

DHQ: Digital Humanities Quarterly

Preview

2022

Volume 16 Number 1

Elli Bleeker <elli_dot_bleeker_at_huygens_dot_know_dot_nl>, Huygens Institute for Dutch History and Culture
 Bram Buitendijk <bram_dot_buitendijk_at_di_dot_huc_dot_know_dot_nl>, Royal Netherlands Academy of Arts and Sciences
 - Humanities Cluster, Digital Infrastructure department.

Ronald Haentjens Dekker <ronald_dot_dekker_at_di_dot_huc_dot_know_dot_nl>, Huygens Institute for Dutch History and Culture

Vincent Neyt <vincent_dot_neyt_at_uantwerpen_dot_be>, University of Antwerp

Dirk Van Hulle <dirk_dot_vanhulle_at_ell_dot_ox_dot_ac_dot_uk>, University of Oxford

Abstract

The article describes research into the automatic comparison of texts with revisions. The authors argue that in-text variation can best be modelled as nonlinear text, and that a collation tool needs to treat in-text variation differently from the way linear text is treated. They describe in detail how the modelling choices they made influence the development of HyperCollate, a collation software that is able to process TEI-XML transcriptions of texts with variation. Consequently, HyperCollate produces a more refined collation output that corresponds with a human interpretation of textual variance.

1. Introduction

The act of comparing textual versions has always been central to textual criticism and manuscript research. The importance of this type of collation has only increased in the digital age. Most scholarly editors no longer produce collations by hand; they use the output generated by software, either as is, or as the starting point for manual correction. Still, in more complex cases such as manuscripts with revisions – i.e., textual variation within one text version – it remains difficult to have the collation software produce a result that corresponds with what a scholarly editor would produce. The reason for this is, simply put, that the collation software treats the input as linear text, whereas text with revisions is not completely linear. As a consequence, scholarly editors still need to do quite some manual work, such as preparing the input texts or adjusting the collation output. What is more, they lose information about the multilayered nature of the document and the physical traces of the writing process or transmission history.

1

We therefore set out to create collation software that is able to compare texts with revisions. When working on this challenge, we soon realized that conveying our human interpretation of text with revisions to the computer is not merely a technical matter. It is, in fact, deeply rooted in philological questions: what do we understand by text with revisions? What type of revisions can we discern? And how do we expect the collation software to deal with them? Designing and developing a collation tool that understands in-text variation in a way similar to how scholarly editors understand it requires us to formalize our understanding of in-text variation. We argue that in-text variation can best be modelled as nonlinear text, and that a collation tool needs to treat this nonlinear text differently from the way linear text is treated. This will produce a more refined collation output that corresponds with a human interpretation of the textual variance.

2

For the purposes of this article, we focus on in-text variation resulting from authorial revisions on modern manuscripts; revisions by multiple textual agents are not within the scope of this article. As a test-corpus we took small examples from a modern manuscript of Samuel Beckett featuring numerous deletions and additions that Beckett made at various stages in the writing process.^[1] Evidently, the feature of in-text variation is not unique to modern manuscripts: medieval and older manuscripts often contain indistinguishable layers of in-text revisions as well. When in-text revisions are made with the same writing tool they cannot be easily divided into separate revision campaigns. The challenges of automatically collating such multi-layered texts are, then, twofold: first, to adequately process a textual version that contains one or more revisions, and second, to return a collation output that corresponds with what a scholarly editor would intuitively produce. We will discuss our approach to these challenges in the context of HyperCollate, an automatic collation tool developed to take in-text revisions into account when calculating the best match between two textual versions.

3

Section 2 provides a theoretical background for the concepts and practices under discussion: versions, witnesses, in-text revision, and (non)linear text. It also provides a brief overview of the existing approaches to collating texts with in-text variation. Section 3 presents the collation tool HyperCollate. We describe its underlying data model – a hypergraph

4

for textual variation – and how this data model differs from XML tree structures and regular variant graphs. A hypergraph for textual variation offers more flexibility in storing and processing information that is relevant to the study of textual variance. Subsequently, we single out two instances of in-text variation that occur frequently in manuscript texts – single deletions or additions and substitutions – and we describe for each type of variation what we, as human scholars, expect of the collation output (section 3.2). How do we want the collation tool to handle immediate revisions in a textual version? What do we expect to see in the output? The answer to such questions, i.e., our expectations, inform the rules for the collation program. HyperCollate's performance with each type of variation is discussed in section 3.3. The evaluation (section 3.4) includes the question of visualization, as existing visualizations could turn out to be inadequate in displaying the additional information in an insightful way. Section 4, finally, discusses a number of areas in need of further research.

As the development and use of collation tools requires us to clearly articulate what we understand by textual variation and what we expect from the collation, automated collation tools can provide us with a heightened awareness of, and confront us with our assumptions about, the notion of *text*. The workflow of a scholarly editor using an automated collation tool consists of seemingly straightforward steps (transcription ⇒ collation ⇒ visualization) but each step implies decision-making at several stages and substages (selection, tagging, tokenization, normalization, a definition of a match, etc.). The decisions made at one stage influence the outcome of a subsequent stage. Recognizing and understanding this heuristic process will prompt us to rethink our theories of text (cf. Ciula and Eide 2014, 36) and how these theories correspond to the workings of a text analysis tool.

By demonstrating, first, how we made our human definition of in-text variation computer-understandable and, subsequently, how this definition influences the operations as well as the results of a collation tool, we hope to provide our readers with a deeper understanding of the process of modeling textual variation. We intend to show how the development of a collation program has both a philological and a computational perspective. Accordingly, our study reflects a fruitful collaboration between textual scholars and software architects: their respective knowledge is essential in order to successfully store, process, and analyse text. It thus proves a worthy partnership for manuscript research and modelling textual variance.

2. Theoretical and Methodological Background

In what follows we give an overview of the discussion around textual versions and witnesses (section 2.1), of the challenges of transcribing and modeling in-text variation (section 2.2 and 2.3), and of the current approaches to collating texts with internal revisions (section 2.4).

2.1. Text, Work, Versions, Witnesses, and In-text Revision

A few working definitions of the terms *document*, *version*, *witness*, *work*, and *text* are in order. Following Peter Shillingsburg, we define a *document* as the “physical vessel” that carries information [Shillingsburg 1996, 174]. A *version* has “no substantial existence” as Shillingsburg puts it [Shillingsburg 1996, 44]. As a theoretical construct, it implies that there are at least two textual representations of the work. A *witness* is a “written document that witnesses a text's genesis” according to Almuth Grésillon [Grésillon 1994, 246].^[2] A witness is therefore a physical object. Whereas the term *document* considers the object in and of itself, the term *witness* considers it in relation to a work. Different documents that relate to the same work are witnesses of it. According to Siegfried Scheibe, “Textual versions are achieved or unachieved elaborations of the text that diverge from one another. They are related through textual identity and distinct through variation.” [Scheibe 1995, 207]. In principle, this implies that a single revision suffices to speak of two versions, or as Hans Zeller noted: “In the most extreme case a version is constituted by a single variant. A holograph with one alteration which does not simply correct an error thus represents two versions of the text” [Zeller 1975, 236]. The notion of the *work* is used in the sense of the “experience implied by the authoritative versions of literary writing” [Shillingsburg 1996, 176], a non-material concept that “serves as a minimal denominator to identify its remaining physical manifestations” [Bordalejo 2013, 71]. In the context of this article, the term *text* will be employed to indicate not so much “the text of the work,” but “the text of the document,” that is, the sequence of marks present in one particular document [Bordalejo 2013, 67].

In textual scholarship, the notion of *version*, *textual version* or *Textfassung* usually implies the entirety of a work, as in Peter Shillingsburg's definition: “A version is one specific form of the work – the one the author intended at some particular moment in time” [Shillingsburg 1996, 44]. The problem, however, is that this “moment in time” is hard to define. If the work is a long novel, it takes a while for an author to revise the entire manuscript or to proofread the page proofs. The revision may consist of more than one phase, or more than one revision campaign (*campagne de révision*). There may be only two versions of the opening chapter, but a dozen versions of the closing chapter. Moreover, the unit

of revision may vary considerably. Some writers' preferred writing unit is a chapter, others tend to work and think in terms of paragraphs.

This reality makes it difficult to establish a general rule on which we can base the collation algorithm. For the purposes of this article, we work with the sentence as a relatively small unit of revision. Our examples are based on the manuscripts of Samuel Beckett and the transcriptions in the Beckett Digital Manuscript Project (BDMP).^[3] These documents constitute what is usually referred to as “modern” manuscripts, to distinguish these autographs from ancient and medieval manuscripts, many of which were produced by scribes. Indeed the majority of medieval manuscripts are copies and not autographs, but that does not mean that all medieval manuscripts are copies. Medieval autographs do exist, and it is perfectly possible to apply automatic collation to them. Instead of using the term *modern manuscripts* we will therefore speak of “manuscripts with revisions” or even more generally of “texts with revisions.” *In-text* implies that we focus on textual variation within one version of a text.

10

2.2. Transcribing Linearity and Nonlinearity

As mentioned in the introduction, studying manuscripts with revisions is intrinsically linked to the act of linearizing text. Even when we read a text, we mentally arrange the characters in a linear order; vertically or horizontally, from left to right or vice versa, depending on the language. This perhaps subconscious act of linearization is more striking when we make a transcription of a complex text with revisions. Since the goal of a transcription is, in the first place, to render a text more readable, linearizing the text on the document is desirable – but not without cost. Texts with revisions contain traces of a writing process over time; so when words and characters are placed in a linear order, the manuscript page loses that diachronic aspect. The desire to preserve the nonlinear nature of the text coupled with the need to make it readable (and processable) confronts textual scholars with a dilemma. Accordingly, linearization has been the topic of long debates in genetic criticism and textual scholarship [Grésillon 1994] [Lebrave 1992] [Van Hulle 1999] [Van Hulle 2004] [Ferrer 2014]. The “flattening” or “de-manuscripting” of a text is generally accepted as a necessary evil, although it is lamented that the most typical feature of a written draft – its nonlinearity – is reduced to a straightforward sequence of text.

11

In print, nonlinear text can be indicated with certain diacritical marks or forms of layout; in digital transcriptions the revisions can be captured with markup elements and visualized with HTML and/or CSS.^[4] The TEI Guidelines suggest two ways of expressing an instance of nonlinear text: substitutions can be recorded (1) by means of a `<subst>` element grouping one or more `` and `<add>` elements,^[5] or (2) by means of an `<app>` element containing two or more `<rdg>` elements.^[6]

12

In both cases, a human encoder will think of these TEI elements as indicating the momentary suspension of the linear sequence of text characters. The text stream runs from left to right, diverges at the points where variation occurs (indicated with an open tag `<subst>` or `<app>`) and converges when the variation ends (indicated with a closing tag `</subst>` or `</app>`). We can say that there are multiple coexisting, simultaneous paths through the text. The start of each path is marked with a ``, an `<add>`, or a `<rdg>` element; the end of each path with the corresponding ``, `</add>`, or `</rdg>`. By “coexisting” we mean that the textual content of the ``, `<add>`, or `<rdg>` is in the same location in the sentence. This corresponds with the TEI Guidelines' remark that “Since the purpose of this [the `<subst>`] element is solely to group its child elements together, the order in which they are presented is not significant.”^[7]

13

As described in section 2.4, a current approach to the automated collation of in-text revisions is to flatten the TEI-XML transcription to a stream of plain text characters, thereby ignoring the TEI-XML markup elements that indicate nonlinear text. In order to fully understand the challenges that arise when we *do* wish to avoid flattening and instead take a text's nonlinear features into account, we first need to take a closer look at how in-text revisions can be modelled in TEI-XML.

14

2.3. Modeling Nonlinearity

A TEI-XML transcription of a straightforward text consists typically of “fully ordered” data. For textual scholars, the value and meaning of this data are primarily in the textual content, not in the TEI-XML elements. As a consequence, the order of the TEI-XML elements is meaningful. In the TEI-XML expert below, changing the order of the two `<s>` elements with their textual content would alter the meaning of the text:

15

`<p>`

```

<s>The sun shone, having no alternative, on the nothing new.</s>
<s>Murphy sat out of it, as though he were free, in a mew in West Brompton.</s>
</p>

```

We have argued in the previous section that the textual data of a TEI-XML transcription of a manuscript with in-text revisions can be understood as an ordered stream of text characters, which splits into two or more paths when textual variation occurs. Or, put metaphorically by Michael Sperberg-McQueen, this type of text forms “curves that intersect, run together for a while, and then split apart again, like the channels in a river delta” [Sperberg-McQueen 1989]. By way of illustration, let’s take the following TEI-XML example:

16

```

<p>
  <s>The
    <subst>
      <del>quick</del>
      <add>brown</add>
    </subst> fox.
  </s>
</p>

```

The stream of textual data starts by being fully ordered with “The.” At the point in the text where variation occurs (marked with `<subst>`), the stream splits into two paths that run parallel to one another. The textual data *within* the two `` and `<add>` paths is again fully ordered as each path contains a linear stream of text characters. From a purely informational perspective, the deleted word “quick” is on the same level in the XML tree as the added word “brown”: they are both the textual content of the child elements of the `<subst>` element and they occur in the same place in the text. In other words, their position vis-à-vis their parent element is the same. At the end of the variation (marked with `</subst>`) the two paths converge again into one stream.

17

The order in which TEI-XML elements are placed also carries philological or semantic value. In the field of digital scholarly editing it is generally acknowledged that markup represents valuable editorial knowledge and that the tags reflect an interpretation of the inscriptions on the source document. Compare for instance

18

```

<s>The
  <subst>
    <del>quick</del>
    <add>brown</add>
  </subst> fox.
</s>

```

and

```

<s>The
  <del>quick</del>
  <add>brown</add> fox.
</s>

```

In the first case, the encoder groups together the deletion and the addition as part of one and the same revision. The elements `` and `<add>` are in the same hierarchical location in the XML tree (again, both are children of the `<subst>` element).^[8] In the second case, the deletion and the addition are interpreted as separate interventions. The `` element precedes the `<add>` element, which denotes a chronological order: first the word “quick” was deleted, then the word “brown” was added. In order for a collation tool to produce an output that corresponds with this interpretation of in-text revision, it should both recognize the start of parallel paths marked with `<subst>` or

<app><rdg>, and take into account the chronological order implied by the order of the and <add> tags without a <subst> or <app> parent.

2.4. Current Approaches to Collating In-text Revisions

This section presents a concise discussion of the existing approaches to collating in-text revisions. We leave aside the manual collation method of TEI Parallel Segmentation and others and concentrate on the methods of automated collation.^[9] As mentioned above, current automated collation tools are generally text-centered and do not handle TEI-XML well: they ignore all tags and attributes, or collate the elements as plain text. The nonlinear aspect of texts with revisions as described above, is ignored. If a scholar would want to automatically compare XML files, they could opt for (1) passing along information marked as relevant to the study of in-text variation (with CollateX, collation software; see section 2.4.1), or (2) creating plain text files for each level of in-text variation (for instance with MVD, a storage format for text; see section 2.4.2). As far as we know, both approaches are implemented by at least one edition project.

19

2.4.1. Passing Along Relevant Markup Elements

The JSON input format of the collation tool CollateX allows for extra properties to be added to words [CollateX]. A number of projects make use of the JSON input format to “pass along” additional information about relevant markup elements through the collation pipeline. In a pre-processing step, the text is tokenized and transformed into JSON word tokens. Editors make a selection of elements that they wish to attach to the JSON token as a *t* property (“t” for token). The collation is executed on the value of the token’s *n* property (“n” for normalized), but the *t* property with the tag(s) is included in the JSON alignment table output and can be processed in the table’s visualization. This approach is used, among others, by the aforementioned BDMP; the online critical edition of the Primary Chronicles, or *Povest’ vremennyx let*, created by David J. Birnbaum; and the work on the *Declamations* of Calpurnius Flaccus done by Elisa Nury [Birnbaum 2015] [Nury 2018, 7.1]. The advantages of this approach are, first, that a witness with in-text variation can be collated as one text and, secondly, that the method approximates the goal of having nonlinear revisions marked as such in both in- and output. The main disadvantage is that the collation algorithm still treats the input witness as linear text: the witnesses are collated as such and any information about revision campaigns is ignored during the collation process.

20

2.4.2. Creating Separate Plain Text Files For Each Level of Variation

Another approach is to transcribe a manuscript with in-text revisions as two or more separate plain text versions and then collate these temporary versions against each other. The advantage of this method is that it succeeds in aligning the two parts of a substitution vertically instead of horizontally (which the previous approach is forced to do), but the drawback is that the deletions and additions are spread over two or more (sub)versions. What is more, the revisions on a manuscript do not always lend themselves to a clean separation into different layers.

21

The approach is presented by Desmond Schmidt’s Multi-Version Document system, as integrated into Ecdosis, “A CMS for scholarly editors.”^[10] In a 2019 article, Schmidt describes in detail how this layered approach also allows for the inclusion of in-text variation [Schmidt 2019, §60]. At the transcription phase, a manuscript text is divided into “levels” by the scholarly editor. All changes to the “baseline text” of a document (level 1) are grouped as being on level 2. Subsequent changes to level 2 are grouped as level 3. For each of these levels a separate text is created, a “layer:” “We do not claim that a layer is a text the author ever completed; it is not a *version* as such,” explains Schmidt, “It is simply a compact way to capture local changes.”^[11] Each of these artificial layers is fed into Ecdosis as plain text, with a few additional features encoded in stand-off markup. The Ecdosis approach is implemented in the Charles Harpur Critical Archive. For example, the text of Harpur’s poem “To the Comet” is divided into three layers. The document identifier “english/harpur/poems/h595f/layer-2” points to the second layer of the poem’s text. Layer 2 is followed by “h595f/layer-final,” in which corrections on the third level are carried out on the text of layer 2.^[12] The scholarly editors of the Charles Harpur Critical Archive see no issue in this “mechanical” grouping of in-text variation:

22

Assignment to levels is mechanical and describes a purely local succession of changes, which can almost always be determined. Authors leave many clues as to the succession of variants: position (above the line, then below, in the margin, on another page), the carrying over of changed text between levels, sense and crossing-out. Where the local temporal succession can’t be established this is usually because the corrections themselves can’t be read. ((*Ibid.*))

The versions and layers in plain text are merged into one Multi-Version Document. It stores the text shared by each version and layer only once. Note that the comparison algorithm works at the character level, not at word level. In the table visualization, therefore, words may be cut up (see figure 1).

h595f/layer-1	It is necessary to note thi	s, lest the youthful reader should suppose that the Poet's imagination was making "t
h595f/layer-2	isphere	But it is necessary to note these facts, lest the youthful reader should suppose that the Poet's imagination was making "t
h595f	isphere	But it is necessary to note these facts, lest the youthful reader should suppose that the Poet's imagination was making "t
h595g/layer-1	nisphere.	But it is necessary to note these facts, lest the youthful reader should suppose that the Poet's imagination was making bc
h595g	nisphere.	But it is necessary to note these facts, lest the youthful reader should suppose that the Poet's imagination was making bc

Figure 1. Table visualization of the collation output of the artificial layers of *The Comet* by Charles Harpur. Text characters added with respect to the base layer (not shown in this figure) are in blue.

2.4.3. Diff Algorithms to Compare XML Documents

Evidently, XML is a widely used standard for (online) publishing and data-exchange, and there is a general need for comparing and versioning XML documents. Accordingly, there exists a wide variety of *diff* algorithms or *deltas* to find the differences between two XML documents, each with its own strengths and focus [Peters 2005]. Because most XML documents are data-centric and contain record-based data, deltas typically consider XML documents as hierarchically structured trees and compare them as such. Considering our type of material – text-centric XML documents in which the left-to-right order of the siblings is significant – we specifically focused on tools that can compare ordered XML trees.

We found a number of useful tools and algorithms, like the commercial tools Microsoft's XML Diff and Patch or the XML compare tool of DeltaXML, that can compare both ordered and unordered XML trees.^[13] Such tools can come in handy during the making of a digital edition, for instance when scholarly editors wish to compare and align TEI-XML transcriptions of the same text that were made by two different scholars of the same project. Furthermore, there exist XML editors with a built-in function to compare XML documents. Oxygen XML Editor, for example, allows its users to compare two XML documents on the level of lines, words, or characters, or to ignore attributes or namespaces.^[14]

There is, however, an important difference between current XML comparison tools and algorithms, and how we designed HyperCollate. Existing tools compare XML documents and focus on the parent nodes, whereas HyperCollate *interprets* XML documents in order to better compare the versions of a text. It does not ignore the structure of the XML tree, but – in line with the research objectives of scholarly editors – it focuses on comparing text and is only concerned with the TEI-XML tags that determine the flow of a text with revisions. Given this fundamental difference in premise, an extensive analysis of the quality and use of diff algorithms and tools is beyond the scope of this article.^[15]

3. Automated Collation of Nonlinear Text

3.1. Nonlinearity and Variant Graphs

As the previous section showed, automated collation software usually does not take in-text revisions into account. Yet we find that texts with revisions have a complex richness that deserves to be treated with attention to detail, in transcription as well as in collation, so as to produce a more accurate and detailed record of the textual variance. The collation result of HyperCollate needed to correspond with the scholarly editor's interpretation of the in-text variance, which meant that the tool should be able to recognize and respect the nonlinear nature of the input text as it was expressed in TEI-XML markup.

The first step we took in enabling a collation tool to recognize in-text variation in a witness was establishing formal (i.e., exhaustive and unambiguous) rules for the way in which the program needed to process it. As described in section 2.2, we regard a revision as a temporary interruption in the linear text stream, so we wanted HyperCollate to be able to treat it as such. In order for this to work, the collation tool would have to recognize and interpret the markup elements indicating the start and end of nonlinearity in the TEI-XML input transcription: `<app>` and `<subst>`. Secondly, we regard the elements ``, `<add>`, or `<rdg>` as indicators of the start of a separate path through the text – one "branch" of the text stream – and we would want the collation program to select the best match from these branches.

A fitting model of nonlinear text is presented by the variant graph data structure. The variant graph consists of a set of nodes that represent textual objects, and edges that connect one node to another node in the graph. The graphs are acyclic, which means that they are read from left to right without "looping" back. As Tara Andrews and Caroline Macé write, a variant graph is an elegant way to represent nonlinear text: "one may imagine a text running from beginning to end with a number of points of divergence, where each witness takes a single path through these divergences" [Andrews 2013, 506]. Variant graphs are relatively well-known in the field of automated collation and occur both as internal data model [Schmidt and Colomb 2009] [CollateX] and for visualization purposes (e.g., the stemma graph or the variant graph, [Andrews 2013, 507, 510–1]). As Elisa Nury writes, the first use of a variant graph to express textual

variation can be traced to E.C. Colwell and E.W. Tune in 1964, although it was with the 2009 article of Desmond Schmidt and Robert Colomb that the variant graph gained a foothold in the field of textual scholarship [Schmidt and Colomb 2009] (See [Nury 2018, 73–7]). Their variant graph has the text and sigla placed on the edges of the graph (figure 2). Common text is merged, and variant text results in separate branches.

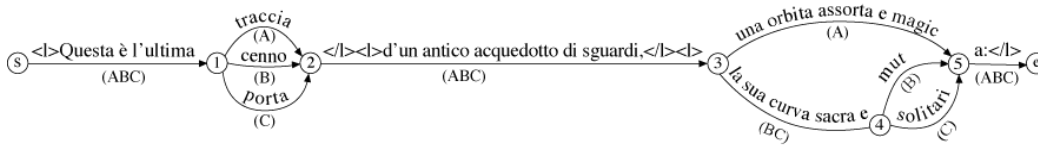


Figure 2. Representation of the variant graph data model of Schmidt and Colomb (source: CollateX documentation, section “The Data Model: Variant Graphs,” on <https://collatex.net/doc/>).

CollateX adopted the variant graph, but modified it to place the text in the nodes of the graph, and the sigla on the edges (see figure 3). This ensures not only a better readability, but also improves processing as the edges have only one label (one or more sigla). 29

There is another, more subtle yet important distinction between the variant graph as proposed by Schmidt and Colomb, and the variant graph as implemented by CollateX. The data structure of the variant graph in figure 3 appears simpler than it is. At first glance, the textual content of the first node (“The”) appears to be shared by witness 1 (W1) and witness 2 (W2). In fact, each node in the CollateX variant graph represents a *set of text tokens* that originate from one or more versions; the tokens in a set are considered equal by the CollateX collation algorithm. The main point being that the tokens are not modeled on textual content, as is the case with the variant graph of Schmidt and Colomb. In CollateX’s variant graph model, arbitrary tokens can be compared and their correspondences, differences, and order can be represented in a variant graph structure [CollateX]. This seemingly small feature paves the way for HyperCollate’s approach to collation. 30

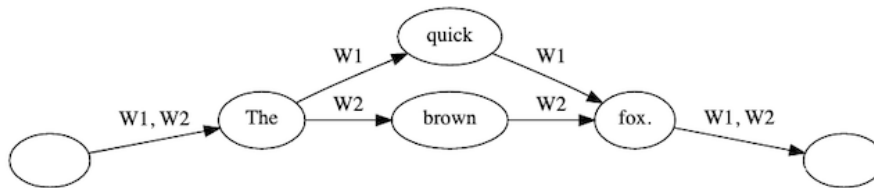


Figure 3. Variant graph of CollateX: witness sigils on the edges and the text in the nodes. Text that is considered as a match shares a node.

HyperCollate uses a *hypergraph for textual variation*. This data structure is more flexible than trees, and it differs from regular variant graphs in that it has *hyperedges* which can connect multiple nodes to one another. As a result, the nodes of a hypergraph can be traversed in more than one order. Finally, the hypergraph data model of HyperCollate contains text nodes as well as markup nodes. This means that the markup node of, say, a `` element can be connected to multiple text nodes which do not immediately have to follow one another. As a result, it is possible to express multiple, overlapping or nonlinear dimensions of a text in markup.^[16] Moreover, the hypergraph also enables the representation of individual witnesses with in-text variance as a hypergraph (see figure 4). 31

HyperCollate is trained on the two ways of encoding in-text revisions as proposed by the TEI and discussed in section 2.2: it acts whenever it encounters either a `<subst>`, an `<app>`, or their children ``, `<add>`, or `<rdg>`. These TEI-XML elements are interpreted as instances of nonlinear text, with the opening tags marking a break in the linear sequence of the text characters, and the closing tags identifying the return to linearity. Let’s take for Witness 1 the same example of a simple TEI-XML transcription we used above: 32

```
<s>The
  <subst>
    <del>quick</del>
    <add>brown</add>
  </subst> fox.
</s>
```

Figure 4 shows the witness hypergraph of this example. Note that the hypergraph contains not only information about the text of the witness, but also its markup. In this visualization, the markup information is shown as an XPath. We see, for instance, that the XPath of the text token “quick” is `/s/subst/del`.

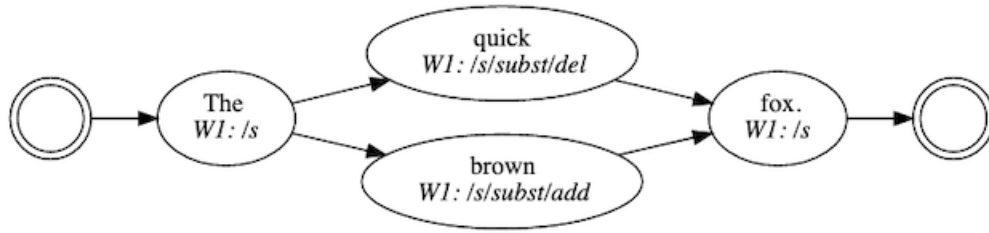


Figure 4. Visualization of the hypergraph of Witness 1 (W1) as stored in HyperCollate.

Another way to visualize the hypergraph of Witness 1 is shown in figure 5, below. Here, the tree of markup nodes is projected onto the variant graph of the text. The text still reads from left to right, with the stream of text temporarily separating into two concurrent branches. The markup nodes that are associated with each text node in the XML tree are visualized as colored nodes.

33

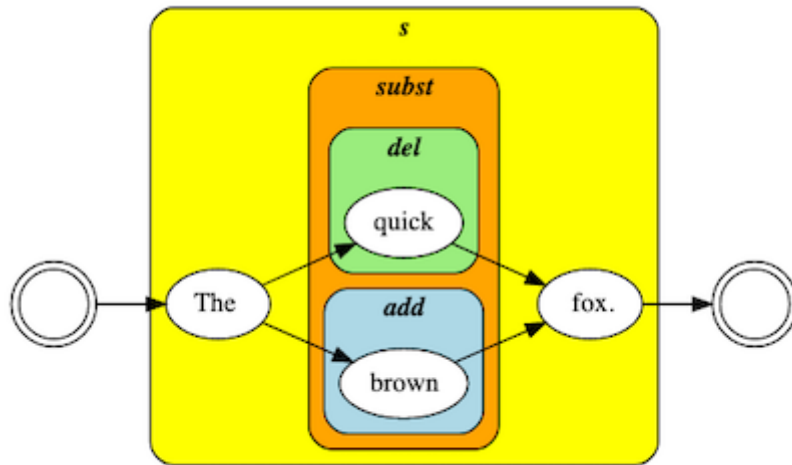


Figure 5. Different visualization of the hypergraphs of Witness 1 as stored by HyperCollate, with the markup information visualized in colors.

3.2. HyperCollate

Having illustrated how HyperCollate stores the information in a TEI-XML transcription, we can now move on to demonstrate how this information is processed and collated.

34

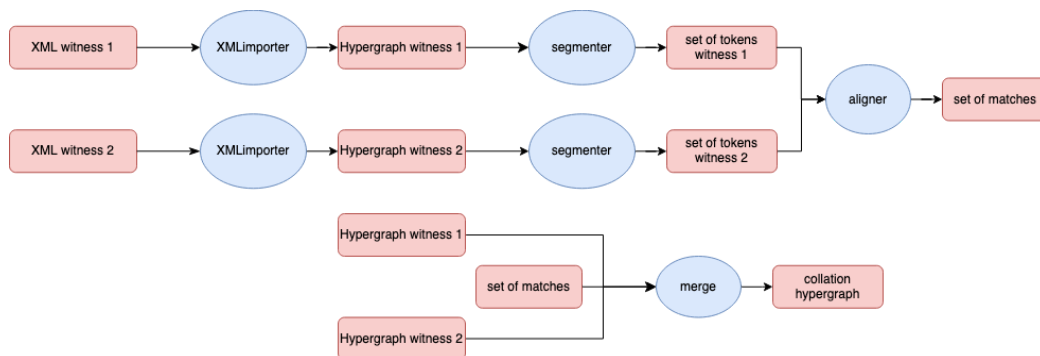


Figure 6. Schematic workflow of HyperCollate.

Figure 6 represents the pipeline of HyperCollate. First, the TEI-XML transcribed texts (here “XML witness 1” and “XML witness 2”) are transformed into individual variant hypergraphs. These variant hypergraphs are subsequently segmented and the resulting tokens are aligned. The alignment is carried out on the level of the text; the markup

35

elements `<app>` and `<subst>` are recognized as occurrences of nonlinearity. Subsequently, the resulting set of matches are merged with the two individual hypergraph witnesses. The result of the collation is a collation hypergraph in which all text nodes that are not aligned are unique to one of the two hypergraph witnesses, and all the text nodes that are aligned are reduced to one node. There are labels on the edges indicating which node is part of which hypergraph. For every witness, then, there is always a fully connected path through the hypergraph from the start text node to the end text node, following the sigils on the edges. The collation hypergraph can be visualized or exported in alignment table format, SVG, PNG, or dot format. If there are more than two witnesses, the result of that first collation can be used as the basis of a new collation, following the method of progressive alignment.^[17]

HyperCollate does not regard the input TEI-XML files as plain text files (which would result in the aforementioned “flattening” of the input text): instead, each input witness is transformed into a hypergraph and these individual witness hypergraphs are subsequently collated against each other. In short, HyperCollate can compare TEI-XML files with in-text revisions without having to convert them to a plain text character string. During the alignment, HyperCollate looks at all branches within each witness hypergraph. If the textual content of one branch (the start of which is indicated by a ``, `<add>`, or an `<rdg>` open tag) has a match in the other witness, the text is aligned; if not, the text remains in a separate branch in the hypergraph.

36

Furthermore, the tag names (``, `<add>`, `<rdg>`, `<subst>`, and `<app>`) are not only interpreted but preserved as well, in order to enable further querying, transformation or visualization of the collation result. The visualization of the information-rich collation hypergraph opens up new possibilities and new questions. For example, if we visualize the collation output as a variant graph, should we distinguish the paths that represent variation within one witness from the paths constituted by variation between different witnesses, and if so: how?

37

As shown in section 3.3.3, figure 20, we have currently opted for placing extra marks on the edges of the visualization of the collation graph, right after the witness sigil: a minus [-] for an in-text deletion and a plus [+] for an in-text addition. In case of an in-text substitution within a substitution, this will lead to a sequence of multiple [+][-] which may not lead to a clear visualization. Of course, we can say that at this point the source manuscript text also presents a complex case of revision and this complexity is merely reflected in the collation output. Nevertheless, visualizing the collation produced by HyperCollate remains an interesting challenge (see also [Bleeker et al. 2019]).

38

3.2.1 Deletions and Additions

Consider the example in figure 7, transcribed first with the TEI `/<add>` method, and then with the TEI `<app>/<rdg>` method.

39

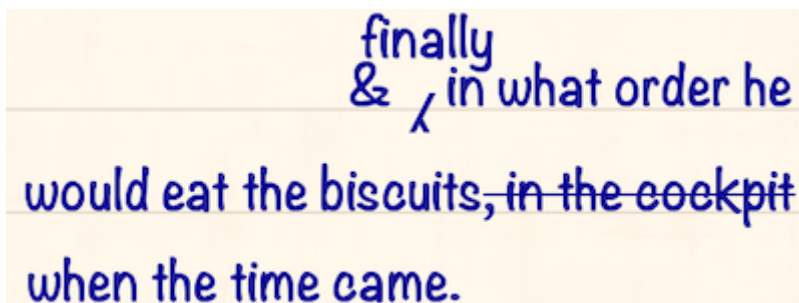


Figure 7. Reconstruction of a fragment from Samuel Beckett: Murphy, Draft notebook, University of Reading, Cat. Nr. 5517/3, page 20r.

```
<xml>
  <s>& <add>finally</add> in what order he would eat the biscuits<del>, in the
  cockpit</del> when the time came.
  </s>
</xml>
```

Example 1. Transcription using the TEI `del/add` method.

```

<xml>
  <s>&
    <app>
      <rdg varSeq="1"/>
      <rdg varSeq="2">finally</rdg>
    </app> in what order he would eat the biscuits
    <app>
      <rdg varSeq="1"> , in the cockpit</rdg>
      <rdg varSeq="2"/>
    </app> when the time came.
  </s>
</xml>

```

Example 2. Transcription using the TEI app/rdg method.

Human expectations and algorithmic rules

With this TEI-XML transcription as one of the witnesses in the input for HyperCollate, we expect that the collation output retains the information about which words in a witness were in-text additions or deletions. Additions or deletions demarcate the beginning of a split in HyperCollate's token stream. From that point on there are two substreams (the "branches"): one with the `` or `<add>` and their textual content, and one without. The substreams merge after the closing `` or `</add>` element.

40

Individual witness hypergraphs containing single addition(s) and deletion(s)

Figure 8 shows the individual variant graph of the witness with the `/<add>` tagging, and figure 9 using the app/rdg method of transcribing.

41

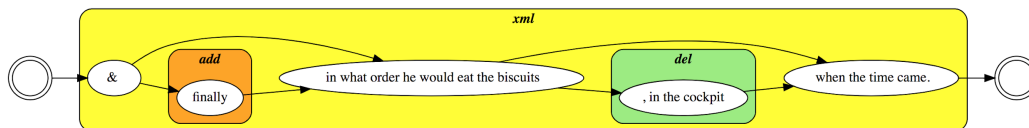


Figure 8. Internal graph model of the del/add tagging of the example.

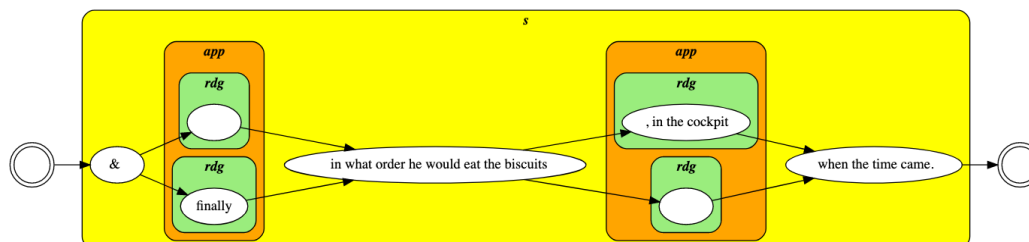


Figure 9. Internal graph model of the app/rdg tagging of the example.

3.2.2 Instant Revisions

Instant revisions or *currente calamo* changes are revisions made within the initial writing of a sentence, with the change occurring on the same line, made in one and the same writing session. They pose a particular challenge for transcription, because they cannot be encoded as a substitution: the demarcation of the added text is problematic. Would all of the text following an instant deletion be an addition? The TEI Guidelines propose to add the attribute `@instant` with the value "true" to the `` element, to distinguish between a regular correction.

42

In the following example, Beckett wrote "threw up his" and then crossed the three words out to continue the sentence with "gave such a jerk at Russell's arm that that poor little man was nearly pulled off his feet."

43

& now threw up his gave such a jerk
at Russell's arm that...

Figure 10. Reconstruction of a fragment from Samuel Beckett: Murphy, Draft notebook, University of Reading, Cat. Nr. 5517/2, page 16r

In this case it not make sense to tag “gave such a jerk” as an addition, not would it make sense to tag the rest of the sentence as such. 44

Hence we argue that an instant revision does *not* constitute a case of nonlinear text. In contrast to regular in-text corrections and substitutions, there is no other way of reading the text: the instant revision is part of the same writing campaign as the text surrounding it. 45

```
<s>& now <del instant="true">threw up his</del> gave such a jerk</s>
```

Example 3. Transcription using the TEI del/add method.

Human expectations and algorithmic rules

Following our interpretation of an instant revision, we would *not* want HyperCollate to treat it as an instance of nonlinear text. An instant deletion should be considered to be part of the same path as the text surrounding it. We do want the information about the instant revision to be present in the output, so that we can visualize the deletion appropriately. 46

Individual witness hypergraphs containing an instant revision

As the instant revision, encoded with `<del instant="true">`, does not trigger HyperCollate in the same way a regular `` does, the individual witness hypergraph has only one path through the text. The information about the deletion is retained (see figure 11). 47

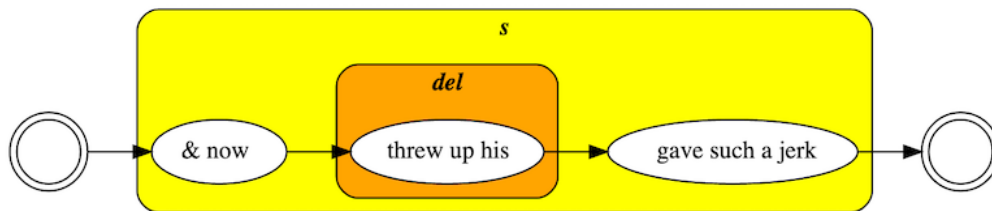
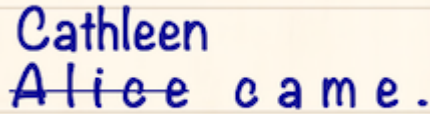


Figure 11. Internal graph model of an instant revision.

3.2.3 Substitutions

Deletions followed by additions that are (semantically) related can be grouped by means of a `<subst>` (substitution) element, or as two readings in an `<app>` element. This can be illustrated with a simple example from Beckett (see figure 12). 48



Cathleen
~~Alice~~ came.

Figure 12. Reconstruction of a fragment from Samuel Beckett: Murphy, Draft notebook, University of Reading, Cat. Nr. 5517/2, page 20r.

```
<xml>
  <subst>
    <del>Alice</del>
    <add>Cathleen</add>
  </subst> came.</p>
</xml>
```

Example 4. Transcription using the TEI del/add method.

```
<xml>
  <app>
    <rdg varSeq="1">Alice</rdg>
    <rdg varSeq="2">Cathleen</rdg>
  </app> came.</p>
</xml>
```

Example 5. Transcription using the TEI app/rdg method.

Human expectations and algorithmic rules

With this TEI encoded input, we expect HyperCollate to recognize the substitution as a break in the linearity of the text and to treat the two parts in a nonlinear way. Moreover, since not all text encoders group `` and `<add>` elements within a `<subst>` tag (see section 2.3., HyperCollate should differentiate between a substitution and an unconnected deletion and addition by providing clear rules on how to trigger its special treatment of substitutions for transcriptions that don't include `<subst>` elements. For now, HyperCollate implements a heuristic: if a `` element is directly followed by an `<add>` element, unspaced, we take them as belonging together, so the heuristic places a `<subst>` around the two. If the `` and `<add>` elements are separated by a white space, they are not grouped together, and treated in a linear way. Of course, as with all heuristics, this is not an optimal solution, as some scholarly editors do use a white space between a `` and an `<add>` element, even if they consider them to belong to the same substitution.

49

The `<subst>` or `<app>` element demarcates the beginning of a split in HyperCollate's token stream. From that point on there are two substreams: one substream with the `` or first `<rdg>` element and its textual content, and one substream with the `<add>` or second `<rdg>` element and its textual content. The ``, `<add>`, and `<rdg>` elements are unordered: they are at the same distance from the root node `<xml>`.

50

A `<subst>` element can contain more than one `` and more than one `<add>` element. In principle each `` and `<add>` represents a new branch. A `<subst>` can occur within another `<subst>`, the result is simply that the stream of text within that branch splits up again.

51

Hypergraph of a substitution

Figure 13 shows the internal graph model for the substitution in the example encoded with the `/<add>` method.

52

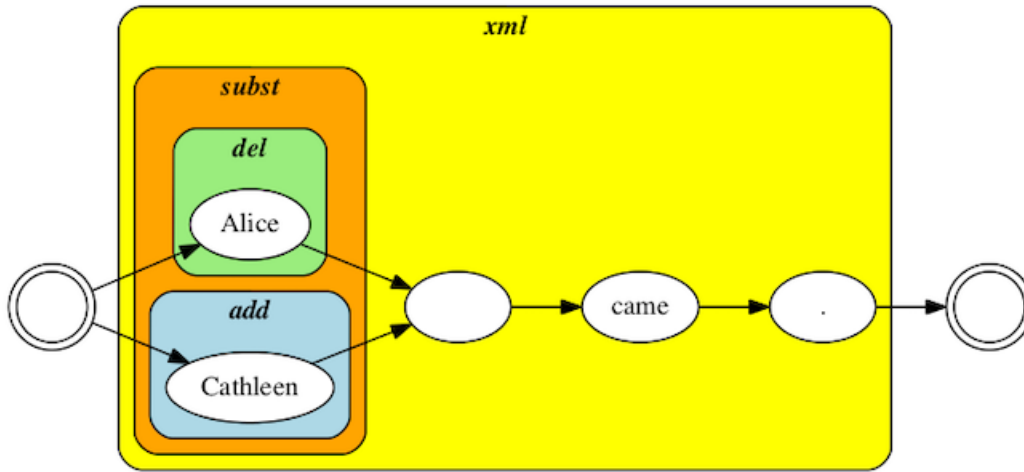


Figure 13. Internal hypergraph model of the `<add>/` tagging of the example.

The `<app>/<rdg>` method equally divides the token stream up into two paths (see figure 14).

53

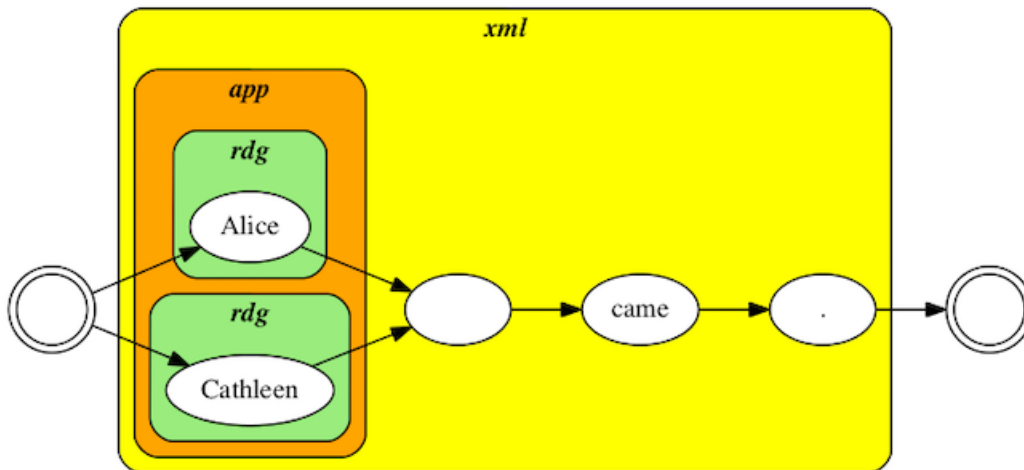


Figure 14. Internal hypergraph model of the `app/rdg` tagging of the example.

3.2.4 Multiple Alignment Options in Longer Substitutions

In the case of the substitution of a group of words, there will usually be more than one way to align the witnesses. The differences in alignment stem from a difference in focus: does the scholarly editor want to give priority to the unit of the substitution, or should the alignment of matching words receive priority? The traditional approach in collation of starting from exact string matches and grouping the variants in columns between the matches has the advantage of accentuating the similarities between witnesses, but the unit of the substitution can become obscured as it is spread over multiple columns. Holding the substitution together in the alignment, on the other hand, keeps the focus on the nonlinear nature of manuscripts as witnesses. It marks the spot where “something happens” on the manuscript and the totality of a textual operation is presented as one block. The drawback is a potential loss of information if the occurrence of a matching word across witnesses is not indicated in the collation output.

54

Consider the examples below.

55

stayed his hand.
Murphy seized him by the arm.

Figure 15. Reconstruction of a fragment from Samuel Beckett: Murphy, Draft notebook, University of Reading, Cat. Nr. 5517/6, page 26r.

```
<xml>Murphy
  <subst>
    <del>seized him by the arm.</del>
    <add>stayed his hand.</add>
  </subst>
</xml>
```

Example 6. Transcription using the TEI del/add method.

```
<xml>Murphy
  <app>
    <rdg varSeq="1">seized him by the arm.</rdg>
    <rdg varSeq="2">stayed his hand.</rdg>
  </app>
</xml>
```

Example 7. Transcription using the TEI app/rdg method.

Given a collation with two slightly different witnesses (B: “Murphy stayed his hand.”, C: “Murphy stayed the arm.”), different scholarly editors might arrive at different alignment tables, but they will largely fall into one of two categories: alignment on matching tokens or on the unit of the substitution. Currently, HyperCollate aligns the witnesses on the unit of substitution and will produce the following alignment table:

56

[A]	Murphy	[+] stayed [-] seized him by	[+] his hand [-] the arm	.
[B]	Murphy	stayed	his hand	.
[C]	Murphy	stayed	the arm	.

Figure 16. Collation output expressed in an ascii table with the alignment favouring the unit of the substitution.

In the future, HyperCollate aims to offer users two options in its output alignment: to give dominance to the matching of tokens, or to preserve the unit of the substitution as much as possible. Users will be then able to indicate their preference by way of a parameter in the collation command. The collation hypergraph and the two possible alignment options are discussed further in section 3.3.4.

57

3.3. Collation Output Visualization

Now that we have established the types of instances of nonlinearity and made our expectations explicit of how HyperCollate should handle such instances, it is time to discuss the current visualizations of the output of HyperCollate.

58

3.3.1 Deletions and Additions

The collation hypergraph can be visualized, among other formats, as an alignment table (see figure 17).

59

[A]	& [+]	finally	in what order he would eat the biscuits	[-]	, in the cockpit	when the time came.
[B]	&	finally	in what order he would eat the biscuits			when the time came.

Figure 17. ASCII alignment table of the collation output. In the TEI-XML transcription, the in-text revisions in witness "A" were tagged with <add> and .

In figure 16 the "[+]" and "[-]" in the "A" version correspond to the <add> and elements in the input XML. In the case of the <app>/<rdg> method, the value of the @varSeq attribute is included in the output, as shown in figure 18.

60

[A]	&	<2> finally		<2>		
	&	<1>	in what order he would eat the biscuits	<1>	, in the cockpit	when the time came.
[B]	&	finally	in what order he would eat the biscuits			when the time came.

Figure 18. ASCII alignment table of the collation output, with the in-text revisions in witness "A" coded with <app> and <rdg>.

3.3.2 Instant Revisions

Collated against a fictional second witness B, the instant revision produces the following ASCII alignment table (figure 19). Note that the visualization options of the ASCII alignment table are limited, which is why the instant deletion is visualized as a regular deletion and the text that follows, "gave such a," is placed in the same cell and directly above the instant deletion.

61

[A]	&	now	gave such a	
			[-] threw up his	jerk
[B]	and	now	gave such a	pull

Figure 19. ASCII alignment tables of the collation output of an instant revision.

The HTML alignment table is more expressive and produces a more accurate visualization of the collation result (see figure 20).

62

A	&	now	threw up his gave such a	jerk
B	and	now	gave such a	pull

Figure 20. HTML alignment tables of the collation output of an instant revision.

The collation hypergraph visualization (figure 21), finally, demonstrates once more that the instant deletion does not produce an alternative path through the text of witness A. The graph also shows that even though the text tokens "&" and "and" as well as "jerk" and "pull" are aligned in the alignment table visualization, they are not considered a match by the collation algorithm.

63

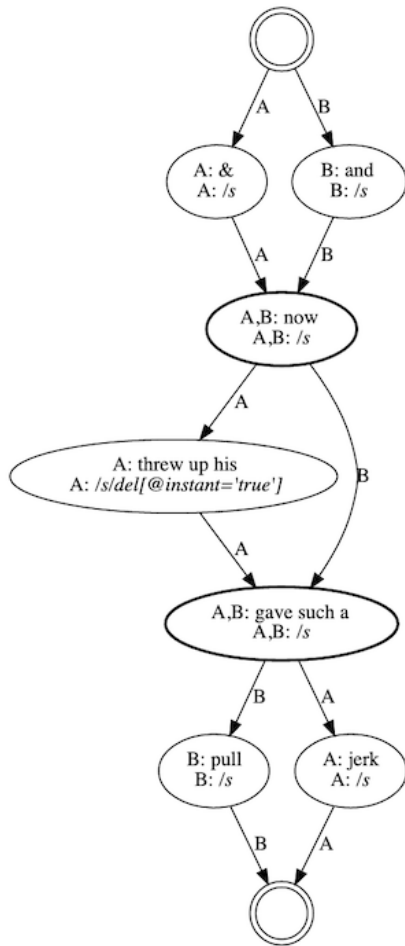


Figure 21. Collation hypergraph of the collation output of an instant revision.

3.3.3 Substitutions

Figure 22 shows the ASCII alignment tables for a collation of Beckett’s substitution of “Alice” with “Cathleen” against another version stating “Cathleen came.” When the substitution is encoded with the <add> method, the word in the element is preceded by “[−]”, and the word in the <add> element with “[+]”. In the case of a transcription that uses the <app/><rdg> method, HyperCollate reproduces the values of the @varSeq attributes in the <rdg> elements.

64

[A]	[+] Cathleen		
	[-] Alice		came.
[B]	Cathleen		came.

[A]	<2> Cathleen		
	<1> Alice		came.
[B]	Cathleen		came.

Figure 22. ASCII alignment tables of the collation output, del/add method on the left, app/rdg on the right.

The collation hypergraph (see figure 23), from which the ASCII alignment tables are derived, holds the most contextual information: each node contains a full XPath expression. In a case where two witnesses match on a word, but are not on the same level of the XML tree, two different XPath expressions are listed in the node, bringing both the similarities and the differences to the user’s attention.

65

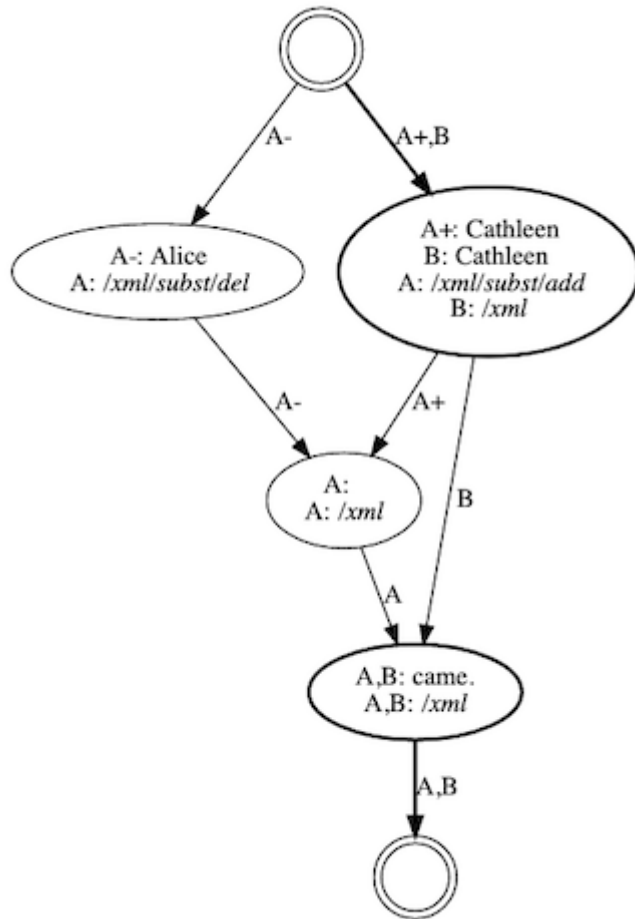


Figure 23. The collation hypergraph of the substitution example.

The instance of nonlinear text is visualized by two separate edges and nodes labeled with the same siglum, “A”. In order to make it easier to follow the path of a certain revision campaign, the edges contain not only information about the witness sigil, but also whether the text is part of a deleted or added branch. This information is visualized on the edges with [-] and [+] signs for deletions and additions respectively. Furthermore, information that is shared by more than one witness is given a thicker edge and node border (see e.g., Jänicke et al. [2015]). In this example, the words “Cathleen” and “came” are considered matches between A and B and thus share text nodes with a thicker border. Finally, the information about the location of the tag in the XML tree is added to the node as an XPath expression.

66

3.3.4 Longer Substitutions

In the case of a collation of the Beckett example shown in figure 15 with two other witnesses (B: “Murphy stayed his hand.”, C: “Murphy stayed the arm.”), the collation hypergraph identifies two variant groups across the witnesses (figure 24).

67

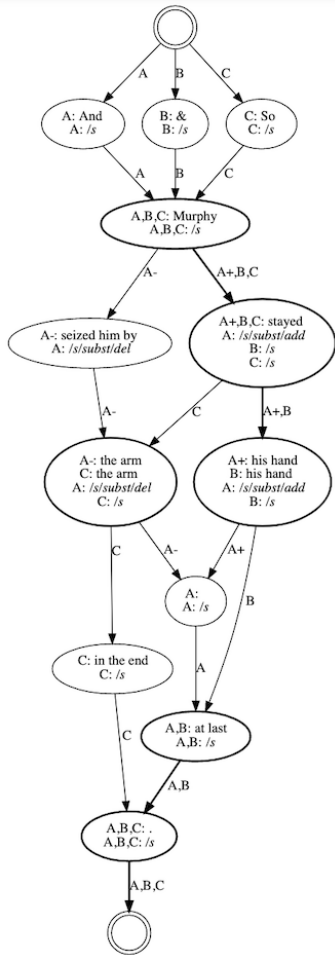


Figure 24. Collation hypergraph of an example with three witnesses, one of which contains a long substitution.

HyperCollate can express this graph in two possible alignment tables.

68

Alignment on matching tokens

The first option presents an alignment of each of the matching tokens in a separate column (figure 25). It splits the substitution up into token-per-token units in which a deleted and added token are placed together in a column based solely on their position in the token stream.

69

[A]	Murphy	[+] stayed [-] seized	[+] his [-] him	[+] hand [-] by	[-] the	[-] arm	[+] . [-] .
[B]	Murphy	stayed	his	hand			.
[C]	Murphy	stayed			the	arm	.

Figure 25. Collation output expressed in an ascii table with the alignment favouring matching tokens.

This alignment maximally emphasizes the matching words across witnesses.

Alignment on the unit of the substitution

Alternatively, HyperCollate can take its lead from the unit of the substitution to group words together in its output options, as in figure 26.

70

[A]	Murphy	[+] stayed [-] seized him by	[+] his hand [-] the arm	.
[B]	Murphy	stayed	his hand	.
[C]	Murphy	stayed	the arm	.

Figure 26. Collation output expressed in an ascii table with the alignment favouring the unit of the substitution.

In this particular example the two variant groups from the collation graph are presented as separate columns without further subdivision. A variant of the output in an HTML table draws attention to the matching words in these columns (“stayed,” “his hand” and “the arm”) by way of a corresponding background color (see figure 27).

A	And	Murphy	stayed seized him by	his hand the arm	at last	.
B	&	Murphy	stayed	his hand	at last	.
C	So	Murphy	stayed	the arm	in the end	.

Figure 27. Collation output expressed in an alignment table in HTML.

3.4. Evaluation

In its genetic modules, the BDMP offers users the option of collating all the prepublication versions of one particular sentence in a work by incorporating on-the-fly collation with CollateX 1.7.1. By using CollateX’s pre-tokenized JSON input feature and attaching the extra properties “del” and “add” to tokens (section 2.4.1), the BDMP can visualize additions and deletions in the generated alignment tables in the same way as it does in the transcription visualizations: deletions struck through and additions in superscript. A comparison of the alignment tables from CollateX and HyperCollate can illustrate the improvement of HyperCollate’s treatment of witnesses in the collation process (see figure 28).

71

Alice	Cathleen came.	[A] [+] Cathleen [-] Alice	came.
	Cathleen came.	[B] Cathleen	came.

Figure 28. CollateX alignment (left) versus HyperCollate alignment (right) of the example from section 3.2.3.

Although the CollateX table adequately intuits the in-text revision on the manuscript through the formatting convention, the alignment that CollateX outputs is still conceptually less correct than a visualization that places “Alice” and “Cathleen” on the same level. HyperCollate combines the advantages of the two current approaches described in sections 2.4.1 and 2.4.2 and does not have the drawbacks. A witness with in-text variation is treated as one witness both in input and output, and the collation algorithm is able to align the two parts of the substitution vertically instead of horizontally in its output by treating them as unordered during alignment, which is a considerable step forward.

72

Murphy	seized him by	the arm	stayed his hand	.	[A]	Murphy	[+] stayed	[+] his hand	[+] .
Murphy			stayed his hand	.			[-] seized him by	[-] the arm	[-] .
Murphy	stayed	the arm		.	[B]	Murphy	stayed	his hand	.
				.	[C]	Murphy	stayed	the arm	.

Figure 29. CollateX alignment (left) versus HyperCollate alignment (right) of the example from section 3.2.4.

In the case of the example discussed in sections 3.2.4 and 3.3.4, the method of collating as one linear witness with markup “passed along” (section 2.4.1), the CollateX output is inadequate, as the alignment table in figure 28 shows. Treating the two parts of the substitution in a linear way is detrimental to the alignment. The words “the arm” and “stayed” are transposed between the first and third witness, and as “the arm” occurs first in the first witness, CollateX marks “the arm” as invariant between those two witnesses in favor of aligning “stayed” which occurs in all three witnesses. The HyperCollate output succeeds quite well in conveying the nonlinearity in the first witness and grouping and aligning the differences with the subsequent witnesses.

73

Splitting the first version up into two (sub)versions as discussed in section 2.4.2 produces the desired alignment (figure 30), but misrepresents the manuscript by including the word “Murphy” in version “A-layer1” as well as in “A-layer2,” as the word “Murphy” only occurs once on the document.

74

A-layer1	Murphy	seized him by	the arm	.
A-layer2	Murphy	stayed	his hand	.
B	Murphy	stayed	his hand	.
C	Murphy	stayed	the arm	.

Figure 30. CollateX alignment table output with the substitution in version “A” split up into two (sub)versions.

4. Future Work

At the moment of writing, HyperCollate can already be used via Github or in a Kotlin notebook.^[18] The collation tool is still in a prototype state and we can identify several areas of further development. On the one hand, these developments regard the tool’s functionalities, such as generating different output formats and collating witnesses regardless of the order in which they are inputted (“order independent collation”). At the moment, we use the progressive alignment method, which means that the order in which the witnesses are fed to the collation program influences the outcome. Evidently, we do not want the order in which the witnesses are inputted to affect the collation output, but this is a computationally highly complex issue for which as of yet no clear-cut solution has been found.

75

Another area of future work is more philological, concerning questions such as “How should HyperCollate handle open variants?” and “Can we include other types of markup in the collation?” In the following paragraphs, we will briefly expand on these and other questions.

76

4.1. Output Formats

The output of collation software is often used as the basis for further processing or editorial correction. It is therefore important to offer a myriad of options that can be edited afterwards. HyperCollate can currently output collations either as a hypergraph (expressed in SVG, dot, PNG formats) or as an alignment table in ASCII. An HTML implementation of a comparable table format is in development.

77

We are also working on output expressed in XML. With the encoding schema we hope to stay as close to the TEI parallel segmentation method as possible. As a starting point we envisage the following:

78

```
<app>
  <rdg wit="#A-" type="deletion" varSeq="1">
```

```

        <del>Alice</del>
    </rdg>
    <rdgGrp type="tag_variation_only">
        <rdg wit="#A+" type="addition" varSeq="2">
            <add>Cathleen</add></rdg>
        <rdg wit="#B">Cathleen</rdg>
    </rdgGrp>
</app> came.

```

This encoding complies with the TEI guidelines except that, like in the collation hypergraphs, a “+” or a “-” is added to the siglum “A” when that version splits up into two branches which are placed in two <rdg> elements. As the XML counterpart of a node in the graph where the witnesses have the same text but a different XML context, they are placed as separate readings in a reading group (<rdgGrp>). The element has a @type attribute with the value “tag_variation_only” to indicate that the readings only differ in the level of the XML elements. The schema is feasible when it concerns simple substitutions, but we will need to determine how well it handles long substitutions or substitutions within substitutions.

79

4.2. Substitutions Within a Word

Substitutions on the character-level within a word constitute a particular challenge to the automatic collation process. HyperCollate works at the granularity level of the word as a token, and revisions within a word produce isolated characters as tokens.

80

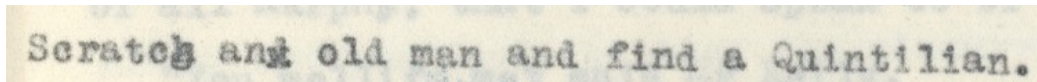


Figure 31. Samuel Beckett: Murphy, Typescript, Harry Ransom Center, Cat. Nr. SB 5/2, page 11r.

In TEI-XML, Beckett’s typo in “Scratch” (figure 31) could be transcribed as: `Scratchg<add>h</add>`. Currently this encoding produces three separate tokens: “Scratch,” “g” and “h,” none of which will be aligned with the occurrence of “Scratch” in another version. This is a very challenging problem for any collation algorithm, because it creates the need for an alignment within an alignment: aligning words in a first step and characters subsequently.

81

There are a few avenues of development to consider. An approach of “near-matching” might produce adequate results. Another option would be to encode the entire word within a <mod> element: `<mod>Scratchg<add>h</add></mod>`,^[19] which could trigger HyperCollate to translate this into `<subst>Scratchg<add>Scratch</add></subst>`. Finally, using the <app>/<rdg> transcription method, a construction with an extra reading, `<rdg type="lit">`,^[20] might be considered:

82

```

<app>
  <rdg wit="A" varSeq="1">Scratchg</rdg>
  <rdg wit="B" varSeq="2">Scratch</rdg>
  <rdg type="lit">Scratch<del>g</del><add>h</add></rdg>
</app>

```

4.3. Open Variants

Open variants are also an example of nonlinearity in text. The TEI guidelines suggest that open variants should be encoded with a <seg> element around the inline part of the open variant, followed by one or more <add> elements. For HyperCollate to apply the same treatment to open variants as to substitutions, it will need to recognize a <seg> element and treat it in a similar way to or <add>. Because the <seg> is a general element with several uses, HyperCollate will have to look for a <seg> element with a @type attribute with the value “alternative.”

83

4.4. Transpositions

A transposition “occurs when metamarks are found in a document indicating that passages should be moved to a different position.”^[21]

84

and saw it could be still made to serve.

Figure 32. Samuel Beckett: Malone Dies, Typescript, Washington University, Cat. Nr. MSS008/2/47, page 49r.

To tag transpositions, the TEI proposes that both parts be tagged in an element with a `@xml:id` attribute (the element most commonly used is `<seg>`), and that the two tags be declared as a transposition inside the `teiHeader`, in a `<transpose>` element within a `<listTranspose>`. The use of `@type="transposition"` inside the `<seg>` is not required. 85

A transposition can be said to be a nonlinear textual feature; it signals two possible readings of a fragment with a difference in word order. Transposition metamarks, however, do not violate the linearity of the text, as the text fragment does not have two alternatives for one word or phrase, but merely suggests a reordering of two or more words. A special treatment by HyperCollate of this textual feature is not needed to provide a meaningful result. Currently, the source XML is handled by HyperCollate linearly like all other material. The actual word order is collated, and the second, implied order can still be deduced from the markup in the transcription. The same applies to transpositions of letters within words. 86

Making the two word orders explicit in the `<app>/<rdg>` method of transcription, however, will allow for the two versions to be collated: 87

```
<s>and saw it could <app type="transposition"><rdg varSeq="1">be still</rdg><rdg
varSeq="2">still be</rdg></app> made to serve.</s>
```

5. Conclusion

In-text variation is essential for the study of textual variance and the TEI Guidelines offer several ways to encode this kind of information. Consequently, the need to include the information in the automatic collation of manuscript texts is apparent. Up to now, including in-text variation was only possible through workarounds that either require significant coding skills from the users – passing along markup tags as shadow-tokens on the JSON input of CollateX – or by manually creating pseudo-versions from the layers of in-text variation and treating them as regular versions. These approaches do ensure that information about the in-text variation is present in the collation output so that the nonlinear features of the individual witnesses can be visually reconstructed. However, the information is not part of the alignment process itself: the collation tools treat the input texts as a linear, ordered stream of characters and thus ignore the multilayered character of manuscript text. 88

In this paper, we argued that in-text variation provides meaningful information about the temporality of the writing process. We demonstrated that this type of variation constitutes nonlinear, partially ordered data and we argued that it should be treated as such by collation software. We therefore set out to develop a collation program, HyperCollate, that understands each witness as a stream of text tokens that splits into two or more “branches” at the point where the internal variation occurs and converges at the end of the variation. The occurrences of variation are marked by the scholarly editor with designated TEI-XML tags. HyperCollate thus produces a collation output that represents the multidimensional nature of the source text more adequately and thus corresponds with how the in-text variation was conceptualized and encoded. 89

The paper began with a description of our understanding of different types of in-text variation, and subsequently illustrated how that understanding translates logically. We then illustrated how the input of HyperCollate, TEI-XML transcriptions, are transformed into individual witness hypergraphs with partially ordered textual data. HyperCollate compares two witness hypergraphs and finds the best match between the branches of text. In case of more witnesses, HyperCollate compares them progressively. Accordingly, the TEI-XML transcriptions that hold so much scholarly information can be used as direct input without any need for pre-processing or flattening. We concluded that including the in-text variation produces a more refined output that more accurately represents the textual variation as interpreted by the scholarly editor during the transcription phase. 90

One interesting finding is that in case of a longer substitution, the best way to collate versions with in-text variation depends on the preference of the scholarly editor. They can either prefer to have the collation program align on the unit 91

of substitution, or to align on the individual words within the substitution. Future users of HyperCollate will be able to indicate their preference in the collation command. We also found that the visualization of HyperCollate's in- and output may depend on the user's preference. Both the input (a witness hypergraph) and the output (a collation hypergraph) of HyperCollate can be visualized as variant hypergraphs with or without markup nodes as colored hyperedges.

The visualizations are an attempt to provide more insight into the text and its revisions, but visualizing such a complex data structure is quite challenging. The output of HyperCollate simply contains a lot of information, and it is difficult to visualize all that information in a clear and insightful way. This becomes evident with the alignment table visualization: the alignment table allows scholarly editors to see at a glance which words are added and deleted, but at the same time using the [-] and [+] signs for in-text deletions and additions simplify the revision process. Still, an alignment table is generally easier to read than a collation hypergraph, and so far user-testing has shown that the alignment table does feel intuitive to scholarly editors. We will continue to experiment with the visualizations of the collation hypergraph, e.g., by adding [-] and [+] signs on the edges of the collation hypergraph or by using thicker edges if a token is present in many versions.

92

In conclusion, this article aimed to show how formalizing our understanding of in-text variation as nonlinear, partially ordered text can help to communicate our human understanding to a collation tool. We trained HyperCollate on two TEI encoding suggestions for in-text variation, `app/rdg` and `subst/del/add`, in order to illustrate that the formalization can work for different encoding styles. The larger argument made in this article concerns the benefits of reflecting upon our textual models and to what extent they impact our encoding models and subsequently the in- and output of text analysis tools.

93

The development of HyperCollate was propelled by a knowledge exchange between textual scholars and research software engineers. This exchange took the form of long discussions, much prototyping, experimenting, and testing. Realizing the value of our collaboration and ongoing knowledge exchange, we have tried to be transparent about our methodology and detailed our textual theories and expectations of the software as well as the decision making process of HyperCollate. Indeed, we found collaborating on the development of a collation program an insightful, iterative learning process that emphasizes the benefits of including computational methods in manuscript research.

94

Works Cited

- Andrews 2013** Andrews, Tara. 2013. "The third way: philology and critical edition in the digital age." *Variants*, vol. 10, pp. 61–76.
- Andrews and Macé 2013** Andrews, Tara L. and Macé, Caroline, 2013. "Beyond the tree of texts: Building an empirical model of scribal variation through graph analysis of texts and stemmata." *Literary and Linguistic Computing*, vol. 28, issue 4, pp. 504-521.
- Barabucci 2018** Barabucci, Gioele. 2018. "Diffi: diff improved; a preview." Presented at the 2018 ACM Symposium on Document Engineering (DocEng 2018), August 28-31, 2018, Halifax, Canada. *Proceedings of the ACM Symposium on Document Engineering 2018*, pp. 1-4.
- Barabucci 2020** Barabucci, Gioele. 2020. "The CMV+ P Document Model, Linear Version." *Versioning Cultural Objects: Digital Approaches*, vol. 13, pp. 153-170.
- Barabucci et al. 2013** Barabucci, Gioele, Borghoff, Uwe, Di Iorio, Angelo and Schimmler, Sonja. 2013. "Document Changes: Modeling; Detection; Storing and Visualization (DChanges)." Presented at the 2013 ACM Symposium on Document Engineering (DocEng 2013) in September 10-13, 2013, Florence, Italy. DOI: <http://dx.doi.org/10.1145/2494266.2494322>.
- Birnbaum 2015** Birnbaum, David. 2015. "Using CollateX with XML: Recognizing and Tracking Markup Information During Collation." Computer-supported collation with CollateX. Online: <http://collatex.obdurodon.org/xml-json-conversion.xhtml> (last accessed October 22, 2021).]
- Bleeker 2020** Bleeker, Elli, Bram Buitendijk and Ronald Haentjens Dekker. "Marking up microrevisions with major implications: Non-linear text in TAG." 2020. Presented at Balisage: The Markup Conference 2020, Washington, DC, July 27 - 31, 2020. *Proceedings of Balisage: The Markup Conference 2020. Balisage Series on Markup Technologies*, vol. 25. <https://doi.org/10.4242/BalisageVol25.Bleeker01>.
- Bleeker et al. 2018** Bleeker, Elli, Bram Buitendijk, Ronald Haentjens Dekker, and Astrid Kulsdom. "Including XML markup in the automated collation of literary text." Presented at the XML Prague conference 2018, February 8-10, 2018. *XML Prague Conference Proceedings*, pp. 77-96. Available at <http://archive.xmlprague.cz/2018/files/xmlprague-2018-proceedings.pdf#page=89> (last accessed October 22, 2021).
- Bleeker et al. 2019** Bleeker, Elli, Bram Buitendijk, and Ronald Haentjens Dekker. 2019. "From Graveyard to Graph: Visualisation of Textual Collation in a Digital Paradigm." *International Journal of Digital Humanities / Special Issue on Digital Scholarly Editing*, vol. 1, issue 2, pp. 141-163.

- Bordalejo 2013** Bordalejo, Barbara. 2013. "The Texts We See and the Works We Imagine: The Shift of Focus of Textual Scholarship in the Digital Age." *Ecdotica* 10: 64-76.
- Ciula and Eide 2014** Ciula, Arianna, And Øyvind Eide. 2014. "Reflections on Cultural Heritage and Digital Humanities: Modelling in Practice and Theory." *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*. New York: ACM.
- CollateX** Haentjens Dekker, Ronald and Gregor Middell. 2015. The Interedition Development Group *CollateX* (version 1.7.1). [Computer program]. Available at: <https://collatex.net/> and <https://mvnrepository.com/artifact/eu.interedition/collatex/1.7.1>.
- Colwell and Tune 1964** Colwell, Ernest C., and Ernest W. Tune. 1964. "Variant Readings: Classification and Use." *Journal of Biblical Literature* 83.3, pp. 253-261.
- DeRose 2004** DeRose, Steve. 2004. "Markup Overlap: a review and a horse." Presented at *Extreme Markup Languages 2004*. Montréal, Québec, August 2-6, 2004. <http://xml.coverpages.org/DeRoseEML2004.pdf> (last accessed October 22, 2021).
- Ferrer 2014** Ferrer, Daniel. 2014. *James Joyce: Brouillons d'un baiser. Premiers pas vers Finnegans Wake*. Paris: Editions Gallimard.
- Grésillon 1994** Grésillon, Almuth. 1994. *Eléments de critique génétique lire les manuscrits modernes*. Paris: Presses universitaires de France.
- Haentjens Dekker and Birnbaum 2017** Haentjens Dekker, Ronald, and David J. Birnbaum. 2017. "It's more than just overlap: Text As Graph." Presented at Balisage: The Markup Conference 2017, Washington, DC, August 1 - 4, 2017. *Proceedings of Balisage: The Markup Conference 2017. Balisage Series on Markup Technologies*, vol. 19. <https://doi.org/10.4242/BalisageVol19.Dekker01>.
- Haentjens et al. 2018** Haentjens Dekker, Ronald, Elli Bleeker, Bram Buitendijk, Astrid Kulsdom and David J. Birnbaum. 2018. "TAGML: A markup language of many dimensions." Presented at Balisage: The Markup Conference 2018, Washington, DC, July 31 - August 3, 2018. *Proceedings of Balisage: The Markup Conference 2018. Balisage Series on Markup Technologies*, vol. 21. <https://doi.org/10.4242/BalisageVol21.HaentjensDekker01>.
- Janicke et al. 2015** Jänicke, Stefan, Annette Geßner, Greta Franzini, Melissa Terras, Simon Mahony, and Gerik Scheuermann. 2015. "TRAViz: A visualization for variant graphs." *Digital Scholarship in the Humanities* vol. 30, issue suppl. 1, pp. i83-99.
- Kline 1998** Kline, Mary-Jo. 1998. *A Guide to Documentary Editing*. Baltimore: John Hopkin's University Press.
- Lebrave 1992** Lebrave, Jean-Louis. 1992. "La critique génétique: une discipline nouvelle ou un avatar de la philologie?" *Genesis* 1, pp. 33-72.
- Nury 2018** Nury, Elisa. 2018. *Automated Collation and Digital Editions: From Theory to Practice*. PhD thesis, London: King's College London. Available at: [https://kclpure.kcl.ac.uk/portal/en/theses/automated-collation-and-digital-editions\(1ffc4aa0-5ad5-4ca1-869a-ab3d528eed4a\).html](https://kclpure.kcl.ac.uk/portal/en/theses/automated-collation-and-digital-editions(1ffc4aa0-5ad5-4ca1-869a-ab3d528eed4a).html) (last accessed October 22, 2021).
- Peters 2005** Peters, Luuk. 2005. "Change detection in XML trees: a survey." *3rd Twente Student Conference on IT*. Available at http://www.poleia.lip6.fr/~gancarsk/grbd09/2005_03_B_Peters,L.J.-Change_detection_in_XML_trees_a_survey.pdf (last accessed October 22, 2021).
- Pierazzo 2015** Pierazzo, Elena. 2015. *Digital Scholarly Editing: Theories, Models and Methods*. Surrey: Ashgate Publishing, Ltd.
- Plachta 1997** Plachta, Bodo. 1997. *Editionswissenschaft: eine Einführung in Methode und Praxis der Edition neuerer Texte*. Stuttgart: Reclam.
- Scheibe 1995** Scheibe, Siegfried. 1995. "On the Editorial Problem of the Text." *Contemporary German Editorial Theory*. eds. Hans Walter Gabler, George Bornstein and Gillian Borland Pierce. Ann Arbor: The University of Michigan Press.
- Schmidt 2019** Schmidt, Desmond. 2019. "A Model of Versions and Layers." In *Digital Humanities Quarterly*, vol. 13, no. 3, available from <http://digitalhumanities.org/dhq/vol/13/3/000430/000430.html>.
- Schmidt and Colomb 2009** Schmidt, Desmond and Robert Colomb. 2009. "A data structure for representing multi-version texts online." *International Journal of Human-Computer Studies*, vol. 67, issue 6, pp. 497-514. <http://dx.doi.org/10.1016/j.ijhcs.2009.02.001>
- Shillingsburg 1996** Shillingsburg, Peter. 1996. *Scholarly Editing in the Computer Age: Theory and Practice* (3rd Edition). Ann Arbor, Mich.: University of Michigan Press.
- Sperberg-McQueen 1989** Sperberg-McQueen, C. 1989. "A directed-graph data structure for text manipulation". Paper presented at the *The 9th International Conference on Computers and the Humanities (ICCH) and 16th International Association for Literary and Linguistic Computing (ALLC) Conference*, University of Toronto, June 1989. Available from <http://www.w3.org/People/cmsmcq/1989/rhine-delta-abstract.html>.
- Van Hulle 1999** Van Hulle, Dirk. 1999. "Authenticity or hyperreality in hypertext editions: notes towards a searchable recherche." *Human IT: journal for information technology studies as a human science*, vol. 1, pp. 227-244.

Van Hulle 2004 Van Hulle, Dirk. 2004. "Compositional Variants in Modern Manuscripts." *Digital Technology and Philological Disciplines*, pp. 513-527.

Zeller 1975 Zeller, Hans. 1975. "A New Approach to the Critical Constitution of Literary Texts". *Studies in Bibliography*, vol. 28, pp. 231-264.



This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.