Reading and Writing
https://doi.org/10.1007/s11145-021-10234-6

Extraction of transforming sequences and sentence histories from writing process data: a first step towards linguistic modeling of writing

Cerstin Mahlow[1] · **Malgorzata Anna Ulasik**[1] · **Don Tuggener**[2]

Accepted: 26 November 2021

Abstract
Producing written texts is a non-linear process: in contrast to speech, writers are free to change already written text at any place at any point in time. Linguistic considerations are likely to play an important role, but so far, no linguistic models of the writing process exist. We present an approach for the analysis of writing processes with a focus on linguistic structures based on the novel concepts of transforming sequences, text history, and sentence history. The processing of raw keystroke logging data and the application of natural language processing tools allows for the extraction and filtering of product and process data to be stored in a hierarchical data structure. This structure is used to re-create and visualize the genesis and history for a text and its individual sentences. Focusing on sentences as primary building blocks of written language and full texts, we aim to complement established writing process analyses and, ultimately, to interpret writing timecourse data with respect to linguistic structures. To enable researchers to explore this view, we provide a fully functional implementation of our approach as an open-source software tool and visualizations of the results. We report on a small scale exploratory study in German where we used our tool. The results indicate both the feasibility of the approach and that writers actually revise on a linguistic level. The latter confirms the need for modeling written text production from the perspective of linguistic structures beyond the word level.

Keywords Writing process · Keystroke-logging · Transforming sequence · Text history · Sentence history · Written text production · Linguistic modeling

✉ Cerstin Mahlow
cerstin.mahlow@zhaw.ch

[1] School of Linguistics, Zurich University of Applied Sciences, Theaterstrasse 17, 8401 Winterthur, Switzerland

[2] School of Engineering, Zurich University of Applied Sciences, Winterthur, Switzerland

Published online: 01 January 2022

Introduction

Producing written text is a non-linear process: during production, writers are free to modify the text at any place and at any point in time, without leaving any traces in the final product—in contrast to speech. Recording these processes with keystroke-logging tools (for an overview of current technology and its evolution, see Lindgren et al., 2019a) allows researchers how writers produce texts. Writing process data is collected and stored in chronological order as the writer adds and removes characters. This data thus allows to follow, replay, and investigate the non-linear evolvement of the text.

Linguistic aspects of writing are, somewhat surprisingly, an under-researched topic: relevant disciplines mainly analyze the *product* (the final text) on various levels, but not the *process*. Even though linguistic considerations play an important role in the text production process—Cookson mentions that "Writing is a linguistically based process" (Cookson, 1989, p. 20)—and metalinguistic awareness is one competence of professional writers (Horning, 2006, p. 119), there are currently no suitable approaches that would allow us to process large amounts of production data and test the explanatory power of linguistic theories. Approaches within structural linguistics, functional linguistics, and cognitive linguistics have been developed to cover products, i.e., texts. Whether the models and implementations based on those theories can be used for analyzing and modeling writing processes has not been systematically studied, yet.

To the best of our knowledge, no in-depth linguistic modeling of the writing process has been attempted so far, although the need to do so has been stated repeatedly (Alamargot & Chanquoy, 2001; Cookson, 1989; Weingarten et al., 2004). Current models of writing mainly aim at explaining underlying cognitive processes (Galbraith 1999; Kellogg, 1996, 2001) and consider language only as a tool to make ideas visible through "translation" or "transcription" (Bereiter & Scardamalia, 1987; Hayes, 2012; Hayes & Chenoweth, 2006). Only some models include general "linguistic knowledge" as part of the working memory (Hayes, 1996) or when explaining how the "translator" works: it "translates the ideas into a linguistic string, selecting lexical items, ordering them, and supplying inflections consistent with the information" (Chenoweth & Hayes, 2003). In the most recent model by Hayes (2012), and in the additions proposed by Leijten et al. (2014), linguistic aspects are also hidden in the translator.

Natural language generation (NLG) is concerned with the generation of texts that could have been produced by a human (for an overview of working systems see Dale, 2020), but the writing process is not of concern (Mahlow & Dale, 2014). Writing models, however, do aim to model the writing process, and their implementation as computer programs would therefore be considered simulations of the human writing process.

To finally model the writing process we need "[...] a sufficient level of formalisation in order that a computer simulation for example could be possible" (Alamargot & Chanquoy, 2001, p. 3). Hayes (2012) reports on what still seems to be the only attempt so far to actually create a model that could be run as computer

program to simulate writing. He states: "Running programs force us to be very specific about how writing processes work and about the structure of the memory resources that the writing processes rely on." (Hayes, 2012, p. 386) We therefore need new writing models or significant additions to existing ones.

The overall goal of our research is to better understand the production of linguistic units in general by modeling it. In line with Stachowiak (1973), we consider models as abstract representations for specific purposes. Gladkij and Mel'čuk (1973) defined modeling as formalization in the sense of consistent, unambiguous, and explicit description of linguistic data (Gladkij & Mel'čuk, 1973, footnote 2). Models are thus considered an implementation of a theory: "The connection between a model and a theory is that a model *satisfies* a theory; that is, a model obeys those laws of behavior that a corresponding theory explicitly states or which may be derived from it." (p. 143 Weizenbaum, 1976, emphasis in the original). Models are also a prerequisite to design and implement running computer programs to test theories (Weizenbaum, 1976, p. 145) and to construct automated tools to support the task at hand (Cookson, 1989, p. 20).

To truly be able to create simulations, we need to model not only cognitive processes and lexical choices, but also the evolvement of linguistic structures on the syntactic and morphosyntactic level. We call these aspects *linguistic modeling of the writing process*. Hayes successfully models children's writing by extending the models of Bereiter and Scardamalia (1987) and considers modeling adult writing "a task that seemed once and, perhaps, seems still, too difficult to accomplish [...]" (Hayes 2012, p. 386). With this work, we aim to contribute to tackle this task.

Modeling the writing process with a focus on linguistic structures (specifically sentences) and their transformation requires explicitly linking process and product. This linking will let us explore the process for observable phenomena both (a) in the final product or in various stages of the *text-produced-so-far* (TPSF) at any given point during writing, and (b) when inspecting the production data of a complete writing session or at intermediate stages.

We propose the concept of linguistically motivated *transforming sequences* to cover text evolvement. We provide a first implementation in the form of an open-source stand-alone analysis tool called *Text History Extraction tool* (THEtool)[1] intended as a proof of concept and as a starting point for further development and exploration.

Our approach is also aimed at practical applications that rely on live processing for visualization or specific support for writers based on morphosyntactic and syntactic information *during* writing. Linguistic modeling of written text production during writing is required for three main types of language-aware (Piotrowski & Mahlow, 2009) or language-sensitive (Dale & Douglas, 1996) applications:

(a) Language-aware editing support for writers based on morphosyntactic information. Piotrowski and Mahlow (2009) propose information functions, move-

[1] Available at github: https://github.com/mulasik/wta.

ment functions, and operations as types of editing functions, they give examples (Mahlow and Piotrowski 2009a) and point out challenges and limitations (Mahlow & Piotrowski, 2009b). Their goal is to help writers by offering functions operating on linguistic units in a way writers would refer to them when talking about their texts (Mahlow & Piotrowski, 2008).

(b) Feedback during writing based on linguistic insights, like flagging inconsistent tenses or highlighting discourse elements to make argumentative chains visible.

(c) Visualization of the syntactic unfolding of sentences to complement the established representation of changes within words, e.g., replacement of one preposition by another, replacement of a complex noun phrase by a pronoun, deletion of phrases. The ability to trace edits—a writer may have tried several variants before settling on one—would contribute to the detection of the origins of side effects of editing and provide insights for the design of support for error prevention or correction.

In the next section, we look at related work both in writing research aiming at linguistic insights and with respect to methods and techniques from natural language processing (NLP) to be used and adapted for our purposes. In "Method" section we report on the method we developed, the theoretical reasoning which led to the concepts of *transforming sequences*, *text history*, and *sentence history*. The implementation as proof of concept is described and illustrated with an example in "Implementation" section. In "Findings" section we share first insights we gained when applying our method on a small set of keystroke logs in German and report on challenges and limitations with the implementation we provide.

Related work

In the following, we first summarize literature relevant to the role of linguistics at large in writing research. We then report on current work towards the application of natural language processing (NLP) techniques to writing process data. Finally, we look at NLP for both their recent interests in the production of written text and for the current state of incremental parsing, which we consider relevant for applying NLP to process data.

Linguistics in writing research

Linguistics does play a role in writing research, but usually targets units above or below the sentence level. Weingarten et al. (2004) suggest to investigate "whether written language and its historical and individual developments have to be treated in the same theoretical framework [...] as other types of linguistic systems," but then concentrate on words only. They accordingly propose a "model of written word production," which later is used by Ballier et al. (2019) for keystroke analysis, but again on the word level only. A related line of work focuses on cognitive aspects involved in planning and producing linguistic units: Martin et al. (2010) as well as Nottbusch

et al. (2007) look at sentences but explore planning and pausing at specific boundaries in dedicated experiments rather than in natural situations. Similarly, Nottbusch (2010), Torrance and Nottbusch (2012), and Torrance (2015) investigate planning aspects to explore sentence production, but only consider words and (short) sentences as linguistic units. Baaijen et al. (2012) and Galbraith and Baaijen (2019) work on modeling bursts (Kaufer et al., 1986), taking into account not only time aspects, but also type of bursts as either production or revision bursts. They do not report on any implementation and focus on pauses as indicators.

Progression analysis, as developed by Perrin (2002, 2006) and used in various studies (e.g., Ehrensberger-Dow & Perrin, 2009; Perrin 2019; Perrin & Wildi, 2009), is used for media-linguistic research emphasizing ethnographic aspects (Perrin, 2013). This approach focuses on *language use* rather than *modeling on linguistic structures*.

In studies of text revision, linguistics is taken into account as "language knowledge" or "metalinguistic knowledge" (Horning, 2006), or when categorizing revision, for which various taxonomies have been developed: Bridwell (1980) and Sommers (1980) focus on observable revisions at the surface of the text and distinguish syntactic structures involved. Faigley and Witte (1981) and later Fitzgerald (1987) aim at categorizing changes in meaning of the text on the basis of observable structural changes. Allal and Chanquoy (2004) and Lindgren (2005) introduce the notions of pretextual and precontextual revisions, i.e., mental changes before the author transcribes them—but as such revisions are not directly observable through the process and product data we do not consider them here. The distinction between *meaning-changing* and *meaning-preserving* revisions by Faigley and Witte (1981) has also been used in recent work on revision taxonomies for Wikipedia edit histories (e.g., Daxenberger & Gurevych, 2012). Although Wikipedia allows for the extraction of very detailed version histories of articles (Daxenberger & Gurevych, 2013), these taxonomies rely on product data only, as no information on edits between saves is available.

Only very recently revision taxonomies considering process information have been developed (e.g., Conijn et al., 2021). Bowen (2019) and Bowen and Van Waes (2020) report on an investigation to combine process and product data. They carefully examine both types of data as produced by two writers over various writing sessions in a natural setting. The method described is a combination of automatic analysis as established in writing research—pauses, S-notation, revision matrix, etc.—and manual coding of the enriched process data, taking into account the final product and using the replay functionality of the keystroke-logging and analysis tool Inputlog.[2] Their analyses show the explanatory power of Systemic Functional Linguistics (Halliday, 1976), but their process for extraction and annotation relies on manual work and is therefore not suited for automatic processing of large amounts of process data or analysis during writing as would be needed for implementing supportive functionality or immediate feedback. However, the explicit linking of process

[2] https://www.inputlog.net.

and product and, therefore, combined analysis of both process data and product data is a general approach we deem essential to further advance writing research.

Application of NLP techniques to writing process data

Leijten et al. (2012, 2015) were the first to apply existing NLP tools to S-notation (Kollberg & Severinson Eklundh, 2002) data, enriching them automatically with basic linguistic information like part-of-speech (POS) in order to facilitate higher-level analyses of the data and gain insights into aspects like fluency (Leijten et al., 2019). They work on Dutch and English and use Inputlog for logging and analysis. The starting point is the final product, which is then linked to the corresponding process data: a word can undergo various revisions, which can be inspected or analyzed statistically.

Cislaru and Olive (2018) use a linguistic approach to French process data, focusing on *bursts* from a linguistic perspective. They process P-bursts (Kaufer et al., 1986; Baaijen et al., 2012) and analyze one burst at a time to explore the status of routines in writing at the discourse and psycholinguistic levels. They consider sequences of characters produced without interruption; the boundaries of units are defined by pauses in the logging data, i.e., their approach relies on timestamps. They had writers use either Inputlog or Scriptlog (Johansson et al., 2018) for writing and used analyses provided by those tools as the starting point of their investigations (Cislaru & Olive, 2018, p. 19). The focus is on process data which allows for explanations of phenomena visible in the product. However, they rely on manual categorization and annotation, as they found automatic processing of French data not to be feasible (Cislaru & Olive, 2018, p. 89). Using their approach for automatic analysis and visualization during writing or on a larger scale is thus hardly possible.

So far, none of the approaches is mature enough for automatic processing of large amounts of keystroke-logging data from writing in natural settings to test the explanatory power of linguistic theories and to develop linguistic models of the writing process. However, a combination and further extension of the general principles seems very promising. Using NLP techniques in general, as some research groups do, offers the possibility to work and test hypotheses on large amounts of data.

Only recently, the NLP community has become interested in writing process data. Plank (2016) explores the relation of linguistic theories and writing, and studies how information on the process supports the analysis of the product. They show that typing patterns can be seen as a proxy for shallow syntactic parsing of the final product on the word level and can be used to improve performance of such parsers. Schneier (2020) examines the production of discourse elements in social media contexts to understand from a linguistic point of view how individuals compose texts to achieve their communicative goals. They explore keystroke-logging data by focusing on the production of paralinguistic variables in text messaging. Goodkind (2021) aims to visualize linguistic patterns and their specific production time as can be observed during writing and captured as keystroke-logging data, but aims at characters and words only. All approaches make use of the temporal aspect of writing: writers tend to pause longer before larger units, i.e., pauses between words are longer as within

words, pauses between clauses are longer as between words, etc. as previously shown by Nottbusch (2010) and Torrance and Nottbusch (2012).

Current state of incremental parsing

Applying NLP to writing poses two main challenges: (a) the text is growing—syntactic parsing should be *incremental* but in a non-linear way to add newly written parts to the parse tree and update revised parts; (b) the text is unfinished—processing has to be *robust*, so that it can handle ill-formedness, incompleteness, and inconsistency (the "I3", as Van De Vanter, 1995 calls them), problems Cislaru and Olive (2018) faced for French and therefore opted for manual annotation.

Incremental parsers analyze their input word by word and start to construct a parse tree immediately. They are typically (though not exclusively) used in the context of acoustic speech recognition; approaches for incremental parsing therefore usually assume a linear evolvement of text. 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 without leaving any trace—temporal and spatial dimensions are not identical. This means that written text does not evolve linearly, and re-parsing is needed if earlier text is changed. In order to avoid costly re-parsing of unchanged text, it would be preferable to instead 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., Cook & Welsh, 2001). It is rarely addressed in NLP, though; one such example is Wirén (1989). In general, incremental parsers for natural languages (e.g., Costa et al., 2003; Huang & Sagae, 2010; Nivre, 2008; Roark 2001) handle linearly unfolding text.

Research on *robust parsing* is investigated separately. It aims at parsers that are able to handle ungrammatical input gracefully and is done in the context of applications such as processing learner language or user-generated content (e.g., Farzindar & Inkpen, 2020; Leacock et al., 2010), and, of course, grammar checking (e.g., Clément et al., 2011; Heidorn, 2000; Jensen et al., 1983).

Related work: conclusion

What we can conclude from work in related fields is the following: There is a growing awareness and interest in writing processes and respective data. Current developments in the field of NLP might be integrated into tools for the analysis of writing process data. The parsing needed can be seen as a novel, challenging NLP task which we leave for future work to explore in detail. In our experiments presented hereafter, we apply an existing syntactic parser in an exploratory fashion to gain insights into its benefits and limitations in our setting. In conclusion, the explicit automatic linking of process and product during various stages of the writing process or a specific writing session has not been explored systematically but is the most promising direction we will follow.

Method

In this section we outline the design principles and main concepts for our approach considering both process and product of writing with a focus on linguistic structures. We point to general aspects used to operationalize those. As proof of concept we will later show our implementation as *Text History Extraction tool* (THEtool) in "Implementation" section.

A major goal of our efforts is to complement established analyses of text production—i.e., on character level and time aspects—with a perspective on linguistic structures, in particular syntax. Sentences are the most important building blocks for creating written texts. They are at the interface of two levels of analysis: internally, the structure of sentences is governed by relatively strict, codified rules. The order of sentences in a text, however, is governed by completely different considerations (see work on sentence ordering (Lapata, 2003; Rehm et al., 2020)).

A challenge in this regard is to determine which points and spans in the keystroke-based process data are relevant for investigation and processing. We consider raw logging data the starting point for any application. The XML-based idfx format produced by Inputlog (Leijten et al., 2012) is the de-facto standard for storing and exchanging keystroke-logging data. As proof of concept we design a tool as stand-alone application. This way, our method and subsequently our tools will work independently from a specific logging method or application, and allow for seamless integration with a variety of other applications and further processing.

Furthermore, we aim to incorporate NLP tools to supplement the analysis with relevant morphosyntactic and syntactic annotations. These annotations should be produced automatically to feasibly scale them up to data volumes that are suitable for statistical analyses. The application of NLP tools requires text, i.e., (intermediate) product data, not process data as stored in the idfx files. Our approach therefore relies on two main concepts:

- *Version* a point in the production history of a text that is deemed relevant based on particular criteria, a version is thus a specific text-produced-so-far (TPSF);
- *Transforming sequence* the textual material (i.e., product data) combined with the edit operations (i.e., process data) that comprises the difference between two adjacent versions.

This constitutes a constant linking of process and product—a version is understood as a preliminary product, differences in versions are visible on the surface and can be explained by using the editing operations involved. We introduce a novel way of visualizing these concepts as histories on different levels.

In the next sections we explain the components of our approach on a conceptual level.

Version

The *version* is one of the fundamental concepts of our approach. A version is a specific point in the history of a text product depending on criteria routed in the writing process. In contrast, work on Wikipedia versions (e.g., Daxenberger & Gurevych, 2013) only considers actively saved versions—i.e., product data only—, but does not include process data.

Technically, we could create a version after each keystroke, which would result in a large amount of versions to be parsed, but two adjacent versions would differ in one character only. We apply the definition as proposed by Mahlow (2015a): A version is the current text-produced-so-far (TPSF) when a *change in production mode* occurs which also includes spatial aspects. A change in production mode is defined as switching between one of the modes (a) continuous writing at the leading edge of the TPSF (i.e., append) ignoring white insertions as proposed by Lindgren et al. (2019b), (b) continuous deletion of something, (c) continuous insertion of something into existing text (Mahlow, 2015a). Two adjacent versions thus differ by bursts. Insertion of characters at two different points in the TPSF will result in two versions, the same holds for deletion.

A further option for defining a version is the use of temporal aspects like Cislaru and Olive (2018) do, assuming that longer pauses indicate cognitive processes, and writers might consider the TPSF at that point to constitute a version or draft. A version as defined by temporal aspects only will not differ from the previous one on the surface. One could also combine both temporal and procedural aspects to define criteria for versions.

We mark versions with an index indicating the sequence and order of versions. Thus V_i is the version with index i where V_{i-1} is the previous and V_{i+1} is the following version.

Transforming sequence

When considering versions as defined by change in production mode, two versions V_i and V_{i+1} differ on the surface. This difference is due to writing and revising (append, i.e., extending the text at the edge of the TPSF, deletion, or insertion). We call the operation together with the resulting visible difference of V_i and V_{i+1} a *transforming sequence*. This concept is the second fundamental concept and a genuine novel contribution.

On the sentence level, a single transforming sequence can result in complex edit operations, such as (a) sentence modification, (b) sentence deletion, (c) sentence insertion, (d) sentence split, and (e) sentences merge, and any combination thereof. We use sentences as large structural unit in texts that can be recognized automatically quite reliably with state-of-the-art NLP tools, i.e., applying sentence segmentation. Note that one could define edit operations using smaller structual units (e.g., phrases, clauses) accordingly.

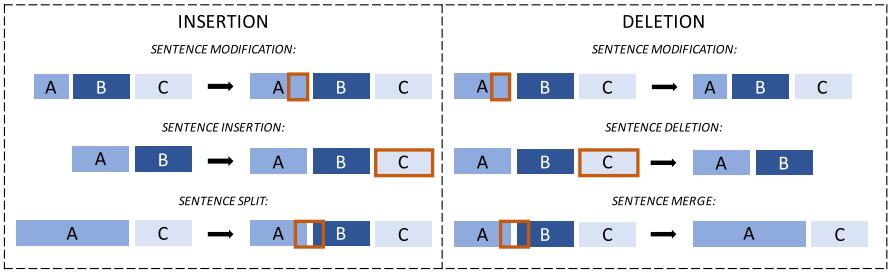


Fig. 1 Sentence-level edit operations resulting from one transforming sequence. The transforming sequence is marked as an orange box. The left column illustrates the operations resulting from a transforming sequence of type insertion and the right one lists possible operations resulting from a transforming sequence of type deletion. Depending on the position and content of the transforming sequence, the impact of the transforming sequence on the sentences differs. (Color figure online)

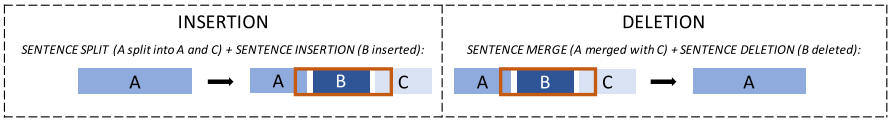


Fig. 2 Sentence-level edit operations resulting from one transforming sequence containing more than one sentence

Figure 1 provides an overview of sentence-level edit operations resulting from transforming sequences with different contents. We consider three sentences involved, A, B, and C. In these examples, the transforming sequence is either a character sub-sequence within the sentence (first row), represents a complete sentence (second row), or contains segments involving parts of two consecutive sentences (last row).

If the transforming sequence affects multiple sentences or sentence segments, it results in multiple successive sentence-level operations, as shown in Fig. 2. The first example shows a modification of sentence A that is split into two sentences, sentence B is inserted in between. The example on the right visualizes a merge operation: sentence B, originally placed between sentences A and C is deleted, and A and C then merged into one resulting sentence now labeled A. Longer operation chains are also possible. If the edit consists in removing a complete paragraph, the number of sentence-level deletion operations corresponds to the number of sentences which the paragraph contained.

The distinction between various sentence-level edit operations and their dependency on the transforming sequence constitutes the foundation for creating text and sentence histories we introduce as novel visualization and exploration format.

Text history and sentence history

We introduce *text history* and *sentence histories* as concepts for analysis and visualization of the text production processes using transforming sequences. *Text history*

comprises the collection of versions that yield the final text product. *Sentence histories* explain the evolvement of individual sentences across the respective versions of a text. Note that one could define histories of other units (e.g., clauses) in a similar way.

In the *text history*, we look at the text as a whole and track the changes leading from version V_i to V_{i+n} on the character level. Sentences are marked explicitly as linguistic units using general sentence splitting techniques, and are subsequently extended with dependency information and POS tags.[3] The text history comprises all versions captured throughout the tracking of the text production process; it captures the text production process at large and is intended to gain insights into general writing habits of the writer. Versions might be filtered for relevance according to specific criteria.

A *sentence history* is created for each of the sentences—this also includes sentences that are not part of the final product due to revision. A single sentence history contains all versions of a particular sentence in chronological order. Note that the versions will not carry successive indices as a writer might have come back to this sentence several times while writing or modifying other sentences in between, so this sequence might be a list like $< V_1, V_2, V_5, V_{89}, V_{90} >$.

The foundation for creating the sentence history are sentence categories as discussed "Transforming sequence" section. Each TPSF contains a list of new, changed, deleted, and unchanged sentences. Iterating over the TPSFs chronologically and analyzing their sentence lists, allows to automatically track the evolution of each sentence and also spot sentence deletion. Additionally, a global list of all sentences is maintained, which is used to identify sentences which have been deleted and then inserted again in an unchanged form (i.e., cut and paste).

The sentence history is intended to investigate edit operations performed on the sentence level, which provides insights of variants a writer might have considered during text production.

Relevance

The text history and the sentence histories derived from a writing session contain all text and sentence versions captured. However, not all of them may be of interest, depending on the context and the research purpose: We introduce the concept of *relevance* (on the TPSF and sentence level) to cover these aspects. The definition and subsequent operationalization of relevance is the basis for filtering the text and sentence history. Relevance of versions with respect to the research question is thus an essential aspect of THEtool that has to be configured carefully. We give an example in "Tool setup" section.

[3] Note that both pieces of information comes from NLP tools which can be switched off or replaced, e.g., to use constituent structure information.

An edit operation is an act of either removing or inserting a sequence without interruption. As long as the writer keeps deleting or inserting subsequent characters one after another without changing direction or moving the cursor to a different position (e.g. with arrows or a mouse click), the whole operation is treated as a single edit operation and the removed and inserted characters are aggregated and stored in a data structure called transforming sequence. As soon as the operation gets interrupted, the current text version is saved and the sequence is set back to an empty string. This way, it is ensured that there is only one transforming sequence per edit and per TPSF and that this transforming sequence belongs to one of the two edit categories: deletion and insertion. Depending on the length of the uninterrupted edit operation, the transforming sequence can contain just a single character or even multiple sentences.

Fig. 3 Text produced during a short writing session which we use to illustrate the working of our tool

An·edit·operation·is·an·act·of·either·removing·or·inserting·a·sequence{·{at·the·beginning,·in[t·]^{16}|$_{17}${·the·middle·or·at·the·end·of·the·text·}^{17}}^{15}|$_{16}$without·interruption}^{1}.|$_{1}$·A[S·lo]^{2}|$_{2}$s·long·as·th[w·rti]^{3}|$_{3}$e·writer·keeps·deleting·or·inserting·subsequent·characters·one·after·another·without·changing·di[te]^{4}|$_{4}$re[stio]^{5}|$_{5}$ction·or·moving·the·cursor·to·a·different·position[.]^{7}|$_{8}${·(e.g.·with·arrows·or·a·mouse·click)}^{27}|$_{28}${,}^{8}|$_{9}$·the·whole·operation·is[t]^{6}|$_{6}$·treated·as·a·single·edit·operation{·and·the·removed·and·inserted·characters·are·aggregated·and·stored·[as·one·sequence{in·a·data·structure}^{29}|$_{30}${,·a·so]^{28}|$_{29}$·called·transforming·sequence}^{10}.·As·soon·as·the·operation·gets·interrupted,·the·current·text·version·is·saved{·and·the·sequence·is·set·back·to·an·empty·string}^{11}}^{9}|$_{10}$.|$_{7}$·|$_{11}$·This·wa{,}^{30}|$_{31}$y·it·is·ensured·that·there·i[s]^{12}o|$_{12}$·only·one·tr[a]^{13}s|$_{13}$ns[f]^{14}r|$_{14}$orming·sequence·per·edi{·and·per·TPS[F{F·and·that·t}^{32}|$_{33}$}^{21}t.|$_{15}$T]^{31}|$_{32}$his·transf[r]^{18}|$_{18}$orming·sequence·belongs·to·one·of·th[·]^{19}|$_{19}$e·two·edit·[v]^{20}|$_{20}$categories:·deletion·and·insertion.|$_{21}$·Depending·on·the·length·of·the·{uninterrupte[r]^{37}|$_{38}${d}^{38}.}^{26}|$_{27}$edit·operation[·]^{22}|$_{22}$,·the·transforming·sequence·[dn·]^{23}|$_{23}$can·contain·[from]^{33}|$_{34}${just}^{34}|$_{35}$·a·single·character·[up·to]^{35}|$_{36}${or·even}^{36}|$_{37}$·[several]^{24}|$_{25}${multiple}^{25}|$_{26}$·sentences.|$_{24}$·

Fig. 4 S-notation of the process data used for illustration

Implementation

In this section we first report on our implementation in general and the design considerations to operationalize the principles laid out in the previous section. No manual intervention during processing keystroke-logging data is needed, all steps are done automatically to ultimately enable the analysis of large amounts of writing process data. We will illustrate how THEtool works by using a small example, the logging data recorded when writing one of the paragraphs of this paper,[4] see Fig. 3.

Figure 4 provides the visualization for this text in the established S-notation for comparison. It marks breaks (using the pipe symbol), deletions (in square brackets), and insertions (in curly brackets) on the character level. The sequence of actions is indicated by increasing index numbers: the first interruption of continuous writing is marked as the break with index 1 after having written "*An edit operation is an act of either removing or inserting a sequence.*". Then the action with index 1 is executed:

[4] Note that this paragraph does not appear here in the form of the final version, as editing and revising continued after we stopped the recording to use it for explanation.

The writer inserts something ("*without interruption*") before the sentence full stop they just typed. Some actions later, at breakpoint 15 (somewhat in the middle of the paragraph), the writer comes back to this first sentence and adds "*at the beginning int*", stopps again to delete the last "*t*" (action 16) and then inserts "*the middle or at the end of the text*". This sentence then will not be edited further. This example shows that the writer goes back and forth and revises on various levels: words and phrases are added and deleted, typing or spelling errors are corrected. Although the example is rather small and only contains 37 editing/revising actions (the highest index used), it is hard to read and to follow.

General features of THEtool

THEtool is a stand-alone open-source application implemented in Python[5] for automatically parsing raw keystroke logging data as it is collected and stored when users write. It processes the data and generates text and sentence histories without manual intervention.

The input file processed by THEtool is an idfx file in XML format containing each keystroke performed by a writer as a separate log entry. The XML structure is flat and allows for fast processing. Keystroke logging tools either use idfx as default storage format (e.g., Inputlog) or can export their internal format into idfx for exchange purposes (e.g., Scriptlog which we used here). Listing 1 shows an extract of a writing session: A writer produces the English pronoun *it*.

[5] https://www.python.org/.

Listing 1 Example of raw logging data in XML format, showing the sequence needed to produce the pronoun *it* in our example, i.e., pressing and releasing "i", and pressing and releasing "t".

```
<event id="388" type="keyboard">
    <part type="wordlog">
        <position>567</position>
        <documentLength>569</documentLength>
        <replay>True</replay>
    </part>
    <part type="winlog">
        <startTime>233465118</startTime>
        <endTime>233465123</endTime>
        <key>VK_I</key>
        <value>i</value>
        <keyboardstate></keyboardstate>
    </part>
</event>
<event id="389" type="keyboard">
    <part type="wordlog">
        <position>568</position>
        <documentLength>570</documentLength>
        <replay>True</replay>
    </part>
    <part type="winlog">
        <startTime>233465771</startTime>
        <endTime>233465776</endTime>
        <key>VK_T</key>
        <value>t</value>
        <keyboardstate></keyboardstate>
    </part>
</event>
```

THEtool creates a set of output files storing the processing results. The main outputs are unfiltered and filtered text and sentence histories with metadata stored as JSON data structures, a commonly used lightweight data exchange format, as well as visualizations of the text and sentence histories in SVG format.

The application code and a detailed documentation on how to configure and run THEtool is available on GitHub.[6]

Design considerations: TPSF data structure and TPSF data points

The central building block of THE TOOL is the *TPSF data structure* (*TDS*). It is used to store the current text version together with further details.

Since the understanding of *version* may differ according to research questions and use cases, we currently use two main modes to capture versions from idfx files: the *Pause Capturing Mode* (PCM), which relies on a preset pause duration to yield versions, and the *Edit Capturing Mode* (ECM), which uses a change in production mode to determine versions (see "Version" section). It is possible to combine both

[6] https://github.com/mulasik/wta.

modes—i.e., define criteria for versions as combination of change in production mode and pause duration—and add other modes as needed.

As soon as a new version is detected, it is stored in a TDS. In PCM, a TDS consists of the captured text version and the preceding pause duration. In ECM, a TDS additionally contains the previous text version, version number, the transforming sequence (see "Transforming sequence" section), and the list of sentences which make up this version. It also includes the relevance label and result of the relevance evaluation performed according to the filtering parameters configured by the user (see "Relevance" section).

A text version stored in a TDS is called a *TPSF data point* (*TDP*). Listing 2 shows an example of a TDP in JSON format. Some attributes have been omitted in the example for better overview. The full example is provided in THEtool's GitHub documentation.

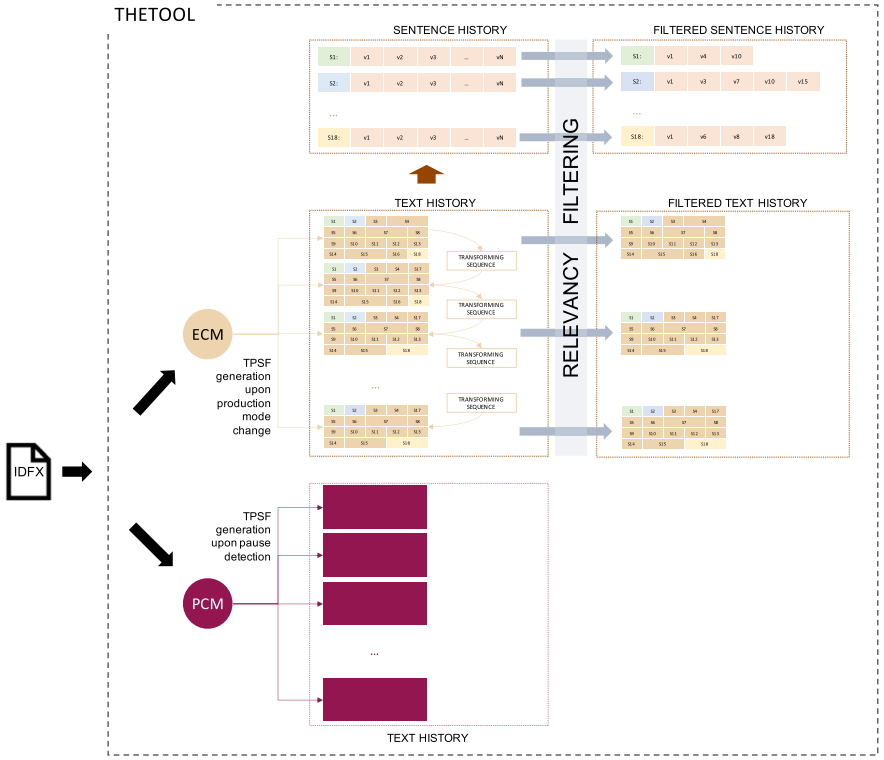


Fig. 5 Compact view on the components and procedures when processing an example writing session captured in an idfx file

Listing 2 Example of TDP relevant to the first sentence from the example text presented in figure 8.

```
{
 "revision_id": 4,
 "previous_text_version": "An edit operation is an act of either
     removing or inserting a sequence. ",
 "preceding_pause": 0.54,
 "result_text": "An edit operation is an act of either removing or
     inserting a sequence without interruption. ",
 "edit": {
    "edit_start_position": 70,
    "transforming_sequence": {
     "label": "insertion",
     "text": " without interruption",
     "tags": [...]
 },
 "sentences": {
    "previous_sentence_list": [
        {
            "text": "An edit operation is an act of either removing
                or inserting a sequence. "
        }
    ],
    "current_sentence_list": [
        {
            "text": "An edit operation is an act of either removing
                or inserting a sequence without interruption. "
        },
    ]
    "new_sentences": [],
    "edited_sentences": [
        {
            "previous_sentence": {
                "text": "An edit operation is an act of either
                    removing or inserting a sequence. "
            },
            "current_sentence": {
                "text": "An edit operation is an act of either
                    removing or inserting a sequence without
                    interruption. "
            }
        }
    ],
    "deleted_sentences": [],
    "unchanged_sentences": []
 },
 "morphosyntactic_relevance_evaluation": [...],
 "morphosyntactic_relevance": true
}
```

In order to make the tool applicable for various purposes the key features are configurable, e.g., the minimal pause duration for triggering the generation of TDS in PCM or relevance parameters for filtering the TPSFs. As the NLP tools applied for splitting sentences, tokenization, or spell checking are language dependent, the user has to set the input text language. So far, we successfully applied THEtool to English, German, and Greek input texts.

All TPDs form the text history which constitutes the basis for other outputs, such as sentence histories. Based on the TPD relevancy label, the text history can be filtered, i.e., TPDs labelled as non-relevant will be excluded for the following steps. Figure 5 illustrates the main processing steps, the following sections provide more details.

Keystroke-logging data processing

THEtool iterates over the idfx file collecting keystroke data such as cursor position, key name, keystroke value, start and end time until an event occurs that triggers the TDP creation for a version V_i.

TDP creation

A complete TDP consists of all details of the particular text version including the *transforming sequence* (see "Design considerations: TPSF datastructure and TPSF datapoints" section). In each step, the TDP is enriched with the newly collected details as an incremental process. After the data processing for a particular TDP is accomplished, THEtool moves back to the input file to parse the subsequent keystroke data (if any further data is available) until an eventual version V_{i+1} is captured.

Triggering the TDP creation depends on the tool mode: In *Pause Capturing Mode* (PCM) it is a pause of a certain length in the text production process. The configuration allows to set a static threshold in milliseconds or a writer-dependent threshold calculated over several previous writing sessions using individual thresholds like median inter-key intervals (MIKI) or a factor thereof as proposed by Rosenqvist (2015). If the limit is exceeded, the current text version is captured and stored as a TDP.

In *Edit Capturing Mode* (ECM), the trigger for the TDP generation is either a detected edit operation or a change of cursor position, as the start of a different production mode. An edit operation is either appending the text, deleting or inserting a sequence without interruption, as an operationalization of *bursts*. As long as the writer keeps deleting, appending, or inserting subsequent characters one after another without changing direction or moving the cursor to a different position, the whole sequence is treated as a single edit operation. As soon as the edit operation is interrupted (i.e., a change in production mode is detected), the current text version is captured.

Extraction of transforming sequences

The characters deleted, appended, or inserted during a single edit operation are aggregated and stored in a data structure representing the *transforming sequence*: The character sequence is split into tokens, i.e., words, and each token is automatically annotated with morphosyntactic information—e.g., part-of-speech and dependency tags—, and out-of-vocabulary labels provided by respective NLP tools

depending on the language of the writing. For the full overview of the transforming sequence attributes, see Listing 3.

Listing 3 Example of a transforming sequence exported to JSON format.


```
{
        "label": "insertion",
        "text": " without interruption",
        "tags": [
                        {
                            "text": " ",
                            "pos": "SPACE",
                            "pos_details": "_SP",
                            "dep": "",
                            "lemma": " ",
                            "oov": false,
                            "is_punct": false,
                            "is_space": true
                        },
                        {
                            "text": "without",
                            "pos": "ADP",
                            "pos_details": "IN",
                            "dep": "ROOT",
                            "lemma": "without",
                            "oov": false,
                            "is_punct": false,
                            "is_space": false
                        },
                        {
                            "text": "interruption",
                            "pos": "NOUN",
                            "pos_details": "NN",
                            "dep": "pobj",
                            "lemma": "interruption",
                            "oov": false,
                            "is_punct": false,
                            "is_space": false
                        }
        ]
}
```


Note that tags for part-of-speech as well as for dependency might be incorrect due to incompleteness or ill-formedness of the sequence and depend on the robustness of the NLP tool used. It is also possible to add relevant information to each sentence currently under revision. Reparsing of a sentence is triggered by evaluating the information in the TPSF data point, only edited sentences are considered, see listing 2 for an example. There is exactly one transforming sequence per edit operation and per TDS, and this transforming sequence always belongs to one of the three edit categories: append, deletion, or insertion.

Depending on the length of the uninterrupted edit operation, the transforming sequence can contain just a single character or even multiple sentences (e.g.,

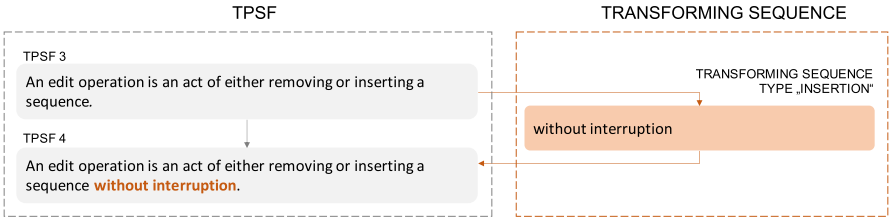


Fig. 6 Examples of multi-word transforming sequences of type insertion. The transforming sequence does not exceed the sentence boundary

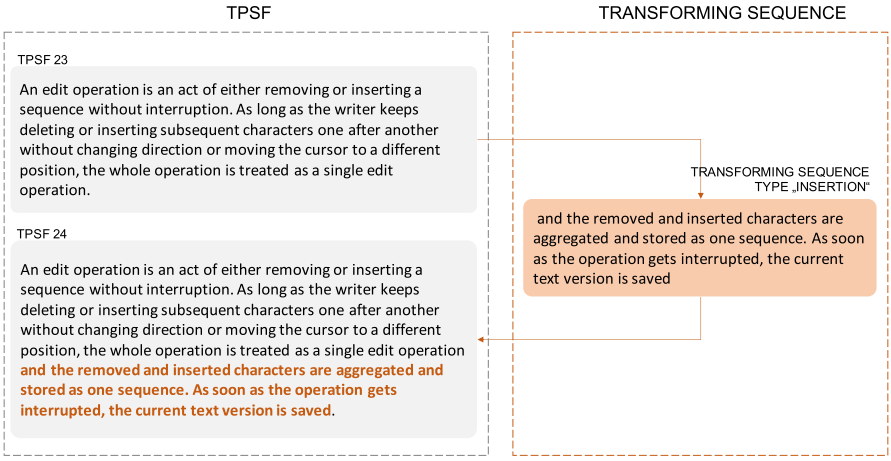


Fig. 7 An example of a transforming sequence spreading across two sentences. It contains a subsequence of a sentence (*and the removed and inserted characters are aggregated and stored as one sequence.*) and one complete sentence (*As soon as the operation gets interrupted, the current text version is saved*)

Fig. 8 Excerpt from an unfiltered text history. It contains versions affected by any edit operation including corrections of typos and spelling errors (on the left). The right column lists transforming sequences which lead from one text version to the other. They can be of type insertion or deletion. The versions detected as a result of the writer just navigating across the text are filtered out, hence the version numbers are not sequential ▶

produced by uninterrupted writing, or by pasting or deleting a longer sequence). See Figs. 6 and 7 for examples of transforming sequences from our short writing session.

The content of the transforming sequence determines which sentence-level operations have been performed, as described in "Transforming sequence" section. The transforming sequence is the main element in retrieving the deltas between sentences in the sentence processing step explained in the next section.

Sentence processing

The text versions stored as character strings in the TDPs are first automatically split into sentences using an NLP component and then automatically assigned to one of two categories: sentences which have stayed unchanged since the last edit and sentences that have been modified, inserted, or deleted—i.e., sentences under revision. To this end, the sentence lists of the current and the previous TDPs are compared to determine two sentence-level deltas: (a) sentences which exist in the current text version as stored in the TDP but do not occur in the previous ones and (b) sentences that appear in the previous text version but do not exist in the current one.

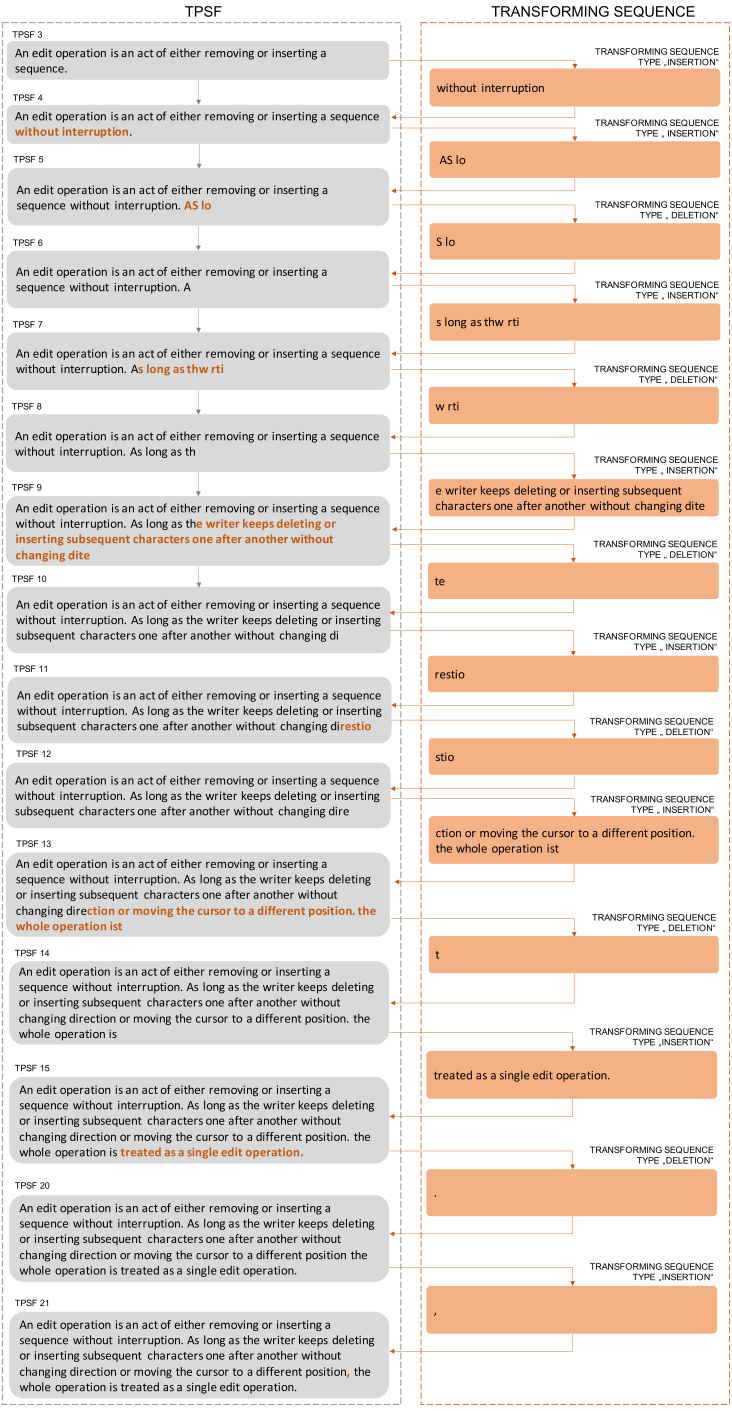
The label for sentence-level edit operations of a TDP depends on the delta and the information stored in the corresponding transforming sequence (edit operation and character sequence) as defined in "Transforming sequence" section: (a) new, (b) merge result, (c) split result, (d) modified, (e) deleted, (f) deleted due to merge, and (g) unchanged.

An important component is matching the modified sentences (including sentences resulting from merges and splits) with their versions from preceding TDPs. The previous sentence version is retrieved based on the transforming sequence and the current sentence text. This automatic pairing is a prerequisite for tracking individual sentence histories at a later stage. For each sentence, the information which sub-sequence of the transforming sequence actually impacts the particular sentence is retrieved. This transforming sub-sequence is extracted and stored as a sentence attribute.

Creation of the text history

The text history is the collection of all TDPs for a particular text; appropriate visualization allows us to examine the text production process and to gain insights into writing habits of the writer. Figure 8 provides a text-based excerpt from a text history of our example data as a sequence of versions in Edit Capture Mode (ECM).

We introduce a novel way of visualizing such text histories, shown in Fig. 9. It represents TDPs on the y-axis, and sentences on the x-axis. The evolvement of a


TPSF
TRANSFORMING SEQUENCE
TPSF 3
An edit operation is an act of either removing or inserting a sequence.
TRANSFORMING SEQUENCE TYPE „INSERTION"
without interruption
TPSF 4
An edit operation is an act of either removing or inserting a sequence without interruption.
TRANSFORMING SEQUENCE TYPE , INSERTION"
AS lo
TPSF 5
An edit operation is an act of either removing or inserting a sequence without interruption. AS lo
TRANSFORMING SEQUENCE TYPE , DELETION"
S lo
TPSF 6
An edit operation is an act of either removing or inserting a sequence without interruption. A
TRANSFORMING SEQUENCE TYPE , INSERTION"
s long as thw rti
TPSF 7
An edit operation is an act of either removing or inserting a sequence without interruption. As long as thw rti
TRANSFORMING SEQUENCE TYPE , DELETION"
w rti
TPSF 8
An edit operation is an act of either removing or inserting a sequence without interruption. As long as th
TRANSFORMING SEQUENCE TYPE , INSERTION"
e writer keeps deleting or inserting subsequent characters one after another without changing dite
TPSF 9
An edit operation is an act of either removing or inserting a sequence without interruption. As long as the writer keeps deleting or inserting subsequent characters one after another without changing dite
TRANSFORMING SEQUENCE TYPE , DELETION"
te
TPSF 10
An edit operation is an act of either removing or inserting a sequence without interruption. As long as the writer keeps deleting or inserting subsequent characters one after another without changing di
TRANSFORMING SEQUENCE TYPE , INSERTION"
restio
TPSF 11
An edit operation is an act of either removing or inserting a sequence without interruption. As long as the writer keeps deleting or inserting subsequent characters one after another without changing direstio
TRANSFORMING SEQUENCE TYPE , DELETION"
stio
TPSF 12
An edit operation is an act of either removing or inserting a sequence without interruption. As long as the writer keeps deleting or inserting subsequent characters one after another without changing dire
TRANSFORMING SEQUENCE TYPE , INSERTION"
ction or moving the cursor to a different position. the whole operation ist
TPSF 13
An edit operation is an act of either removing or inserting a sequence without interruption. As long as the writer keeps deleting or inserting subsequent characters one after another without changing direction or moving the cursor to a different position. the whole operation ist
TRANSFORMING SEQUENCE TYPE , DELETION"
t
TPSF 14
An edit operation is an act of either removing or inserting a sequence without interruption. As long as the writer keeps deleting or inserting subsequent characters one after another without changing direction or moving the cursor to a different position. the whole operation is
TRANSFORMING SEQUENCE TYPE „INSERTION"
treated as a single edit operation.
TPSF 15
An edit operation is an act of either removing or inserting a sequence without interruption. As long as the writer keeps deleting or inserting subsequent characters one after another without changing direction or moving the cursor to a different position. the whole operation is treated as a single edit operation.
TRANSFORMING SEQUENCE TYPE „DELETION"
.
TPSF 20
An edit operation is an act of either removing or inserting a sequence without interruption. As long as the writer keeps deleting or inserting subsequent characters one after another without changing direction or moving the cursor to a different position the whole operation is treated as a single edit operation.
TRANSFORMING SEQUENCE TYPE „INSERTION"
‚
TPSF 21
An edit operation is an act of either removing or inserting a sequence without interruption. As long as the writer keeps deleting or inserting subsequent characters one after another without changing direction or moving the cursor to a different position, the whole operation is treated as a single edit operation.

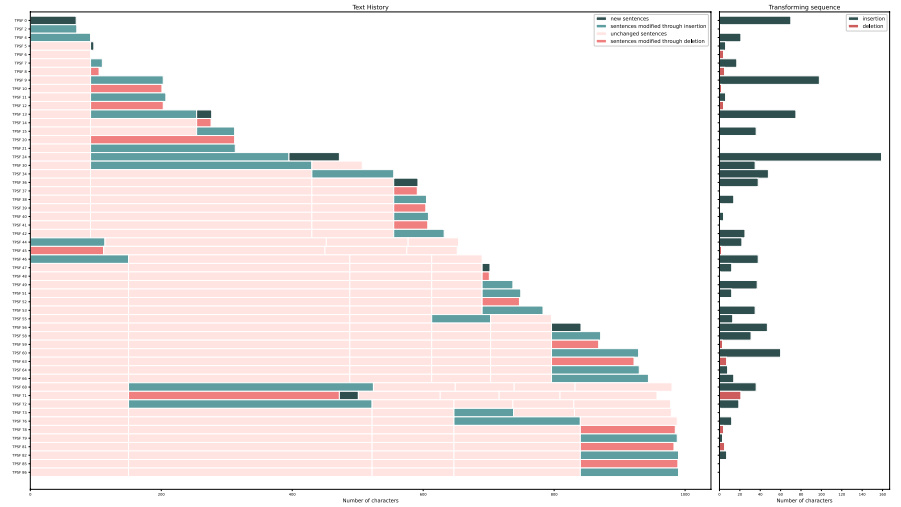


Fig. 9 Text history containing all versions affected by any edit operation including corrections of typos and spelling errors. The versions generated as a result of the writer just navigating across the text are filtered out, hence the version numbers (on the y-axis) are not sequential

text and the corresponding processes can be traced by starting at the top left and then moving downward along the y-axis. The highlighted part per TDP shows which sentence was the target of the edit that triggered this TDP and color-code how the affected sentences were classified. The right part of the plot shows the relative length of the transforming sequence in the unfiltered text history visualization and the sequence of edits preceding the relevant TPSF in the filtered version. The color scheme indicates what kind of edit operation was performed. This visualization offers a compact overview of the flow of the text production and enables a quick visual comparison of writing habits between writers.

Creation of sentence histories

The sentence histories are derived from the text history. THEtool iterates over the TDPs and retrieves new, modified, deleted, and unchanged sentences. Each new sentence discovered in the particular TDP receives a unique id. Modified, deleted, and unchanged sentences which occur in subsequent versions are appended to the existing sentence lists. For modified sentences, the previous version is used for matching against the sentence already stored in the sentence history.

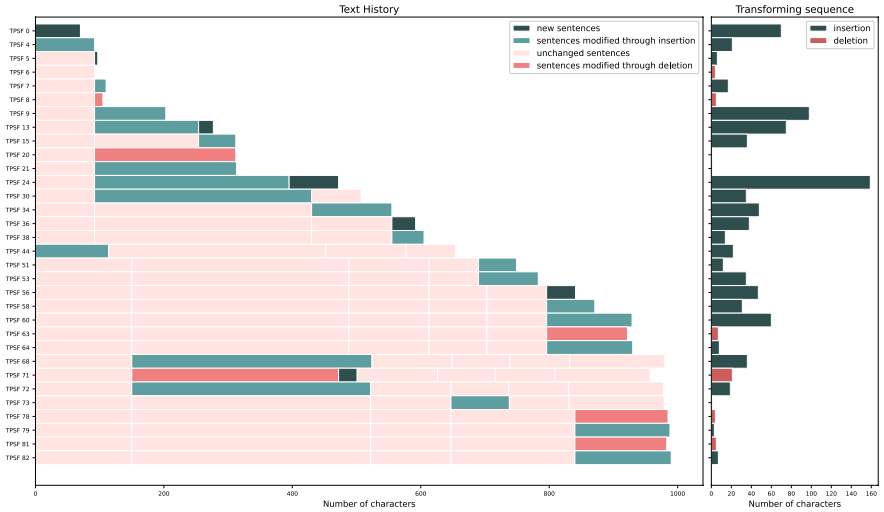


Fig. 10 Text history containing only versions affected by morphosyntactically relevant edit operations. This is the filtered version of the text history presented in Fig. 9

Listing 4 Sentence history example. History of the first sentence from the example text presented in Figure 8.

```
"198991558581007011128590587426244661538": [
    {
        "text": "An edit operation is an act of either removing or inserting a sequence",
        "start_index": 0,
        "end_index": 69,
        "pos_in_text": 0,
        "label": "new",
        "revision_id": 0,
        "sentence_morphosyntactic_relevance": true
    },
    {
        "text":  "An edit operation is an act of either removing or inserting a sequence without
              interruption.",
        "start_index": 0,
        "end_index": 91,
        "pos_in_text": 0,
        "label": "modified",
        "revision_id": 4,
        "sentence_morphosyntactic_relevance": true
    },
    {
        "text":  "An edit operation is an act of either removing or inserting a sequence at the beginning,
             int without interruption.",
        "start_index": 0,
        "end_index": 113,
        "pos_in_text": 0,
        "label": "modified",
        "revision_id": 44,
        "sentence_morphosyntactic_relevance": true
    },
    {
        "text":  "An edit operation is an act of either removing or inserting a sequence at the beginning,
              in the middle or at the end of the text without interruption.",
        "start_index": 0,
        "end_index": 113,
        "pos_in_text": 0,
        "label": "final_version",
        "revision_id": 86,
        "sentence_morphosyntactic_relevance": true
    }
]
```

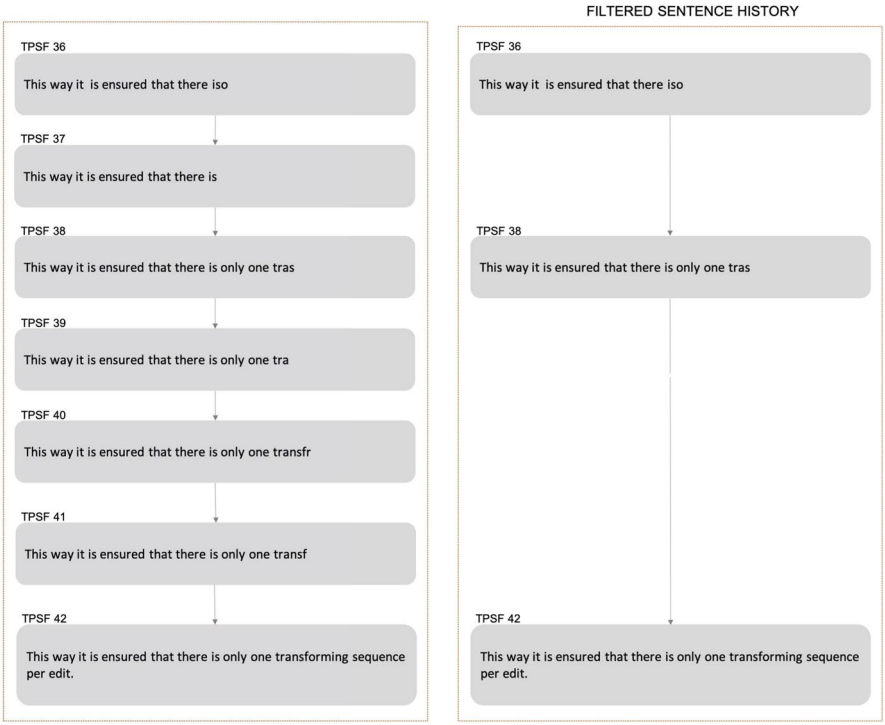


Fig. 11 Excerpt from the history of the fourth sentence from the example text shown in Fig. 8. The list on the left contains all sentence versions including the ones which are labeled as irrelevant. The list on the right is filtered according to the filtering criteria described in "Relevance" section

Figure 11 provides an example of an unfiltered history (on the left) and a filtered history (on the right) of the fourth sentence from our example shown in Fig. 8. The filtered history excludes potential corrections of spelling errors and typos, Listing 4 shows the JSON format.

Relevance evaluation and filtering

Depending on the research question or final application, not all of the versions detected may be of interest. Relevance depends on the research focus and serves as basis for the filter as last step of the TDPs' processing. The current implementation of THEtool offers a number of parameters allowing to configure the relevance. The researcher can define the minimal edit distance between versions and the minimal number of tokens building the transforming sequence. It is also possible to toggle spell checking while determining relevance or include or exclude punctuation-related edits—these parameters are language-dependent and might involve the integration of specific NLP tools.

The automatic relevance evaluation results in two variants of the text history: the full one containing all edits with consecutive version numbers and the shorter filtered one where some versions are excluded. Figures 9 and 10 provide examples of unfiltered and filtered text histories for the example text. The filtered history is an attempt to retrieve only morphosyntactically relevant TPSFs (see "First experiments" section for details on our definition of morphosyntactic relevance).

The relevance evaluation results not only in classifying versions in the text history, the same label is also set for each modified sentence in a particular TDP. This information is then applied to sentence histories. The filtering can be used to perform sentence analysis and syntactic sentence parsing only for relevant sentence versions according to the research interest or final application.

Findings

We conducted a small-scale experiment to apply our method, to identify issues concerning the implementaion, and to discover points of interest from the writing researcher's perspective. The results are intended to stimulate the design of further larger scale studies both in natural writing settings as well as in experimental setups.

First experiments

Seven subjects (three female, four male; aged 34–62; all with an academic background) wrote short descriptive comments after watching a 2-min video on how to assemble a DIY macro lens for a smartphone on Youtube.[7] They wrote for about 10 min in German. The texts had a median length of 236 words, ranging from 118 to 374 words, and the average length was 196 words. The subjects used the current version of Scriptlog (Johansson et al., 2018; Strömqvist & Karlsson, 2002). We did not control for typing skills and habits (i.e., monitor gazers vs. keyboard gazers Johansson et al., 2010).

All writers can be considered experienced writers, i.e., they are familiar with the genre, the topic and communicative intention, and at a high proficiency level concerning the language—they have metarhetorical, metastrategic, and metalinguistic awareness, as described by Horning (2006). The subjects agreed to make the data available for scientific purposes as supplementary material,[8] consisting of the text produced and the idfx file for each participant. As we were focusing on exploring revising and editing and did not pose any restrictions except guiding questions we did not consider demographic data for our analyses.

[7] https://www.youtube.com/watch?v=KsKoKWyU8J8.

[8] Also available on GitHub.

Tool setup

We applied THEtool in ECM mode, i.e., versions are determined according to changes in production mode.[9] We did not use temporal aspects.

To obtain sentences and words from the text versions stored as character strings in the TDPs, we applied *spaCy* for sentence segmentation and tokenization, an open-source Python software library for advanced NLP (Honnibal et al., 2020). spaCy offers a set of trained pipeline packages for multiple languages. We used the German language model *de_core_web_md* of spaCy's version 3.0.6. While the results of spaCy's sentence splitting seemed faultless for complete sentences, for incomplete sentences, the sentence boundaries proved incorrect in some cases. Hence, simple heuristics have been implemented to improve sentence splitting, e.g., checking if the sentence starts with an uppercase letter and if it contains a final punctuation. "Discussion" section provides examples on the sentence segmentation issues.

Our aim was to retrieve only edits that potentially impact the syntactic structure of the sentence to allow for reliable automatic analysis and interpretation of the text production from a syntactic perspective: If a syntactic parser is robust enough to successfully process input containing spelling errors or typos, there is no need to reparse the sentence after the writer corrected (or introduced) such minor issues. We thus need to identify edits that potentially impact the syntactic analysis—we call those edits *morphosyntactically relevant*. For example, German orthography requires nouns to be capitalized, but POS taggers and syntactic parsers do not exclusively rely on this feature, i.e., changing *das wort* to *das Wort* in a sentence will not result in a different syntax tree and is considered irrelevant for our purpose.[10] We therefore applied *morphosyntactic relevance* filtering, i.e., the filtered text and sentence histories do not include likely corrections of typos and spelling errors.

Morphosyntactic relevance was configured as follows: the minimum edit distance between the versions must be three or the transforming sequence must contain at least two tokens. Spell checking is disabled to accept versions containing typos—the subsequent syntactic parsing is assumed to handle this input gracefully—and an edit merely referring to punctuation is considered irrelevant. Note, however, that this step involves basic NLP, as the insertion or deletion (or replacement) of a single character could be a correction—and thus deemed irrelevant—, a word-level change resulting in a different part-of-speech (POS), in a semantic change, or both—and therefore considered relevant. Filtering for morphosyntactic relevance is thus not trivial.

[9] See "Relevance evaluation and filtering" section for parameters that trigger relevant text versions in ECM mode.

[10] Note that there might be specific research interests aiming exclusively on correction of spelling errors which can be achieved by applying other filtering criteria.

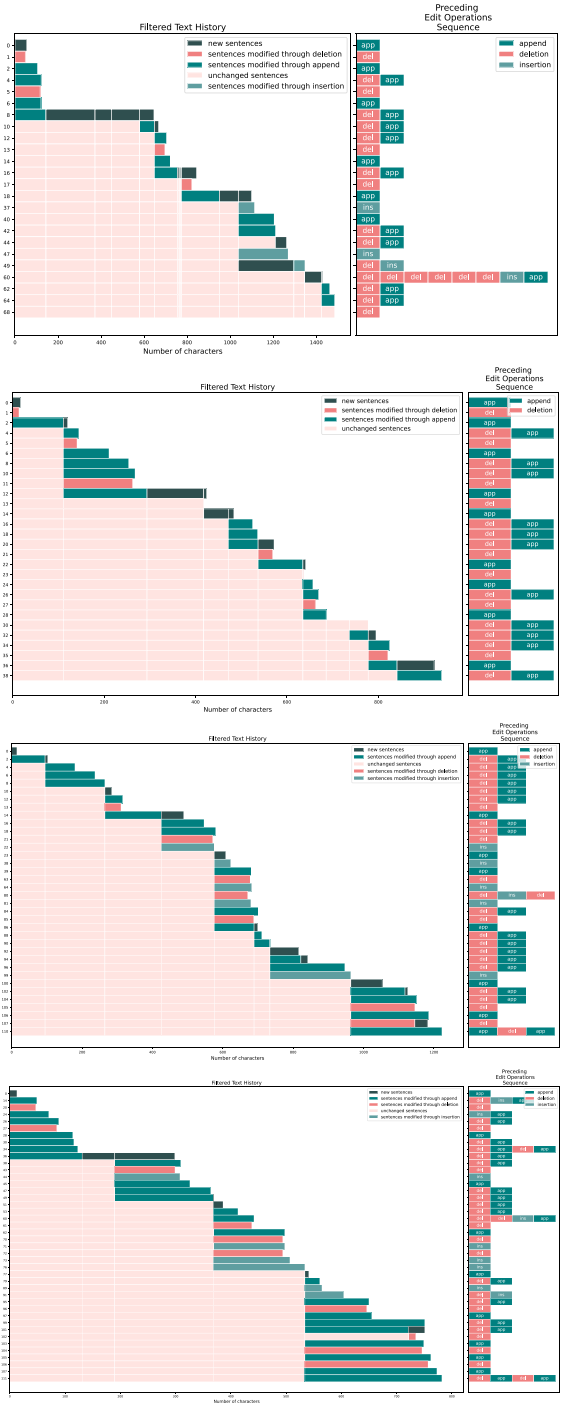


Fig. 12 Text histories of subjects D, J, K, and M

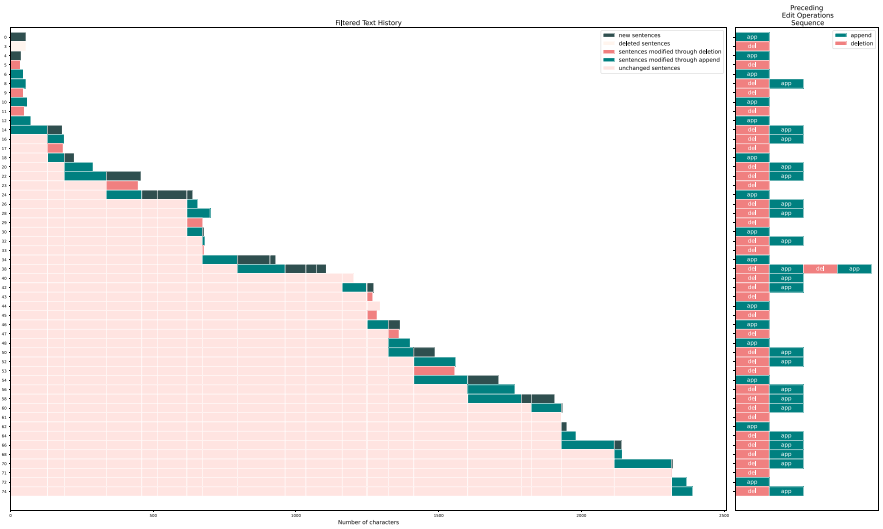


Fig. 13 Text history of subject R

Interpretation of text histories and visualization

When inspecting the seven text histories extracted by applying THEtool to the seven idfx files produced by our subjects, they can be categorized in several ways: (a) by general revision habits—subjects who edited mostly in the current sentence or at the point of inscription (D, J, K, M, and R see Figs. 12, 13) and subjects who sometimes went farther back for editing something (A and C, see Fig. 14)—and (b) by length of the final product (D, J, K, and M produced significantly shorter texts than A, C, and R).

Two of the three writers who wrote longer texts also edited farther back (A and C), but R, who wrote the longest text, mainly revised current sentences. R also mostly carried out small edits to potentially correct for spelling.

Next, we compare R, A, and C in more detail. A writes rather short sentences to produce 269 words. The history of R contains fewer versions for producing 374 words, the last version is V_{74}. R's sentences are also longer then the ones produced by A. We can detect 111 versions for C and 360 versions for A. This means that R changes production mode less often than A and C. The writer just continues writing and adds word after word to the current sentence. While we can extract 138 *relevant* versions for A and 69 for C (the number of lines shown in Fig. 14), i.e., roughly twice as many, A produces more than three times the number of *detectable* versions compared to C (360 for A, 111 for C). Edits by A are most likely typo corrections and are therefore frequently filtered; we will have a look into a sentence history of A in "First experiments" section. Revisions by R occur mostly in the current sentence. The writer does not go back far in the TPSF, while A and C tend to do so but only occasionally. For R, we see heavy revisions at the very beginning: the writer tries

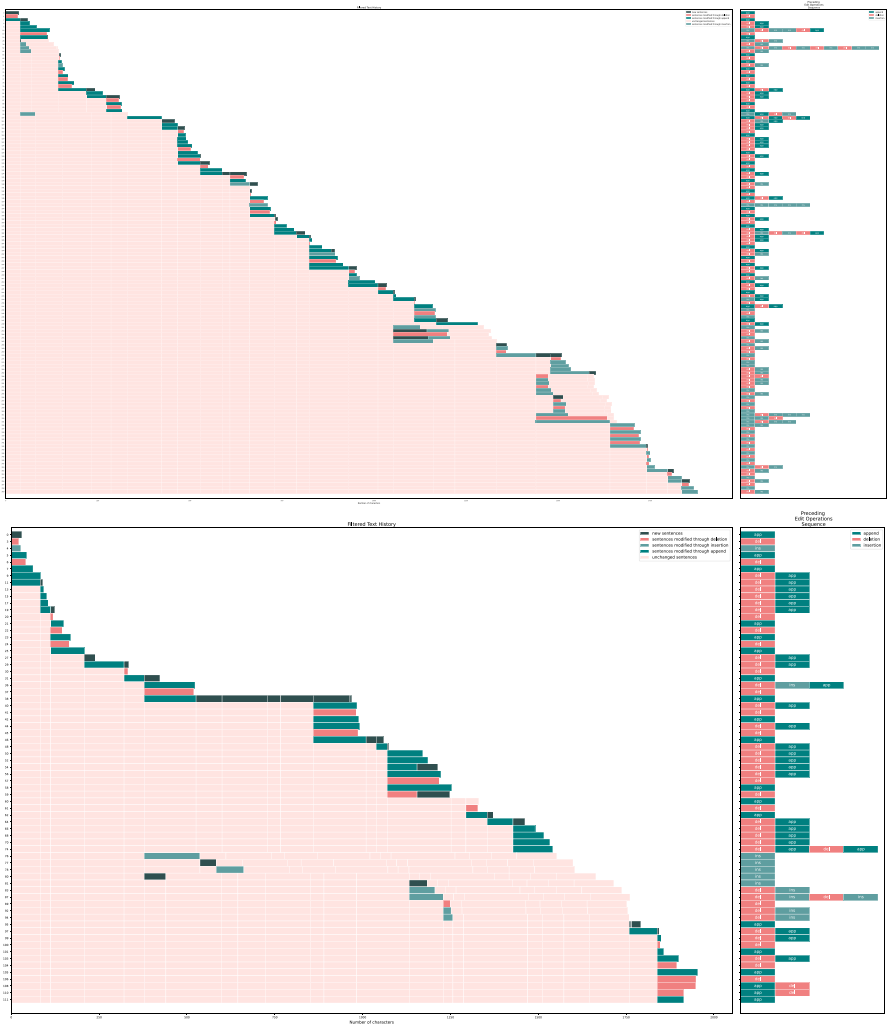


Fig. 14 Comparison of text histories of subjects A and C

various starts but then seems to reach some kind of flow and just writes in an almost linear fashion.

These first insights into revision strategies support the findings by Bowen and Van Waes (2020), i.e., that revisions may most frequently occur at or just before the point of inscription, which Bowen and Van Waes base on detailed exploration of writing sessions by one writer. Our method will allow for systematic investigation in this direction on a larger scale.

Interpretation of sentence histories

We show the sentence histories of two subjects—K and A—to illustrate the kind of insights gained by applying our approach. We first show a sentence produced by subject K in Example 1:[11]

Example 1 Versions of a sentence produced by K. The numbers in parentheses are version indicators. Filtered versions have been excluded:

(37) *Als Beispiel wird die Makroaufna* ('As example the macro...is')

(55) *Als Beispiel wird unter anderem die Makroaufnahme einer Platine vorgeführt, was die sinnvollste Anwendung* ('As example among others the macro shot of a circuit board is demonstrated, which the most sensible use')

(63) *Als Beispiel wird unter anderem die Makroaufnahme einer Platine vorgeführt, was sinnvollste Anwendung* ('As example among others the macro shot of a circuit board is demonstrated, which most sensible use')

(79) *Als Beispiel wird unter anderem die Makroaufnahme einer Platine vorgeführt, was eine sinnvolle Anwendung* ('As example among others the macro shot of a circuit board is demonstrated, which a sensible use')

(80) *Als Beispiel wird unter anderem die Makroaufnahme einer Platine vorgeführt, was eine Anwendung* ('As example among others the macro shot of a circuit board is demonstrated, which a use')

(84) *Als Beispiel wird unter anderem die Makroaufnahme einer Platine vorgeführt, was eine nützliche Anwendung scheint, die anderen* ('As example among others the macro shot of a circuit board is demonstrated, which seems to be a useful use, which others')

(110) *Als Beispiel wird unter anderem die Makroaufnahme einer Platine vorgeführt, was eine nützliche Anwendung scheint.* ('As example among others the macro shot of a circuit board is demonstrated, which seems to be a useful use.')

According to the tool setup, potential corrections of typos and spelling errors are heuristically filtered out to ensure that the difference between two relevant versions is most probably rooted in syntax. The writer stops while writing a compound (V_{37}) and extends the TPSF by inserting a particle construction (*unter anderem* ('amongst others')) as a connector before the determiner of the complex noun phrase to which the unfinished compound belongs. K only then finishes writing the noun *Makroaufnahme* ('macro shot'). The writer makes further revisions before producing a complete sentence: the noun phrase in the relative clause at the edge of V_{55} is altered by first replacing the definite determiner with the indefinite one (sequence of V_{55}, V_{63}, V_{79}), which requires adjusting the adjective for agreement, and then the adjective *sinnvolle* ('sensible') is replaced by the nearly, but not quite synonymous adjective *nützliche* ('useful'). Only then K finishes the clause and starts writing the

[11] Glosses provided here are neither intended to be idiomatic translations nor are they aligned on the word level, but aim to capture the meaning and to approximate the German syntax.

next one. But before finishing this clause (, *die anderen*), K decides to get rid of it and to end the sentence with the relative clause already present in V_{84}.

For comparison, consider Example 2 where we show unfiltered versions extracted for a sentence that subject A wrote. From those, only versions V_1, V_2, and V_9 would be considered relevant—A replaces *YouTube* by *Sachgeschichten*. Versions $V_{3\ldots8}$ involve corrections of typos only and are thus not be considered relevant for our purpose. We can verify whether this filtering works correctly by manually inspecting all detected versions. They effectively contain typos and their corrections but are not spelling errors as the characters typed and deleted are produced by adjacent keys on the keyboard: the writer tries to correct them immediately but sometimes hits the wrong key again.

Example 2 Versions of a sentence produced by A:

(1) *Die Aufmachung ist ganz im YouTube-Stil.* ('The presentation is totally YouTube style.')
(2) *Die Aufmachung ist ganz im -Stil.*
(3) *Die Aufmachung ist ganz im Sax-Stil.*
(4) *Die Aufmachung ist ganz im Sa-Stil.*
(5) *Die Aufmachung ist ganz im Saxch-Stil.*
(6) *Die Aufmachung ist ganz im Sa-Stil.*
(7) *Die Aufmachung ist ganz im Sachgeschc-Stil.*
(8) *Die Aufmachung ist ganz im Sachgesch-Stil.*
(9) *Die Aufmachung ist ganz im Sachgeschichten-Stil.* ('The presentation is totally Learning Stories[12] style.')

Discussion

We first evaluate the correctness of the filtering for morphosyntactic relevance and of the sentence segmentation. We then discuss implications of our experiments for writing research and practical applications.

Error analysis

Automatic detection of actual spelling errors or typos and their correction to exclude morphosyntactic irrelevant edits that would be handled gracefully by a robust syntactic parser is a general challenge. An affected word may either be spelled incorrectly, a typo, or an incomplete word inserted by the writer on purpose, merely not finished.

We randomly sample the text and sentence history of one author, C, and manually compare the filtered text history with the unfiltered one to calculate precision (how many detected TPSFs are actually morphosyntactically relevant) and recall (how

[12] Referring to the "Lach- und Sachgeschichten" from the "Sendung mit der Maus" TV program.

many of these relevant edits are found by the filtering). The filtering produces 69 TPSFs of which 52 are relevant according to our manual evaluation. This yields a precision of 76% for the morphological filtering of author C's text history. The spurious TPSFs are mainly caused by tokens being regarded as correct German words. In one instance, writer C produced the sequence *hab e i* ('hav e I') and then changed it to *habe ich* ('have I'), but *hab* ('hav') is regarded as correct.

For recall, we manually detect 53 relevant edits in the text history. The morphosyntactic filtering recognizes 52 of these, which gives a recall of 98%. The one error occurred because the minimum length for edits (3 characters) prevented the recognition of a removed preposition *um* ('to').

The decision to rely on an established spell checker like LanguageTool might affect the overall performance of our implementation. Here, detailed evaluations on larger data sets are necessary. The combination with criteria developed by Conijn et al. (2019) might improve performance and accuracy and is left for future development.

Our evaluation results in an F1-Score (harmonic mean of precision and recall) of 86%.

The manual evaluation of this first version of the morphosyntactic filtering in the THEtool suggests that it is a valid approach when aiming at triggering automatic syntactic parsing only for versions resulting from morphosyntactically relevant edits. The discussion among the authors of the manual annotation which was performed by one author, there was uncertainty regarding the relevancy of a TPSF and how to annotate it properly in some cases. So clearly the definition of *relevance* has to be done very carefully in line with the specific research question, to be able to operationalize and implement it properly by configuring respective parameters.

For sentence splitting, we used a pretrained spaCy model, which worked faultlessly for complete sentences. However, parsing an inserted, incomplete sentence resulted in erroneous segmentation in some cases, which in turn deteriorated the accuracy of the sentence matching and classification when creating the sentence histories.

For the error analysis of the sentence segmentation, we selected the same text as for morphosyntactic filtering evaluation. Our analysis showed that author C produced 22 sentences in the final version. Manually investigation of the sentence history generated by THEtool found three sentences where splitting proved incorrect but only in some intermediate versions:

- *Dass es nur 2 Minuten sind, hat mich überrascht, weil ja doch sehr viel gezeigt wird: die Idee, das Ausbauen, das Draufmontieren, das Ausprobieren und dann noch eine Zusammenfasseung.* ('That it is only 2 min, did surprise me, because a lot is shown: the idea, the dismanteling, the assembling, the testing and then also a summary.') The sentence has been split incorrectly in intermediate versions three times at the colon. The correct segmentation could be observed in eight cases.
- *Manches ist natürlich logisch, weil die Perspektive wechselt: man sieht den Sprecher von vorn und dann wieder von oben nur seine Hände.* ('Something is

of course logical, because the perspectives changes: one sees the speaker from the front and then again from above only his hands.') The segmentation of this sentence has been incorrect in all intermediate versions, again at the colon. As a result, only fragments of the sentence occur in the sentence history: *(a) sieht den Sre, (b) man sieht den S, (c) man sieht den Sprecher von vorn und dann wieder von oben nur seine Hände.*

- *Und dass die Aufnahmen dann so daneben eingeblendet werden, ist auch gut gelöst.* ('And that the recordings then are next to it faded in, is also well solved.') This sentence has been split incorrectly in two intermediate versions, but has been segmented correctly segmented once completed.

The erroneous segmentation caused confusion in the TPSFs delta analysis and resulted in another error in the sentence history: one sentence has not been discovered at all and is missing in the sentence history.

The incorrect segmentation impacted intermediate versions of 4 out of 22 sentences. From a total of 113 sentence versions stored in the sentence history, the segmentation was faultless in 101 cases.

Implications for future lines of research

As we had a small number of participants in our exploratory study, we cannot draw any general conclusions from these facts but report them as observable evidence, which serves as starting point for the design of specific experiments. Edits like those by K—changing definiteness of a complex noun phrase—involve adjustments to ensure agreement in languages that feature a rich morphology. Such edits are prone to produce side effects, as Mahlow (2015b) shows. We still lack systematic analyses of editing goals and effects of attempts to achieve them to truly understand writing processes on a linguistic level. We take these first insights as a starting point for the design of experiments aimed at the exploration of production of specific linguistic structures like complex noun phrases or multi-word expressions.

Our method provides a way to show that experienced writers actually revise along syntactic structures and that analyses of revisions and bursts should take linguistic units into account, thus leading to linguistically motivated analyses beyond pauses and sequences of characters. For example, considering all relevant sentence histories and looking at the text history, we can characterize the writing habit of A as writing in a sequence of short production episodes and immediate revisions at the point of inscription.

Our approach actually offers new possibilities to inspect individual writing sessions and gain insights in a novel way. The two examples of sentence histories shown enable detailed analyses for individual writing sessions on a larger scale.

For a practical application, one could explore the text and sentence histories in consultancy sessions with the writer. Writers would probably not be able to judge their own writing habits and styles or draw recommendations on how to tackle specific writing tasks just by introspection. Applying the concept of *text history* and *sentence history* together with an appropriate form of the *transforming sequence* to writing process data will allow the implementation of visualizations to provide

writers with another view on their own writing process and allows them to reflect on habits and potential reasons for struggle and how to overcome them by looking for specific support. For example, writer A is a fast typist but has a high percentage of typos, and thus might profit from personalized predictive texting.[13]

Conclusion, future work, and collaboration

With the work reported here, we contribute towards the general aim of analyzing writing process data in novel ways. We have introduced the concept of transforming sequences covering writing including revising and editing. We have proposed text history and sentence history as concepts for covering these aspects. With the TPSF data structure (TDS) and TPSF data points (TDP) we have created data structures that combine process and product data. These data structures allow for further detailed processing of text histories and sentence histories, including deeper use of NLP which we did not report on here. Possible NLP applications beyond POS tagging and dependency labeling include co-reference resolution, multiword extraction, and discourse analysis. We will apply respective tools in further experiments to explore creation and revising of respective units. They could be run on sentences that had been changed recently as well as the complete TPSF at a specific point of the writing process as THEtool allows specific access to those data.

We have introduced and implemented an approach for capturing, exploring, and visualizing these structures. Here, the goal is to provide a robust, high-quality, fully automatic linguistic markup of the text and sentence histories. This enables writing process researchers to analyze large amounts of writing process data without the need for laborious and cumbersome manual annotation work. Our implementation is an operationalization of *bursts*, but currently does not take into account the fine-grained taxonomy developed by Baaijen et al. (2012). Consistent automatic extraction and labeling could serve as test bed for these and potentially contribute to further improvement and refinement of criteria for distinguishing bursts.

In a first experiment, we have applied THEtool as proof of concept on raw keystroke-logging data for linguistically motivated, comparative analyses of writing processes and writers' habits. As we have observed in our initial experiment, writers do revise on linguistic levels and switch production mode even mid-word. Also, they do not necessarily choose complete syntactic constituents as the targets for substitutions and edits, which poses a challenge for current automatic syntactic parsers. From this observation, we can define two lines of future work: (a) Our approach enables the processing of large amounts of writing sessions. Automatic categorization of versions and transforming sequences both on the product and

[13] In fact, writer A told us later that they usually use the software Typinator in all applications involving typing, but which did not help them when using Scriptlog as it requires words to actually be typed. This is in line with observations by Weder (2010) who reports that subjects in her experiments usually rely on spell checkers.

the process level can serve as starting point for systematic design of new and evaluation of established revision taxonomies. For example: instances of immediate insertion after deletion to detect actual replacements could automatically be further distinguished as meaning-changing or meaning-preserving by considering morphosyntactic features like part-of-speech—as shown in Example 1 for the adjective in a complex noun phrase. (b) A genuine contribution to NLP would be the development of robust syntactic parsers that are able to gracefully process such syntactically incomplete linguistic units, as well as improving the quality of the morphosyntactic filtering and sentence segmentation and alignment.

A combination of our method with previous attempts to work on specific smaller linguistic units is possible and opens new research questions. Example 1 included an instance where the writer stopped while typing a word to either delete it or go back and revise. The change in production mode occurred *within* a word. Studies on pausing within words as by Weingarten et al. (2004) can complement our method: investigating whether change in production mode within words occurs at boundaries where writers also would pause but then continue writing would allow to gain more insights into cognitive aspects involved when changing production mode.

For a practical application, we aim for live processing to enable visualization or specific support for writers based on morphosyntactic information *during* writing. The visualization of the syntactic evolvement of sentences will complement the established representation of changes within words. Variants of the visualizations shown here could be included in dashboards used in writing counseling as proposed by Conijn et al. (2020). The *relevance of editing* will help us to identify editing at the linguistic constituent level, e.g., replacement of one preposition by another, replacement of a complex noun phrase by a pronoun, or deletion of phrases. The ability to trace such edits (e.g., a writer may have tried two or more alternative variants before settling on a choice) will contribute to the detection of the origins of side effects of editing and provide insights into how to offer appropriate support for error prevention or correction.

We briefly explored the use of spaCy as a dependency parser to annotate text and sentence versions with syntactic information. While POS tagging was mostly robust for grammatically incomplete sentences, dependency parsing proved to be more error-prone. When well-formed constituents were edited or replaced with other well-formed constituents of the same syntactic type the parsing was more accurate. However, the writers in our experiments did not always produce complete syntactic constituents before switching production mode (for example, they replaced an adjective they had just written immediately with another adjective before producing the head noun of the noun phrase). spaCy struggled especially in these cases to generate correct syntactic parses of incomplete sentences. Syntactic parsing of incomplete sentences in text production research thus remains an open problem we aim to address in future work.

Another phenomenon we observed but did not show in detail here was that some writers apparently have gotten used to rely on auto correction for sentence starts and word starts. If one releases the shift key too slowly, the two leading characters of a word end up in upper case and the second one gets auto-corrected

into lower case in many applications—a feature Scriptlog does not offer. We therefore will have to inform subjects about the features available in editors for dedicated writing experiments to reduce frustration for the writer and to mitigate arbitrary effects in the logging data and the final product that are not typical for the writer. As a community, we should also work on how to observe and analyse writing at the workplace or in everyday situations where people use physical or virtual keyboards and a mix of input methods.

To support future research on sentence and text-level writing processes, we release the implementation of THEtool publicly at github https://github.com/mulasik/wta and invite collaboration on its further development.

Acknowledgements We thank Åsa Wengelin who provided the recent version of Scriptlog, and Klaus Rothenhäusler who packaged it for easy installation on various operating systems. We thank the seven colleagues and friends who agreed to write for us and to make their data available. We also thank the three anonymous reviewers whose comments and questions helped us to improve the article and the current implementation.

Funding No specific funding.

Availability of data and material (data transparency) and code availability All data and code is available open-source at our GitHub repository at https://github.com/mulasik/wta.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Alamargot, D., & Chanquoy, L. (2001). General introduction. In D. Alamargot & L. Chanquoy (Eds.), *Through the models of writing, studies in writing* (Vol. 9, pp. 1–29). London: Kulwer.

Allal, L., & Chanquoy, L. (2004). Introduction: Revision revisited. In L. Allal, L. Chanquoy, & P. Largy (Eds.), *Revision. Cognitive and instructional processes, studies in writing* (Vol. 13, pp. 1–7). Boston: Kluwer.

Baaijen, V. M., Galbraith, D., & de Glopper, K. (2012). Keystroke analysis. *Written Communication, 29*(3), 246–277. https://doi.org/10.1177/0741088312451108.

Ballier, N., Pacquetet, E., & Arnold, T. (2019). Investigating Keylogs as time-stamped graphemics. In Y. Haralambous (ed.) *Proceedings of graphemics in the 21st century, Brest 2018*, Fluxus Editions, Brest (pp. 353–365). https://doi.org/10.36824/2018-graf-ball.

Bereiter, C., & Scardamalia, M. (1987). *The psychology of written composition*. Hillsdale, NJ: Lawrence Erlbaum.

Bowen, N. E. J. A. (2019). Unfolding choices in digital writing: The language of academic revisions. *Journal of Writing Research, 10*(3), 465–498. https://doi.org/10.17239/jowr-2019.10.03.03.

Bowen, N., & Van Waes, L. (2020). Exploring revisions in academic text: Closing the gap between process and product approaches in digital writing. *Written Communication, 37*(3), 322–364.

Bridwell, L. S. (1980). Revising strategies in twelfth grade students' transactional writing. *Research in the Teaching of English,14*(3), 197–222.

Chenoweth, N. A., & Hayes, J. R. (2003). The inner voice in writing. *Written Communication, 20*(1), 99–118. https://doi.org/10.1177/0741088303253572.

Cislaru, G., & Olive, T. (2018). *Le processus de textualisation*. De Boeck Supérieur, Louvain-la-Neuve: Analyse des unités linguistiques de performance écrite.

Clément, L., Gerdes, K., & Marlet, R. (2011). 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) Berlin: Springer. https://doi.org/10.1007/978-3-642-20169-1_4.

Conijn, R., Speltz, E. D., van Zaanen, M., Waes, L. V., & Chukharev-Hudilainen, E. (2021). A product- and process-oriented tagset for revisions in writing. *Written Communication*. https://doi.org/10.1177/07410883211052104.

Conijn, R., Van Waes, L., & van Zaanen, M. (2020). Human-centered design of a dashboard on students' revisions during writing. In C. Alario-Hoyos, M. J. Rodríguez-Triana, M. Scheffel, I. Arnedillo-Sánchez, & S. M. Dennerlein (Eds.), *Addressing global challenges and quality education* (pp. 30–44). Cham: Springer.

Conijn, R., van Zaanen, M., Leijten, M., & Van Waes, L. (2019). How to typo? Building a process-based model of typographic error revisions. *The Journal of Writing Analytics, 3,* 69–95.

Cookson, S. (1989). Designing computational writing tools within a linguistic model of the writing process. In N. Williams & P. O. Holt (Eds.), *Computers and writing: Models and tools* (pp. 17–21). Oxford: Intellect.

Cook, P., & Welsh, J. (2001). Incremental parsing in language-based editors: User needs and how to meet them. *Software: Practice and Experience, 31*(15), 1461–1486. https://doi.org/10.1002/spe.422.

Costa, F., Frasconi, P., Lombardo, V., & Soda, G. (2003). Towards incremental parsing of natural language using recursive neural networks. *Applied Intelligence, 19*(1–2), 9–25. https://doi.org/10.1023/a:1023860521975.

Dale, R. (2020). Natural language generation: The commercial state of the art in 2020. *Natural Language Engineering, 26*(4), 481–487. https://doi.org/10.1017/S135132492000025X.

Dale, R., & Douglas, S. (1996). Two investigations into intelligent text processing. In M. Sharples & T. van der Geest (Eds.), *The new writing environment: Writers at work in a world of technology* (pp. 123–145). Berlin: Springer.

Daxenberger, J., & Gurevych, I. (2012). A corpus-based study of edit categories in featured and non-featured Wikipedia articles. In *Proceedings of COLING 2012, The COLING 2012 organizing committee*, Mumbai, India (pp. 711–726). https://aclanthology.org/C12-1044.

Daxenberger, J., & Gurevych, I. (2013). Automatically classifying edit categories in Wikipedia revisions. In *Proceedings of the 2013 conference on empirical methods in natural language processing, association for computational linguistics, Seattle, Washington, USA* (pp. 578–589). https://aclanthology.org/D13-1055.

Ehrensberger-Dow, M., & Perrin, D. (2009). Capturing translation processes to access metalinguistic awareness. *Across Languages and Cultures, 10*(2), 275–288. https://doi.org/10.1556/acr.10.2009.2.6.

Faigley, L., & Witte, S. (1981). Analyzing revision. *College Composition and Communication, 32*(4), 400–414. https://doi.org/10.2307/356602.

Farzindar, A.A., & Inkpen, D. (2020). *Natural language processing for social media*, 3rd edn. No. 13 in Synthesis lectures on human language technologies. Morgan & Claypool Publishers LLC. https://doi.org/10.2200/s00999ed3v01y202003hlt046.

Fitzgerald, J. (1987). Research on revision in writing. *Review of Educational Research, 57*(4), 481–506. https://doi.org/10.2307/1170433.

Galbraith, D. (1999). Writing as a knowledge-constituting process. In M. Torrance & D. Galbraith (Eds.), *Knowing what to write: Conceptual processes in text production* (pp. 137–157). Amsterdam: Amsterdam University Press.

Galbraith, D., & Baaijen, V. M. (2019). Aligning keystrokes with cognitive processes in writing. In E. Lindgren & K. Sullivan (Eds.), *Observing writing* (pp. 306–325). Leiden: Brill.

Gladkij, A. V., & Mel'čuk, I. A. (1973). *Elemente der mathematischen Linguistik*. Salzburg: Wilhelm Fink.

Goodkind, A. (2021). *Typeshift: A user interface for visualizing the typing production process*. arXiv arXiv:2103.04222.

Halliday, M. A. K. (1976). *Halliday: System and function in language: Selected papers*. London: Oxford University Press.

Hayes, J. R. (1996). A new framework for understanding cognition and affect in writing. In C. M. Levy & S. Ransdell (Eds.), *The science of writing: Theories, methods, individual differences and applications* (pp. 1–27). Hillsdale, NJ: Lawrence Erlbaum.

Hayes, J. R. (2012). Modeling and remodeling writing. *Written Communication, 29*(3), 369–388. https://doi.org/10.1177/0741088312451260.

Hayes, J. R., & Chenoweth, N. A. (2006). Is working memory involved in the transcribing and editing of texts? *Written Communication, 23*(2), 135–149. https://doi.org/10.1177/0741088306286283.

Heidorn, G. E. (2000). 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). New York, NY: Marcel Dekker.

Honnibal, M., Montani, I., Van Landeghem, S., & Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python. https://doi.org/10.5281/zenodo.1212303.

Horning, A. (2006). Professional writers and revision. In A. Horning & A. Becker (Eds.), *Revision: History, theory, and practice, reference guides to rhetoric and composition* (pp. 117–141). West Lafayette, IN: Parlor Press.

Huang, L., & Sagae, K. (2010). Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th annual meeting of the association for computational linguistics, association for computational linguistics, Stroudsburg, PA, USA, ACL '10* (pp. 1077–1086). http://portal.acm.org/citation.cfm?id=1858791.

Jensen, K., Heidorn, G. E., Miller, L. A., & Ravin, Y. (1983). Parse fitting and prose fixing: Getting a hold on ill-formedness. *Computational Linguistics, 9*(3–4), 147–160.

Johansson, R., Wengelin, A., Johansson, V., & Holmqvist, K. (2010). Looking at the keyboard or the monitor: Relationship with text production processes. *Reading and Writing, 23*(7), 835–851. https://doi.org/10.1007/s11145-009-9189-3.

Johansson, V., Frid, J., & Wengelin, Å. (2018). Scriptlog—An experimental keystroke logging tool. In *ELN. 1st literacy summit*, conference date: 01-11-2018 Through 03-11-2018.

Kaufer, D. S., Hayes, J. R., & Flower, L. (1986). Composing written sentences. *Research in the Teaching of English, 20*(2), 121–140.

Kellogg, R. T. (1996). A model of Working Memory in writing. In C. M. Levy & S. Ransdell (Eds.), *The science of writing: Theories, methods, individual differences and applications* (pp. 57–72). Hillsdale, NJ: Lawrence Erlbaum.

Kellogg, R. T. (2001). Commentary on part II: Processing modalities and development of expertise in writing. In D. Alamargot & L. Chanquoy (Eds.), *Through the models of writing, studies in writing* (Vol. 9, pp. 219–228). Boston: Kluwer.

Kollberg, P., & Severinson Eklundh, K. (2002). Studying writers' revising patterns with S-notation analysis. KluwerIn T. Olive & C. M. Levy (Eds.), *Contemporary tools and techniques for studying writing, studies in writing* (Vol. 10, pp. 89–104). Dordrecht: Boston.

Lapata, M. (2003). Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st annual meeting on association for computational linguistics—Volume 1, Association for Computational Linguistics, USA, ACL '03* (pp. 545–552). https://doi.org/10.3115/1075096.1075165.

Leacock, C., Chodorow, M., Gamon, M., & Tetreault, J. (2010). *Automated grammatical error detection for language learners, synthesis lectures on human language technologies* (Vol. 9). San Rafael, CA: Morgan & Claypool. https://doi.org/10.2200/s00275ed1v01y201006hlt009.

Leijten, M., Macken, L., Hoste, V., Van Horenbeeck, E., & Van Waes, L. (2012). 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, association for computational linguistics, Stroudsburg, PA, USA* (pp. 1–8). http://aclanthology.org/W12-0301.

Leijten, M., Horenbeeck, E. V., & Waes, L. V. (2019). Observing writing. In E. Lindgren & K. Sullivan (Eds.), *Analysing keystroke logging data from a linguistic perspective* (pp. 71–95). Leiden: Brill. https://doi.org/10.1163/9789004392526_005.

Leijten, M., Waes, L. V., & Horenbeeck, E. V. (2015). Analyzing writing process data: A linguistic perspective. In G. Cislaru (Ed.), *Writing(s) at the crossroads: The process/product interface* (pp. 277–302). Amsterdam: John Benjamins Publishing Company. https://doi.org/10.1075/z.194.14lei.

Leijten, M., Van Waes, L., Schriver, K., & Hayes, J. R. (2014). Writing in the workplace: Constructing documents using multiple digital sources. *Journal of Writing Research, 5*(3), 285–337. https://doi.org/10.17239/jowr-2014.05.03.3.

Lindgren, E. (2005). *Writing and revising: Didactic and methodological implications of keystroke logging*. Umeå: Umeå Universitet (**PhD thesis**).

Lindgren, E., Knospe, Y., & Sullivan, K. P. (2019). Researching writing with observational logging tools from 2006 to the present. In E. Lindgren & K. Sullivan (Eds.), *Observing writing* (pp. 1–29). Brill: Leiden.

Lindgren, E., Westum, A., Outakoski, H., & Sullivan, K. P. (2019). Revising at the leading edge: Shaping ideas or clearing up noise. In E. Lindgren & K. Sullivan (Eds.), *Observing writing, studies in writing* (pp. 346–365). Leiden: Brill. https://doi.org/10.1163/9789004392526_017.

Mahlow, C. (2015a). A definition of "version" for text production data and natural language document drafts. In *Proceedings of the 3rd international workshop on (Document) changes: Modeling, detection, storage and visualization, ACM, New York, NY, USA, DChanges 2015* (pp. 27–32). https://doi.org/10.1145/2881631.2881638.

Mahlow, C. (2015b). Learning from errors: Systematic analysis of complex writing errors for improving writing technology. In N. Gala, R. Rapp, & G. Bel-Enguix (Eds.), *Language production, cognition, and the lexicon, text, speech and language technology* (Vol. 48, pp. 419–438). Berlin: Springer. https://doi.org/10.1007/978-3-319-08043-7_24.

Mahlow, C., & Dale, R. (2014). Production media: Writing as using tools in media convergent environments. In E. M. Jakobs & D. Perrin (Eds.), *Handbook of writing and text production, handbooks of applied linguistics* (Vol. 10, pp. 209–230). Berlin: De Gruyter Mouton.

Mahlow, C., & Piotrowski, M. (2008). Linguistic support for revising and editing. In: A. Gelbukh A (Ed.) *Computational linguistics and intelligent text processing: 9th international conference, CICLing 2008, Haifa, Israel, February 17–23, 2008. Lecture notes in computer science* (vol. 4919, pp. 631–642) Berlin: Springer. https://doi.org/10.1007/978-3-540-78135-6_54.

Mahlow, C., & Piotrowski, M. (2009a). LingURed: Language-aware editing functions based on NLP resources. *Proceedings of the International Multiconference on Computer Science and Information Technology, Polish Information Processing Society, 4,* 243–250.

Mahlow, C., & Piotrowski, M. (2009b). Opportunities and limits for language awareness in text editors. In R. Domeij, S.J. Kokkinakis, O. Knutsson, S. Sofkova Hashemi (Eds.) *Proceedings of the workshop on NLP for reading and writing—Resources, algorithms and tools (SLTC 2008), Tartu University Library (Estonia), NEALT Proceedings Series*,(vol. 3, pp. 14–18). http://hdl.handle.net/10062/8696.

Martin, R., Crowther, J., Knight, M., Tamborello, F., & Yang, C. L. (2010). Planning in sentence production: Evidence for the phrase as a default planning scope. *Cognition, 116,* 177–92. https://doi.org/10.1016/j.cognition.2010.04.010.

Nivre, J. (2008). Algorithms for deterministic incremental dependency parsing. *Computational Linguistics, 34*(4), 513–553. https://doi.org/10.1162/coli.07-056-r1-07-027.

Nottbusch, G. (2010). Grammatical planning, execution, and control in written sentence production. *Reading and Writing, 23*(7), 777–801. https://doi.org/10.1007/s11145-009-9188-4.

Nottbusch, G., Weingarten, R., & Sahel, S. (2007). From written word to written sentence production. *Studies in Writing*. https://doi.org/10.1163/9781849508223_004.

Perrin, D. (2002). 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). Dordrecht: Kluwer.

Perrin, D. (2006). 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). Amsterdam: Elsevier Science.

Perrin, D. (2013). *The linguistics of newswriting*. Amsterdam: John Benjamins.

Perrin, D. (2019). Progression analysis: Working with large data corpora in field research on writing. In E. Lindgren & K. Sullivan (Eds.), *Observing writing* (pp. 143–162). The Netherlands: Studies in Writing. Brill, Leiden.

Perrin, D., & Wildi, M. (2009). 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). New York, NY: Routledge.

Piotrowski, M., & Mahlow, C. (2009). Linguistic editing support. In *DocEng'09: Proceedings of the 2009 ACM symposium on document engineering* (pp. 214–217) ACM, New York, NY, USA.

Plank, B. (2016). Keystroke dynamics as signal for shallow syntactic parsing. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical Papers* (pp. 609–619).

Rehm, G., Zaczynska, K., Schneider, J.M., Ostendorff, M., Bourgonje, P., Berger, M., Rauenbusch, J., Schmidt, A., & Wild, M. (2020). Towards discourse parsing-inspired semantic storytelling. In A. Paschke, C. Neudecker, G. Rehm, J. A. Qundus, & L. Pintscher (Eds.) *Proceedings of QURATOR 2020—The conference for intelligent content solutions. Conference on digital curation technologies (QURATOR-2020), Berlin, Germany, CEUR Workshop Proceedings*.

Roark, B. (2001). Probabilistic top-down parsing and language modeling. *Computational Linguistics, 27*(2), 249–276. https://doi.org/10.1162/089120101750300526.

Rosenqvist, S. (2015). *Developing pause thresholds for keystroke logging analysis*. Umeå Universitet Bachelor's thesis.

Schneier, J. (2020). Digital articulation: Examining text-based linguistic performances in mobile communication through keystroke-logging analysis. *Frontiers in Artificial Intelligence, 3,* 110.

Sommers, N. (1980). Revision strategies of student writers and experienced adult writers. *College Composition and Communication, 31*(4), 378–388. https://doi.org/10.2307/356588.

Stachowiak, H. (1973). *Allgemeine Modelltheorie*. New York: Springer.

Strömqvist, S., & Karlsson, H. (2002). *Scriptlog for windows: User's manual*. Lund, Sweden: University of Lund, Department of Linguistics.

Torrance, M. (2015). Understanding planning in text production. In C. A. MacArthur, S. Graham, & J. Fitzgerald (Eds.), *Handbook of writing research* (pp. 1682–1690). New York: Guildford Press.

Torrance, M., & Nottbusch, G. (2012). Written production of single words and simple sentences. In V. W. Berninger (Ed.), *Past, present, and future contributions of cognitive writing research to cognitive psychology* (pp. 403–422). London: Psychology Press.

Van De Vanter, M. L. (1995). *Practical language-based editing for software engineers. Lecture Notes in Computer Science. Software engineering and human–computer interaction* (Vol. 896, pp. 251–267). New York: Springer. https://doi.org/10.1007/bfb0035821.

Weder, M. (2010). *Keystroke-Logging und Stimulated-Recall in der Orthographie-Forschung. Bulletin suisse de linguistique appliquée* (pp. 85–104).

Weingarten, R., Nottbusch, G., & Will, U. (2004). Morphemes, syllables and graphemes in written word production. In T. Pechmann & C. Habel (Eds.), *Multidisciplinary approaches to language production* (pp. 529–572). Berlin: De Gruyter Mouton. https://doi.org/10.1515/9783110894028.529.

Weizenbaum, J. (1976). *Computer power and human reason*. New York, NY: W. H. Freeman & Co.

Wirén, M. (1989). Interactive incremental chart parsing. In *Proceedings of the fourth conference of the European chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Morristown, NJ, USA* (pp. 241–248). https://doi.org/10.3115/976815.976848.