is very similar to the error rate in 1988 with 2.26 errors per 100 words.

Table 1 Error types as reported by Connors and Lunsford (1988) and Lunsford and Lunsford (2008) [3, 32], error types found in both studies are marked in bold.

	Connors and Lunsford (1988)		Lunsford and Lunsford (2008)	
Rank	Error Type	%	Error Type	%
1	**No comma after introductory element**	11.5	**Wrong word**	13.7
2	**Vague pronoun reference**	9.8	**Missing comma after an introductory element**	9.6
3	**No comma in compound sentence**	8.6	Incomplete or missing documentation	7.1
4	**Wrong word**	7.8	**Vague pronoun reference**	6.7
5	**No comma in non-restrictive element**	6.5	Spelling error (including homonyms)	6.5
6	Wrong/missing inflected endings	5.9	Mechanical error with quotation	6.4
7	Wrong or missing preposition	5.5	**Unnecessary comma**	5.2
8	**Comma splice**	5.5	Unnecessary/missing capitalization	5.2
9	**Possessive apostrophe error**	5.1	Missing word	4.6
10	**Tense shift**	5.1	Faulty sentence structure	4.4
11	Unnecessary shift in person	4.7	**Missing comma in a non-restrictive element**	3.8
12	**Sentence fragment**	4.2	**Unnecessary shift in verb tense**	3.8
13	Wrong tense or verb form	3.3	**Missing comma in a compound sentence**	.6
14	Subject-verb agreement	3.2	**Unnecessary/missing apostrophe including *its/it's***	3.1
15	Lack of comma in serie	2.7	**Fused (run-on) sentence**	3.0
16	**Pronoun agreement error**	2.6	**Comma splice**	2.9
17	**Unnecessary comma with restrictive element**	2.4	**Lack of pronoun-antecedent agreement**	2.7
18	**Run-on or fused sentence**	2.4	Poorly integrated quotation	2.7
19	Dangling or misplaced modifier	2.0	Unnecessary or missing hyphen	2.5
20	***Its/it's* error**	1.0	**Sentence fragment**	2.4

Table 1 gives a comparison of the 20 top-most identified error types and percentages of total errors. As spelling errors had been identified the most common error by far in the 1988-essays, they had been excluded in this listing, however, in 2008,

they made rank 5 with only 6.5 % of all errors. This change is a clear indicator that using spell checkers helped students correct most of the spelling errors. However, almost two third of the errors identified as frequent in 1988 still are in the list of frequent errors in 2008—marked bold in table 1. The order changed, but "wrong words," "no comma after introductory element," and "vague pronoun reference" are the most common errors, summing up to ca. 30 % of all errors. Here, grammar checkers obviously didn't help students.

When produced by skilled writers, these errors can be considered performance errors rather than competence errors. The fact that they appear in published texts—see the collection in appendix A of [34]—indicates that these errors cannot be automatically detected and corrected by state-of-the-art grammar checkers, as Gubelmann [19] confirmed in a small-scale study for German errors. In the last decade, various studies [59, 26, 48] pointed out general deficits of checkers for English and German in larger studies.

In language-learning scenarios, the focus should be clearly on competence errors and how to advise students. For experienced writers, performance errors are more common, so a different approach is needed for detecting errors and providing appropriate feedback. It would be even better to provide editing functions helping writers to *prevent* performance errors. As shown in [34], a large variety of editing functions based on NLP sources could be developed, the results clearly depend on the quality of the underlying resources. However, from a writing research perspective, the question remains which of these functions are really useful, which kind of errors could be prevented indeed, and how to distinguish performance errors from competence errors.

In this paper, we propose the use of a well-defined error typology. We first discuss the state-of-the-art in error research in several fields in section 2, and illustrate relevant methods in writing research, NLP, and writing technology in section 3. In section 4 we then present the application of the general-purpose concept of *slips* proposed by Norman [39] to writing and in section 5 we illustrate some examples. Using Norman's typology, we can distinguish competence errors from performance errors, and we can determine the best starting point for the design of preventive editing functions.

2 Research on Errors

Errors in writing are investigated in various research fields. In the field of *writing research*, there is only little published research on errors in the last decades (except for learning to write in a second language), for example [3, 53, 32]. However, all studies try to systematize the errors found in writing samples; they take the written evidence as starting point and develop coding schemes. These studies aim to compare writing abilities of students and to design pedagogically motivated writing strategies to help students become better writers. They do not investigate how these errors have been produced nor how to prevent these errors by means of improved

writing technology. A common research question in writing research is to explore how writers react to given or produced errors—thus a behavioral focus is chosen.

Mostly, each study comes up with an ad-hoc coding scheme, making it difficult to compare findings across studies. As an illustration consider table 1: Although one of the authors (Andrea Lunsford) was involved in both studies and the study from 2008 clearly was intended to provide a follow-up, the authors used different namings for (probably) identical error classes—e.g., "no comma after introductory element" vs. "missing comma after an introductory element."

In *applied linguistics* and *second language acquisition*, research on errors started with the collection of learner data in the late 1980s. The aim was to advance the understanding of language acquisition to develop better pedagogical approaches tailored to language learners' needs. However, even recent studies concerned with cognitive aspects of errors do not focus on the process, but on the produced language units (see for example [18, 47]). In the context of the online available Falko corpus[1], Lüdeling et al. [31] argue for annotation on various levels, to be able to distinguish competence errors and performance errors.

They also aim to come up with an explanation for an observed error, but restrict this to meta-explanations referring to the learners language biography like "transfer problems" [31], which clearly refer to competence errors only. This is in line with earlier research, restricting possible explanations to psycholinguistic, sociolinguistic, epistemic, or discourse structure sources.

In error annotation in learner corpora, we find various coding schemes, which raises problems for systematic analysis and the application of NLP methods. As Meurers puts it:

> There is so far no consensus, though, on the external and internal criteria, that is, which error distinctions are needed for which purpose and which distinctions can reliably be annotated based on the evidence in the corpus and any meta-information available about the learner and the activity which the language was produced for. [35, p. 6]

In the field of *computational linguistics*, most of the research on errors is concerned with students (i.e., writing novices) writing in their first or second language (for an overview, see [27])—a similarity to error analysis in writing research and language acquisition. However, the goals are different: Researchers code errors in learner corpora to improve checkers [17], to develop applications for computer-assisted language learning [36], or to make syntactical parsers more robust [14]. Coding schemes are generally developed from scratch (for an overview, see [8]) without reference to the coding schemes developed in writing research.

However, an agreed-upon standard on error annotation would be a prerequisite for the development of any writing support in learning scenarios or in tutoring systems [35]. Also in computational linguistics, error analyses for experienced writers—i.e., skilled authors writing in L1 and familiar with the topic, genre, and tools involved—are rare. One of the few examples are Napolitano and Stent [37], who work on a personalizable writing assistance program for advanced learners of

[1] https://www.linguistik.hu-berlin.de/institut/professuren/korpuslinguistik/forschung/falko

English. The target group are thus L2 writers, but familiar with topic, genre, and tools.

Neither writing research, applied linguistics, nor computational linguistics draw on more general research on errors, which is an important topic in *psychology*. With respect to performance errors, in the early 1980s, Norman [39, 40] proposed a very strong theoretical framework for the classification of *action slips*:

> Call the highest level specification of a desired action an **intention** [...]. An error in the intention is called a **mistake**. An error in carrying out the intention is called a **slip**. [40, p. 254][2]

The concept of slips combines the process and the product: some failure in a procedure results in an error in the product. Currently error analyses in writing research, applied linguistics, and computational linguistics focus exclusively on the product: the text produced is explored manually or automatically to find and code errors to deduce error correction and to propose better language-learning approaches. The same strategy is used by checkers: they process written text to detect errors and suggest corrections. How this error might have been produced is not considered.

According to Norman, slips are frequently caused by the design of the tools used: when the user has to carry out a complex sequence of operations to achieve a goal, this will often result in cognitive overload [40]. Many slips can thus be prevented by better designed tools, which offer functions operating in line with the users' cognitive model and thus reduce cognitive load. To develop such tools to support writing, a detailed and systematic analysis of writing errors and their causes is needed, but has not been carried out so far.

Although the different nature or competence errors and performance errors is addressed in earlier studies in applied linguistics and second language acquisition, e.g., by Corder [5] and Ellis [10], research in this area focuses on competence errors. Surprisingly, we find a divergent terminology. Competence errors ("mistakes" according to Norman) are called "errors"—and are in the focus of research—while performance errors ("slips" according to Norman) are called mistakes—and as Corder puts it: "Mistakes are of no significance to the process of language learning" [5, p. 167]. Additionally, those "mistakes" are not easily distinguishable from "errors" [10, p. 50-54]. With our approach to not only look at the failure in the product, but to consider also the failure in the procedure as inherent in the concept of "slips", we can address this issue. For our purposes, we stick with the terminology proposed by Norman, i.e., mistakes vs. slips.

[2] Emphasis in the original.

3 Relevant Areas

3.1 Research on Writing Processes

When writers use electronic writing tools, writing process data can be collected using keystroke-logging tools. Van Waes et al. [57] and Sullivan and Lindgren [55] provide a good overview of keystroke-logging technology. Originally, logging tools were developed to allow for better analysis of revising operations carried out by writers. The first large-scale study was performed by Flinn in 1987 [12], who logged writers using the Milliken Word Processor [54] with the logging program COMPTRACE over the period of a year.

Today's keystroke-logging tools, such as Inputlog[3] [30], can record all user actions in an unobtrusive way. This has lead to an increase in writing studies investigating writing in the workplace or in other natural writing situations, e.g., [44, 43, 58, 28].

However, manual inspection and interpretation of logging data is a demanding task. In the 1990s, S-notation was developed as a standard notation for encoding the evolvement of text—i.e., insertions, deletions, cursor movements, and pauses [52, 25]. Data from keystroke-logging tools can be transformed into S-notation data automatically. Based on this encoding, it is possible to understand which operations have been carried out by the writer. S-notation may also serve as input for tools that play back the writing session.

In S-notation, the writer's revisions are coded directly in the text. For each insertion or deletion, a break ("|") marks the last position before the action; insertions are enclosed by curly braces, deletions are enclosed by square braces. An index for each of break, insertion, and deletion, indicates the order of those actions. For logging data as presented in the example in figure 1, manual inspection of the S-notation data is possible. However, for real-world data as shown in figure 2, manual inspection is not feasible.

S-Notation:
`Th[r|1]1e q{u}2ick|2 brown [dog]5{f}5|6{[i]7|8{o}8|9x}6|7 jumps over the [{old }4|5]lazy [d|3]3[fox]9|{dog}10|11.|4 The end[. |12]12!`

Produced text:
The quick brown fox jumps over the lazy dog. The end!

Fig. 1 Example writing process data presented in S-notation

The extract in figure 2 shows the creation of one sentence by a journalist encoded in S-notation. The data has been collected between September 2006 and March 2007 as part of 120 writing sessions from 15 journalists from two Swiss broadcasters

[3] `http://www.inputlog.net`

(SRF and RTS). The finished texts comprise TV news items ranging from 30 to 270 seconds. [45] The final product produced here is the rather short sentence:

> *Doch: Fast jedes dritte Medikament, das der Basler Pharmamulti auf den Markt wirft, stammt gar nicht aus der eigenen Forschung, sondern wurde während der Entwicklungsphase eingekauft – von Uni-Labors oder kleineren Biotechfirmen.*

```
Doch: {Fast}148|149[J]149|150{j}150|151edes dritte
[Medikament]199|200{[Präparat]627|628}200|201{Medikament}628|629, das
{de192 [s]192 |193{r }193|194Basler Pharmamulti[s |192]194|195}191,
[Novartis]195|196 auf den Markt wirft, [ist gar |39]39stammt gar
nicht aus der eig[he|40]40enen Forschung, sondern wurde [zu einem
bestimmten Zeitpunkt41[ von |41]41 ]174 |175{[eingekauft|176]176,
[eingekauft|177]177, [zu einem bestimmten Zeitpunkt ]641|642
{während der Entwicklungsphase}642 |643 eingekauft -}175|178
von [Uni[versitäten]201|202]204|205 [{[f]205|206{F}206|207
[a|203]203orschern}202|204 {der Unis}207|208]208|209 {Uni-Labors }209|210
[doer|42]42oder [anderen [P|43]43Firmen]80|81 {kleine[re]179|180n
[Pharma- oder]180|181 Biotechfirmen}81|82 [eingekauft]178|179.
```

Fig. 2 S-notation example (from writing session `sf zvz 070118 2150 keller novartis snt 1`)

Writing processes can also be visualized by progression analysis as a path through the evolving text, as developed by Perrin [42, 43, 44], and used in various studies (see for example [46, 9]). This representation is also used to collaboratively develop writing strategies together with writers. However, on a more technical level, there is a lack of appropriate analysis and visualization algorithms to be able to investigate writing processes on various levels with different granularity for large amounts of keystroke-logging data.

While S-notation is designed for representing natural language text, it only encodes insertion and deletion of *strings of characters*; it does not take into account *linguistic units*. Leijten et al. [29] were the first—and up to now the only—researchers who applied natural language processing (NLP) tools to S-notation data for texts in Dutch and English, enriching them with basic linguistic information in order to facilitate higher-level analyses of the data.

3.2 Research on Writing Technology

Research on tools and automated support for *programmers* is an ongoing topic (e.g., [24, 41, 1]). However, as shown in a research survey in [34, pp. 57–63], research on tools for *writers* of natural language text and their influence on the writing process stopped in the 1980s; projects like RUSKIN [60] or the Editor's Assistant [7] did not result in actual products, systems like AUGMENT [11]—which was in some respects more advanced than today's word processors—disappeared.

Automated linguistic support for writers available in today's mainstream word processors (e.g., Microsoft Word), is restricted to grammar and spell checking; while technically more advanced (see [21]), conceptually it does not go much beyond what tools like the UNIX Writer's Work Bench [33] offered in the 1970s.

As can be seen at conferences like "COMPUTERS AND WRITING" or in journals like "KAIROS"[4], writing research for the most part treats writing technology as a given that cannot be changed; the focus is thus on creative uses of its affordances in order to discover or develop new kinds of writing beyond the production of alphabetical text.

3.3 Incremental Parsing

Applying NLP to writing poses two main challenges: first, the text is growing during writing, so that newly written parts have to be added to the parse tree, and revised parts have to be updated. Second, the text is still unfinished. Parsing has to be robust, so that it can handle ill-formedness, incompleteness, and inconsistency (the "I3", as Van De Vanter [56] calls them). These two challenges are mostly investigated separately in computational linguistics.

For parsing text as it is created, *incremental parsers* are used. Incremental parsers in the field of computational linguistics analyze their input word by word and start to construct a parse tree immediately. Incremental parsing is typically (though not exclusively) used in the context of acoustic speech recognition; approaches for incremental parsing therefore usually assume a linear evolvement of text. However, writing is not incremental in the sense that speech is incremental; in contrast to speakers, writers *can* actually go back and alter the text produced so far. This means that written text does not evolve linearly, and reparsing is needed if earlier text is changed. In order to avoid costly reparsing, it would be preferable to modify the parse tree built-up so far. This extended understanding of incrementality is well known in computer science, in particular in the context of interactive programming environments, e.g., [4]. It is only rarely addressed in NLP, though; one such example is Wirén [61]. In general, incremental parsers for natural languages focus on handling linearly expanding text, e.g., [49, 6, 38, 22].

Research on *robust parsing*, i.e., on parsers that are able to handle ungrammatical input gracefully, is often done in the context of applications such as processing of learner language or user-generated content [27, 13], and, of course, grammar checking [23, 21, 2]. The combination of robust and incremental parsing is mainly researched in the context of speech recognition [15, 16]; Schulze [50] shows the implementation of a robust parser applying an incremental grammar formalism (Left-Associative Grammar [20]) outside the domain of speech processing.

[4] `http://kairos.technorhetoric.net/`

4 Approach

4.1 Research Goals

The spelling and grammar checkers in today's word processors are useful tools, since authors tend to overlook errors in their own writing; instead, they read what *should* be there [44]. However, even state-of-the-art checkers are not capable of detecting and correcting *all* errors. Preliminary research indicates that many of the remaining errors can be explained as action slips. Norman [40] has demonstrated that the occurrence of slips can be minimized by improving the design of systems. The question posed here is thus: How can such non-detectable errors in writing be *prevented* by more ergonomic writing tools, which use online analysis of the writing process?

Our research aims to develop language-aware editing functions integrated into word processors that help to prevent errors. Such functions will be based on a systematic analysis of complex writing errors and interactive incremental NLP resources. We follow a holistic approach, combining methods and resources from writing research, computational linguistics, document engineering, software engineering, and XML technology, thus making this research entirely interdisciplinary. To avoid costly re-implementation of core functionality writers expect (e.g., copy-paste, general navigation, undo function), we rather integrate new functions into existing word processors, but do not build new ones from scratch.

In general, the development of writing technology should (a) be based on insights from writing research—projects like RUSKIN failed, because they did not take into account actual user needs—, (b) make use of NLP methods and tools—since the 1980s, a lot of effort went into the development of NLP resources and tools, so the overall situation today is much better than several decades ago[5]—, and (c) benefit from development in hardware—programs like Epistle from IBM [23] were not commercially successful, because they were computationally too expensive at that time. Here, we address all three obstacles for earlier projects: actual writing process data is used as basis for the implementation, NLP resources today are of a reasonable quality and can be executed fast enough to be integrated into real-world applications running on laptops.

We focus on experienced L1 authors writing in German, where most of the errors could be classified as performance errors—authors would detect the errors themselves when reading the respective text some time later or after a media break (e.g., reading on paper instead on screen or using a different layout or font). Mechanical errors that can be reliably detected and corrected by standard grammar and spell checkers, e.g., sentence-initial capitalization, are not in the scope here.

[5] As an example, in the project *Integrated language tools for writing and document handling* at KTH Stockholm (`http://www.nada.kth.se/iplab/langtools/`), the development of linguistic search and editing functions was planned but had to be postponed first and was skipped in the end, because the required NLP resources would have to be developed in the first place.

For errors that can reliably be detected and corrected by state-of-the-art checkers, there is no specific need to prevent them. For errors that are caused by poorly designed tools resulting in cognitive overload of the writer, we should distinguish errors that would be preventable by improving writing tools and errors writers could easily take care of themselves with some support from the word processor. This will allow the development of language-aware functions for writing support, with an emphasis on interaction with the writer and avoiding patronizing.

4.2 Slips in Writing

We start with a detailed analysis of errors, from which we can create a fine-grained taxonomy of writing errors, applying the concept of slips. The scientific value with respect to errors is two-fold: (a) It is the first extensive systematic application of the Norman [39] classification of slips to writing, and (b) it is the first error analysis to take into account the *process* of writing by exploring keystroke-logging data.

We use the error corpus collected in the LingURed project[6] as starting point and systematically apply the categorization of slips. The corpus consists of over 200 errors from German texts published in printed or electronic newspapers, books, and advertisements, which contain typical errors not detectable by checkers (the examples used here are from this collection). The corpus data is annotated manually on various levels, e.g., error classes as described by Lunsford and Lunsford [32]; correct and incorrect sub-parse trees; erroneous syntactical elements like agreement error wrt. adjectives in noun phrases as instance of the class "agreement error," missing finite verbs as instance of "missing units," and stranded verb particles as instance of "extraneous units." Based on this annotation, we can develop heuristics for identifying errors. For example, to identify duplicate verbs, as in the example *Wir fliegen wie geplant am 30. Oktober nach Kairo fliegen*, a useful heuristic is the presence of multiple occurrences of the same word form in a sentence.

Additionally, the general design principles for avoiding slips proposed by Norman [40] can be transferred to editing functions. Some of the classes Norman [39] proposes can easily be applied to writing errors.

In the rest of this section, we present an overview of applicable classes of slips. The original names and explanations by Norman [39] are in italic. For each class we give a general example from everyday live Norman used in his proposal and we then give examples from writing taken from our error corpus with an explanation on how this slip had been produced. All examples are German, glosses are given where necessary.

[6] `http://lingured.info`

Unintentional activation: when schemas not part of a current action sequence become activated for extraneous reasons, then become triggered and lead to slips

- *Data-driven activation: external events cause activation of schemas*
  - The most common one is the Stroop phenomenon, where names of colors are presented in a color that differs from the name, so participants have difficulties saying the color the word is printed in. [39, p. 9]
- *Capture error (When a sequence being performed is similar to another more frequent or better learned sequence, the latter may capture control)*
  - When counting something, you fall into a common counting schema, i.e., instead of counting page numbers, it goes like counting cards (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King) [39, p. 8]
  - An example in writing is the mixture of the processes of copy-and-paste and cut-and-paste. Both are very similar and a common failure is to copy something to a different point in text when it should have been cut-and-pasted. So the text appears both at the new place and the original one.
    Example from the newspaper "Uckermark Kurier", October 1, 2010, page 21:

    > *Noch gibt es keinerlei Vermutungen, wer in den neu entdeckten Gräbern bestattet wurde. In zwei Wochen werden die archäologischen Arbeiten an den Gräbern gestoppt. Dann werden Umbauarbeiten beginnen. In zwei Wochen werden die archäologischen Arbeiten an den Gräbern gestoppt. Dann werden umfassende Umbauarbeiten beginnen. Die Bischofskirche wird bis 2014 für rund 30 Millionen Euro saniert.*

    The sentence *In zwei Wochen werden die archäologischen Arbeiten an den Gräbern gestoppt.* appears twice. The following sentence *Dann werden umfassende Umbauarbeiten beginnen.* does not have the adjective *umfassende* in the first place, so here some additional revising took place.
    Another example from the newspaper "The New York Times", Article selected for Süddeutsche Zeitung, July 19, 2010:

    > The great hope for robots, said Patricia Kuhl, co-director of the Institute for Learning and Brain Sciences at the University of Washington, great hope for robots, said Patricia Kuhl, co-director of the Institute for Learning and Brain Sciences at the University of Washington, is that with the right kind of technology at a critical period in a child's development, they could supplement learning in the classroom

Loss of activation: when schemas that have been activated lose activation, thereby losing effectiveness to control behavior

- *Forgetting an intention (but continuing with the action sequence)*
  - You start going to some place but once there, you forgot what you wanted to do there and try to come up with an arbitrary justification. [39, p. 9]
  - Due to misdesign of word processors, it is often hard to find a certain point in the text. A common situation is to plan a revision at a certain point—the author

has a clear idea about what to inspect or change there, but he only vaguely remembers the point in the text. He has to find an action sequence to find this point (by scrolling or invoking a search function). After navigating to a certain point in the text, the author forgot what to change here and most probably does not carry out the intended revision. Here a better navigation in the written text and probably a different kind of displaying or presenting already written text is needed. Severinson-Eklundh [51] started to address such issues.

- *Misordering the components of an action sequence (including skipping steps and repeating steps)*
  - Starting the engine of a car although you already drive or forgetting to put water in the coffee machine. [39, p. 10]
  - Following an example from my own writing:

    > *Die Entwickler von von* RUSKIN *stellten erst beim Ende des Projektes feststellten, dass die Autoren ganz andere Funktionen gewünscht und gebraucht hätten.*
    > ('The developers of of RUSKIN discovered at the end of the project discovered, that the authors would have needed totally different functions.')

    to discover in German is expressed with a verb with a separable particle (*feststellen*). The finite verb of the main clause occurs twice, although once in an incomplete form (the particle is missing for *stellten*). The complete form *feststellten* is in the last position of the first part, indicating that this would not be a main clause (which would be possible with *stellten ... fest*). So if *stellten*—because of he missing particle—would be removed, it would result in an ungrammatical sentence. Removing the finite verb in the wrong position (*feststellten*) would result in an incomplete main clause. This sentence had been produced by splitting a longer version into two parts and then revising it—trying to correct the verb position. Here, help with splitting sentences would be necessary. Additionally, we have a duplicate *von* ('of'), also caused by a revision action.

False triggering: A properly activated schema is triggered at an inappropriate time

- *Falls triggering of acts among the things currently active in mind*
  - Trying to put down your glasses although not wearing them. [39, p. 7]
  - In writing, a common slip is to paste something from the clipboard without having copied it to the clipboard in the first place. In the best case scenario, the clipboard had been empty before, so the writer sees that nothing has been pasted and will go back to actually copy (or cut) the desired text part. In the worst case, the clipboard contains some text from an earlier action which will be pasted. The writer will not actually read the pasted part as he already knows what should be there, so this slip will go unnoticed.
- *Spoonerism (Reversal of event components)*

  - Reversal of words or parts of
    "You have tasted the whole worm" instead of "You have wasted the whole term" [39, p. 10]
  - The example is from the book "Nur Engel fliegen höher" by Wim Westfield, published in 2008, page 13:

    Von der grauhaarigen Frau sieht Christin nur zuckenden den Rücken.
    ('From the gray-haired woman, Christin only sees tremoring the back.')

    The noun phrase *zuckenden den Rücken* is in wrong order, it should be *den zuckenden Rücken.* Here, the adjective *zuckenden* had been added later and had been positioned at the wrong place.

- *Blends (Combinations of components from two competing schemas)*
  - Merging *close* and *shut* into *clut*. [39, p. 10]
  - From the newspaper "Süddeutsche Zeitung", July 16, 2010, page 16:

    George Washington heißt bei den Haudenosaunee seitdem Hanadahguyus, Stadzerstörerenn sie einen Brief an den Präsidenten schreiben, dann lautet die Anrede stets: Dear Hanadahguyus!

    Here, the form *Stadtzerstörerenn* is a blend from *Stadtzerstörer. Wenn*.

- *Triggering of schemas meant only to be thought, not to govern action*
  - Say something you intend to mention later. [39, p. 10]
  - In writing, this results in writing the next word in the sentence before the currently needed one, resulting in missing words like in this sentence from the newspaper "Uckermark Kurier", March 22, 2010, page 5:

    Die Terrorismus und Spionage zuständige Ermittlungsbehörde prüft, ob der Anfangsverdacht einer geheimdienstlichen Straftat vorliegt
    ('The investigative authority terrorism and spying checks, whether there is a initial suspicion with respect to intelligence crime.')

    Here the preposition *für* is missing after the determiner *Die*. This kind of slips occurs when writers "think faster than they can type."

Failure in triggering: When an active schema never gets invoked

- Wondering why person A didn't make coffee—you forgot to ask.[39, p. 11]
- When writers pause during sentence production they may reread already written text to check for proper use of anaphora. As they have read the pronoun already, they skip writing it down, resulting in a missing word (the pronoun) in the final sentence. Another example are non-finished revisions like replacing one word by another without deleting the first one as in this example from my own writing:

  In Abschnitt 2.1.2.1 sind wir auf Taxonomien von Redigieroperationen eingegangen, die mehrheitlich aus der Beobachtung von Änderungen in Texten entwickelt wurden, jedoch nicht die ursprüngliche Redigierabsicht Absicht der Autoren berücksichtigen.

Hier two versions of *purpose*, i.e., *Absicht* and *Redigierabsicht*, occur directly one after the other. I don't remember which one I liked most in the end, but I played around with both versions to see which one would read better in the final sentence. And then I forgot to delete one of them.

As we can see from these examples, writing errors that can be categorized as "missing words" or "duplicate" words on the surface level may have various causes. For proper handling—in the best case: preventing these slips—we need to know the actions carried out before the slip. We might not be able to always detect the intention of the writer, however, after having identified certain action patterns, it will be possible to set up experiments triggering those slips where we know the writers' intention beforehand.

Additionally, it will be necessary to investigate keystroke-logging data with respect to pausing information to distinguish performance errors due to revision from misspellings. The latter are often corrected immediately by the writer and can thus be identified by using typing speed. Considering the length of inter-key intervals also helps determining boundaries of linguistic units. From a cognitive point of view it is important to closely relate pausing and revision behavior when analyzing non-linearity. This allows for the isolation of motoric actions.

5 Examples

The following examples illustrate the research process starting with the error analysis.

5.1 Example 1

Let us consider the following example from the corpus:

Alle Studierenden dokumentieren und reflektieren die vorgegebene Kompetenzen
('All students document and reflect on the given [singular] competencies [plural]')

The phrase *die vorgegebene Kompetenzen* is ungrammatical as there is no agreement between determiner, adjective, and noun. If it reads *die vorgegebene Kompetenz* (singular) or *die vorgegebenen Kompetenzen* (plural), the noun phrase would be correct. As the surface of the determiner is ambiguous with respect to number, no change is needed for the determiner. Although it looks like a typo—assuming that the *n* as last letter of the adjective was forgotten—an inspection of the keystroke-logging data (in S-notation) shown below reveals that it is actually a revision error:

```
Alle Studierenden dokumentieren {und reflektieren}[1] [fünf|3][2]
{die}[3] vorgegebene|1 Kompetenzen. Dabei|2
```

The author writes *Alle Studierenden dokumentieren fünf vorgegebene* then stops and adds *und reflektieren*. Then she finishes the first sentence with *Kompetenzen.* and begins the next one with *Dabei*. There she stops again and deletes *fünf* after which she adds *die* (probably she wasn't sure about the exact number but still wanted to express that there would be several competencies).

The agreement error could have been prevented if the writer would have been made aware that changing *fünf* into *die* affects a complex noun phrase; this could be done by highlighting the syntactical structure she is editing, helping to focus on side-effects of the revision. The system could also suggest the writer to automatically apply the needed changes, possibly allowing selection from a list of changes, as in this case the intended number may be either singular or plural.

5.2 Example 2

Another error-prone operation is replacing one term in a text by another term. One strategy is deleting the term to be replaced and then typing the new term. However, some writers prefer to make use of letters already typed. Thus, when replacing *Prüfung* 'examination' by *Überprüfung* 'test' they delete the capital p, and then type *Überp*. In S-notation this may be coded this way:

```
[P|2]1{Überp}2rüfung
```

This is a very complicated approach, as a result we encounter errors like a forgotten *p* as in *Überrüfung* or—having forgotten to delete the *P*—in *ÜberPrüfung*, or a duplicated *p* as in *ÜberpPrüfung*. Again, these errors could be explained as typing errors—especially *Überrüfung*—, we can be sure about the reasons only by looking at the keystroke-logging data.

To prevent these kind of replacement errors, an appropriate query-and-replace function should be used. Today's word processors offer search-and-replace functions, but these functions operate on character sequences only and do not take into account that natural language text consists of word forms. For inflectional languages like German, replacing a word by another involves searching for all word forms of the query word and then replacing it by the corresponding word form of the replace word, i.e., both forms have to have the same morphosyntactic features.

6 Conclusion

The main goal is to support efficient writing by developing appropriate writing technology based on a strong theoretical basis developed in this project: a systematic classification of complex writing errors. For the first time, an error analysis considers both *the product*, i.e., the text where the error is visible for a reader, *and the process*, i.e., the editing operations causing this error. This analysis will lead to a

deeper understanding of the relation between cognitive aspects of the writing process and the affordances of the tools used. We develop functions to improve word processors by preventing complex writing errors. These language-aware functions for writing support will be implemented with an emphasis on interaction with the writer and avoiding patronizing. For errors that can reliably be detected and corrected by state-of-the-art checkers, there is no specific need to prevent them.

The scientific impact is on the theoretical level (1 and 2) and on a more practical level (3 and 4):

1. This research opens a new field in error research in writing by a systematic and empirically sound analysis of product and process as two aspects of errors which are intrinsically tied together
2. We propose a methodological framework for the classification of complex writing errors to be used as starting point for improvement of writing technology.
3. We develop methods and tools for analyzing and visualizing large-scale keystroke-logging data based on XML technology.
4. We implement language-aware editing functions based on incremental interactive NLP resources, to be integrated into word processors that help to prevent errors by offering appropriate support to write efficiently.

References

1. Candillon, W.: XQuery development in the Cloud(9). XQuery code anywhere, anytime. In: XML Prague 2013, pp. 221–236. University of Economics, Prague, Prague, Czech Republik (2013)
2. Clément, L., Gerdes, K., Marlet, R.: A Grammar Correction Algorithm: Deep Parsing and Minimal Corrections for a Grammar Checker. In: P. Groote, M. Egg, L. Kallmeyer (eds.) Formal Grammar. 14th International Conference, FG 2009, Bordeaux, France, July 25-26, 2009, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 5591, chap. 4, pp. 47–63. Springer, Berlin/Heidelberg (2011). DOI 10.1007/978-3-642-20169-1_4
3. Connors, R.J., Lunsford, A.A.: Frequency of Formal Errors in Current College Writing, or Ma and Pa Kettle Do Research. College Composition and Communication **39**(4) (1988)
4. Cook, P., Welsh, J.: Incremental parsing in language-based editors: user needs and how to meet them. Software: Practice and Experience **31**(15), 1461–1486 (2001). DOI 10.1002/spe.422.
5. Corder, S.P.: The significance of learner's errors. Produktinformation International Review of Applied Linguistics in Language Teaching **5**(4), 161–170 (1967). DOI 10.1515/iral.1967.5.1-4.161.
6. Costa, F., Frasconi, P., Lombardo, V., Soda, G.: Towards Incremental Parsing of Natural Language Using Recursive Neural Networks. Applied Intelligence **19**(1-2), 9–25 (2003). DOI 10.1023/a:1023860521975.
7. Dale, R.: Automating editorial assistance. In: IEEE Colloquium on Hypertext, pp. 3/1–3/3 (1990).
8. Díaz-Negrillo, A., Fernández-Domínguez, J.: Error tagging systems for learner corpora. Revista Española de Lingüística Aplicada **19**, 83–102 (2006)
9. Ehrensberger-Dow, M., Perrin, D.: Capturing translation processes to access metalinguistic awareness. Across Languages and Cultures **10**(2), 275–288 (2009). DOI 10.1556/acr.10.2009.2.6.
10. Ellis, R.: The Study of Second Language Acquisition. Oxford Applied Linguistics. Oxford University Press (1994).

11. Engelbart, D.C.: AUGMENTING HUMAN INTELLECT: A Conceptual Framework. Tech. rep., Stanford Research Institute (1962).
12. Flinn, J.Z.: Case studies of revision aided by keystroke recording and replaying software. Computers and Composition **5**(1), 31–44 (1987). DOI 10.1016/s8755-4615(87)80013-0.
13. Foster, J.: "cba to check the spelling": Investigating Parser Performance on Discussion Forum Posts. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 381–384. Association for Computational Linguistics, Stroudsburg, PA, USA (2010).
14. Foster, J., Vogel, C.: Parsing Ill-Formed Text Using an Error Grammar. Artificial Intelligence Review **21**(3-4), 269–291 (2004). DOI 10.1023/b:aire.0000036259.68818.1e.
15. Foth, K., Menzel, W., Pop, H.F., Schröder, I.: An experiment on incremental analysis using robust parsing techniques. In: Proceedings of the 18th conference on Computational Linguistics (COLING '00), pp. 1026–1030. Association for Computational Linguistics, Stroudsburg, PA, USA (2000). DOI 10.3115/992730.992798.
16. Foth, K., Menzel, W., Schröder, I.: Robust parsing with weighted constraints. Natural Language Engineering **11**(1), 1–25 (2005). DOI 10.1017/s1351324903003267.
17. Gamon, M.: Using Mostly Native Data to Correct Errors in Learners' Writing. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 163–171. Association for Computational Linguistics, Stroudsburg, PA, USA (2010).
18. Gries, S.T., Stefanowitsch, A. (eds.): Corpora in Cognitive Linguistics: Corpus-Based Approaches to Syntax and Lexis. De Gruyter Mouton, Berlin, New York (2007)
19. Gubelmann, R.: Systematischer Überblick über die Fundstücke im Fehlerforum zu "Making word processors process words". Master's thesis, Universität Zürich, Insitut für Computerlinguistik, Zürich, Schweiz (2010)
20. Hausser, R.: Foundations of Computational Linguistics: Human-Computer Communication in Natural Language, 2nd rev. and ext. edn. Springer, Berlin, Heidelberg, New York (2001).
21. Heidorn, G.E.: Intelligent writing assistance: Techniques and applications for the processing of language as text. In: R. Dale, H. Moisl, H. Somers (eds.) Handbook of Natural Language Processing, pp. 181–207. Marcel Dekker, New York, NY, USA (2000)
22. Huang, L., Sagae, K.: Dynamic programming for linear-time incremental parsing. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pp. 1077–1086. Association for Computational Linguistics, Stroudsburg, PA, USA (2010).
23. Jensen, K., Heidorn, G.E., Miller, L.A., Ravin, Y.: Parse fitting and prose fixing: getting a hold on ill-formedness. Computational Linguistics **9**(3-4), 147–160 (1983).
24. Ko, A.J., Aung, H.H., Myers, B.A.: Design requirements for more flexible structured editors from a study of programmers' text editing. In: CHI '05: CHI '05 extended abstracts on Human factors in computing systems, pp. 1557–1560. ACM, New York, NY, USA (2005). DOI 10.1145/1056808.1056965.
25. Kollberg, P.: S-notation – a Computer Based Method for Studying and Representing Text Composition. Master's thesis, Kungliga Tekniska Högskolan Stockholm (1998)
26. Kramer, A.: Rechtschreibkorrektursysteme im Vergleich. DITECT versus Microsoft Word (2004). URL `http://www.mediensprache.net/de/networx/docs/networx-35.aspx`
27. Leacock, C., Chodorow, M., Gamon, M., Tetreault, J.: Automated Grammatical Error Detection for Language Learners, *Synthesis Lectures on Human Language Technologies*, vol. 9. Morgan & Claypool, San Rafael, CA, USA (2010). DOI 10.2200/s00275ed1v01y201006hlt009.
28. Leijten, M., Janssen, D., Van Waes, L.: Error correction strategies of professional speech recognition users: Three profiles. Computers in Human Behavior **26**(5), 964–975 (2010). DOI 10.1016/j.chb.2010.02.010.
29. Leijten, M., Macken, L., Hoste, V., Van Horenbeeck, E., Van Waes, L.: From Character to Word Level: Enabling the Linguistic Analyses of Inputlog Process Data. In: M. Piotrowski, C. Mahlow, R. Dale (eds.) Proceedings of the Second Workshop on Computational Linguistics

and Writing (CL&W 2012): Linguistic and Cognitive Aspects of Document Creation and Document Engineering, pp. 1–8. Association for Computational Linguistics, Stroudsburg, PA, USA (2012).
30. Leijten, M., Van Waes, L.: Inputlog: New Perspectives on the Logging of On-Line Writing. In: K.P.H. Sullivan, E. Lindgren (eds.) Computer Keystroke Logging and Writing: Methods and Applications, *Studies in Writing*, vol. 18, chap. 9, pp. 73–94. Elsevier Science, Amsterdam (2006)
31. Lüdeling, A., Walter, M., Kroymann, E., Adolphs, P.: Multi-level error annotation in learner corpora. In: Proceedings of Corpus Linguistics 2005 (2005)
32. Lunsford, A.A., Lunsford, K.J.: "Mistakes Are a Fact of Life": A National Comparative Study. College Composition and Communication **59**(4) (2008).
33. Macdonald, N.H., Frase, L.T., Gingrich, P.S., Keenan, S.A.: The Writer's Workbench: Computer aids for text analysis. IEEE Transactions on Communication **30**(1), 105–110 (1982).
34. Mahlow, C.: Linguistisch unterstütztes Redigieren: Konzept und exemplarische Umsetzung basierend auf interaktiven computerlinguistischen Ressourcen. Ph.D. thesis, University of Zurich (2011)
35. Meurers, D.: Natural Language Processing and Language Learning. In: C.A. Chapelle (ed.) Encyclopedia of Applied Linguistics, pp. 1–13. Blackwell (2013).
36. Meurers, D., Ziai, R., Amaral, L., Boyd, A., Dimitrov, A., Metcalf, V., Ott, N.: Enhancing Authentic Web Pages for Language Learners. In: Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications, pp. 10–18. Association for Computational Linguistics, Stroudsburg, PA, USA (2010).
37. Napolitano, D.M., Stent, A.: TechWriter: An evolving system for writing assistance for advanced learners of English. CALICO Journal **26**(3), 611–625 (2009).
38. Nivre, J.: Algorithms for deterministic incremental dependency parsing. Computational Linguistics **34**(4), 513–553 (2008). DOI 10.1162/coli.07-056-r1-07-027.
39. Norman, D.A.: Categorization of action slips. Psychological Review **88**, 1–15 (1981)
40. Norman, D.A.: Design rules based on analyses of human error. Communications of the ACM **26**(4), 254–258 (1983). DOI 10.1145/2163.358092.
41. Omar, C., Yoon, Y., LaToza, T.D., Myers, B.A.: Active code completion. In: Proceedings of the 2012 International Conference on Software Engineering, ICSE 2012, pp. 859–869. IEEE Press, Piscataway, NJ, USA (2012).
42. Perrin, D.: Progression Analysis (PA): Investigating writing strategies in the workplace. In: T. Olive, C.M. Levy (eds.) Contemporary Tools and Techniques for Studying Writing, *Studies in Writing*, vol. 10, pp. 105–117. Kluwer, Boston, Dordrecht, London (2002)
43. Perrin, D.: Progression analysis: An ethnographic computer-based multi-method approach for investigate natural writing processes. In: L. Van Waes, M. Leijten, C.M. Neuwirth (eds.) Writing and Digital Media, *Studies in Writing*, vol. 17, pp. 173–179. Elsevier Science, Amsterdam (2006)
44. Perrin, D.: Schreibforschung im Kursalltag: Was die Progressionsanalyse praktisch nützt. In: O. Kruse, K. Berger, M. Ulmi (eds.) Prozessorientierte Schreibdidaktik: Schreibtraining für Schule, Studium und Beruf, pp. 279–294. Haupt, Bern, Stuttgart, Wien (2006)
45. Perrin, D.: Language policy, tacit knowledge, and institutional learning: The case of the Swiss public service broadcaster SRG SSR. Current Issues in Language Planning **12**(3), 331–348 (2011). DOI 10.1080/14664208.2011.604953.
46. Perrin, D., Wildi, M.: Statistical Modeling of Writing Processes. In: C. Bazerman, R. Krut, K. Lunsford, S. McLeod, S. Null, P. Rogers, A. Stansell (eds.) Traditions of Writing Research, pp. 378–393. Routledge, New York, NY, USA (2009)
47. Reznicek, M., Lüdeling, A., Hirschmann, H.: Competing target hypotheses in the Falko Corpus. a flexible multi-layer corpus architecture. In: A. Díaz-Negrillo (ed.) Automatic Treatment and Analysis of Learner Corpus Data. John Benjamins, Amsterdam (forthcoming)
48. Rimrott, A., Heift, T.: Evaluating automatic detection of misspellings in German. Language Learning & Technology **12**(3), 73–92 (2008).
49. Roark, B.: Probabilistic top-down parsing and language modeling. Comput. Linguist. **27**(2), 249–276 (2001). DOI 10.1162/089120101750300526.

50. Schulze, M.: Ein sprachunabhängiger Ansatz zur Entwicklung deklarativer, robuster LA-Grammatiken mit einer exemplarischen Anwendung auf das Deutsche und das Englische. Ph.D. thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (2004).
51. Severinson Eklundh, K.: Problems in Achieving a Global Perspective of the Text in Computer-based Writing. In: M. Sharples (ed.) Computers and Writing: Issues and Implementation, pp. 73–84. Kluwer, Boston, Dordrecht, London (1992)
52. Severinson Eklundh, K.: Linear and nonlinear strategies in computer-based writing. Computers and Composition **11**(3), 203–216 (1994). DOI 10.1016/8755-4615(94)90013-2.
53. Sloan, G.: Frequency of Errors in Essays by College Freshmen and by Professional Writers. College Composition and Communication **41**(3) (1990).
54. Smithson, I.: The Writing Workshop. Computers and Composition **4**(1), 78–94 (1986). DOI 10.1016/s8755-4615(86)80009-3.
55. Sullivan, K.P.H., Lindgren, E. (eds.): Computer Key-Stroke Logging and Writing: Methods and Applications, *Studies in Writing*, vol. 18. Elsevier Science, Amsterdam (2006).
56. Van De Vanter, M.L.: Practical language-based editing for software engineers. In: Software Engineering and Human-Computer Interaction, *Lecture Notes in Computer Science*, vol. 896, pp. 251–267. Springer, Berlin, Heidelberg, New York (1995). DOI 10.1007/bfb0035821.
57. Van Waes, L., Leijten, M., Neuwirth, C.M. (eds.): Writing and Digital Media, *Studies in Writing*, vol. 17. Elsevier Science, Amsterdam (2006).
58. Van Waes, L., Leijten, M., Van Weijen, D.: Keystroke logging in writing research: Observing writing processes with Inputlog. GFL – German as a foreign language (2–3), 41–64 (2009)
59. Vernon, A.: Computerized grammar checkers 2000: Capabilities, limitations, and pedagogical possibilities. Computers and Composition **17**(3), 329–349 (2000). DOI 10.1016/s8755-4615(00)00038-4.
60. Williams, N.: Computer assisted writing software: RUSKIN. In: N. Williams, P.O. Holt (eds.) Computers and Writing: Models and tools, chap. 1, pp. 1–16. Intellect, Oxford (1989)
61. Wirén, M.: Interactive incremental chart parsing. In: Proceedings of the fourth conference of the European chapter of the Association for Computational Linguistics, pp. 241–248. Association for Computational Linguistics, Morristown, NJ, USA (1989). DOI 10.3115/976815.976848.