Digital Scholarship in the Humanities Advance Access published December 2, 2014

Computer-supported collation of modern manuscripts: CollateX and the Beckett Digital Manuscript Project

Ronald Haentjens Dekker
Department of IT R&D, Huygens Institute for the History of the Netherlands, Royal Netherlands Academy of Arts and Sciences, The Netherlands

Dirk van Hulle
Department of Literary Studies, University of Antwerp, Antwerp, Belgium

Gregor Middell
Institut für Deutsche Philologie, Universität Würzburg, Würzburg, Germany

Vincent Neyt
Department of Literary Studies, University of Antwerp, Antwerp, Belgium

Joris van Zundert
Methodology Research Program, Huygens Institute for the History of the Netherlands, Royal Netherlands Academy of Arts and Sciences, The Netherlands

Correspondence:
Joris van Zundert.
Huygens Institute for the History of the Netherlands, Royal Netherlands Academy of Arts and Sciences, PO Box 90754, 2509 LT, The Hague, The Netherlands
Email:
joris.van.zundert@huygens.knaw.nl

Abstract

Interoperability is the key term within the framework of the European-funded research project Interedition,[1] whose aim is 'to encourage the creators of tools for textual scholarship to make their functionality available to others, and to promote communication between scholars so that we can raise awareness of innovative working methods'. The tools developed by Interedition's 'Prototyping' working group were tested by other research teams, which formulate strategic recommendations. To this purpose, the Centre for Manuscript Genetics (University of Antwerp), the Huygens Institute for the History of the Netherlands (The Hague), and the University of Würzburg have been working together within the framework of Interedition. One of the concrete results of collaboration is the development and fine-tuning of the text collation tool CollateX.[2] In this article, we would like to investigate how the architecture of a

Digital Scholarship in the Humanities © The Author 2014. Published by Oxford University Press on behalf of EADH. All rights reserved. For Permissions, please email: journals.permissions@oup.com
doi:10.1093/llc/fqu007

digital archive containing modern manuscripts can be designed in such a way that users can autonomously collate textual units of their choice with the help of the collation tool CollateX and thus decide for themselves how efficiently this digital architecture functions—as an archive, as a genetic dossier, or as an edition. The first part introduces CollateX and its internal concepts and heuristics as a tool for digitally supported collation. How this tool can be integrated in the infrastructure of an electronic edition is discussed in part two. The third and final part examines the possibility of deploying CollateX for the collation of modern manuscripts by means of a test case: the Beckett Digital Manuscript Project (www.beckettarchive.org).

1 Computer-supported collation with CollateX

Following John Unsworth's textual scholarship workflow typology of scholarly primitives (Unsworth 2000), it is clear that text comparison is pivotal to any kind of textual scholarship. The role of text comparison becomes paramount in any scholarly editing project that involves critical enquiries about the edited text witnessed in multiple versions. Conducting such a collation is a tedious and error-prone work,[3] especially because the required attention to detail is highly exacting compared with the repetitive and mechanical nature of the task. From that perspective, this type of work seems an ideal candidate for automation, not only because computers can support users in tedious error-prone duties rather efficiently, but specifically because the number of versions is often so large that it is simply not feasible any more to compare each witness against another manually.[4]

The application of computers or other apparatuses to support the collation of texts already has a long-standing tradition in and of itself (Smith 2000), reaching back at least to the usage of optomechanical devices like those pioneered by Charlton Hinman. Since then, the semiautomatic collation of texts has been a well-established area for the application of software, offering support in managing large text traditions, in comparing predetermined passages of different versions as well as in storing and rendering the results. However, it continued to be the user's duty to orchestrate the whole process and guide the computer in comparing relevant passages by manually calibrating the complex input to make it fit rather basic comparison algorithms.[5] Recent advancements in the field of computational biology, a field closely related when viewed from the computational perspective, resulted in renewed attempts to further the degree of automation achieved thus far in the comparison of natural language texts.[6] Protein sequences—not unlike texts in natural language—can be modeled as sequences of symbols, whose differences can be understood as a set of well-defined editing operations (Levenshtein 1966), which transform one sequence into another and can be computed. The analogy goes even further, as the consecutive evaluation of assumed editing operations between protein sequences on the one hand and texts on the other hand bears striking similarities, as they often provide the basis for further stemmatic analysis and genetic reasoning (Spencer and Howe, 2004). The only subtle but crucial problem with this analogy is that, while biologists can afford to leave aside methodological questions about the intentionality of assumed 'editing operations' on protein sequences, philologists cannot always base their reasoning on computed differences between texts. The prime example for this kind of difference is a transposition of two passages, which has been explicitly marked by the author in the manuscript (for instance, via arrows or a numbering scheme) and interpreted as such by the editor, but the intentionality of which cannot be computed deterministically by collation software, even if the transposition itself can be detected.

In addition to this dilemma, there are numerous workflow-related challenges to surmount when proper integration of collation software with a digital editing environment becomes a concern. It makes computer-supported collation not only a computationally complex but also an architecturally challenging problem for software developers. As in the traditional trilogy of ‘recensio’, ‘examinatio’ (including ‘collatio’), and ‘emendatio’ (see for instance Grafton *et al.*, 2010, p. 506), the collation of digital texts is again central to the editorial workflow, with consecutive architectural dependencies on many adjacent building blocks of the editing environment. The workflow may apply specific modeling and encoding of text versions as well as possible automatic linguistic annotation like part of speech tagging to be able to compare texts in a more expressive manner than on plain character string level. This might be the case for instance where lemmatization is applied. Also, specific workflows might require the ability of a human interacting with the collation result or process. Certain idiosyncrasies of witnesses and their texts, for instance, might have to be modeled and/or encoded by human intervention where optical character recognition does not serve well, e.g. in the case of visual poetry; or researchers may need to intervene in the process of automatic linguistic annotation of textual versions to make them comparable in a more sensible way, for instance, in supervised learning methods or in cases of languages that are poorly supported by the current state-of-the-art in natural language processing, such as medieval Dutch or neo-Latin; or it may be necessary to manually annotate the output of densely marked-up and interconnected text versions as a result of their comparison yielding differences previously unnoticed. Many newer approaches to the problem of collation offer interesting solutions to the computational challenge, but most of them do not fully address the architectural challenges, nor do they approach the problem as one which can only be solved semiautomatically given the methodological framework of its domain.

Consequently, existing solutions either remain within the realm of decision-support systems, which mainly help scholars keep the overview while producing essentially handcrafted collation results and transforming them into a commented critical apparatus,[7] or they automate the collation in a way that is tailored to a specific use case and/or runtime environment. The general applicability of the latter approach can then only be approximated via quantitative properties of its specific input and the accuracy of the achieved results.[8] In contrast, we would like to offer a third, rather pragmatic approach, in which we first dissect the problem of collation into smaller more manageable subproblems and then show by an example how each of these subproblems can be addressed in a way that is more fitting to its application domain and with a higher chance of applicability to the variety of requirements stipulated by the many different scholarly environments in which the collation of texts and its adjacent scholarly tasks have to be performed.

1.1 Comparing existing solutions

Our example for this approach is CollateX, a prototypical collation tool, developed in the context of Interedition. Shortly after the project started, it became clear that a proper requirements analysis for a versatile collation tool would need input from a range of stakeholders as wide as possible, including users and interested developers as well as implementers of existing solutions. A collation summit and a collation workshop were held in Gothenburg and Brussels in 2009, co-organized by the European COST Action 32 ‘Open Scholarly Communities on the Web’,[9] which invited implementers of three collation tools—literary scholars, digital humanists, and developers of XML database software—to discuss conceptual commonalities between their fields of expertise as they relate to the collation of texts. The immediate result was the agreement on a modularization of the digital collation process into a couple of well-defined steps, which—if applied in order and/or iteratively—allows the collation of texts to be supported more flexibly by implementations adhering to this separation of concerns.[10] Four basic steps were defined. The first is the tokenization of digital texts to be compared—in effect the segmentation of the texts into the sequence of tokens that will be compared. The second step is the alignment of tokens from different texts, which essentially identifies which segments of tokens match between the text—effectively also identifying where the compared texts differ and thus implying or assuming edit operations in those places. The third step is the analysis of the computed alignment, which introduces an interpretative aspect into the process as edit operations are now qualified (e.g. as deletion, addition, or transposition). The fourth and final step is the output/visualization of collation results. The workflow of these four steps has since become informally known as the 'Gothenburg model'. We will explain the various steps in more detail below.

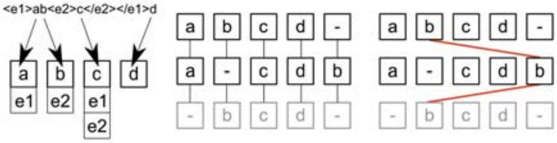


Fig. 1 Schematic representation of the tokenization (left), alignment (middle), and analysis (right) phases of a collation workflow

Although any collation software can compare texts on a character-by-character basis, in the more common use case, before collation each text (or comparand) is normally split up into segments or tokens and compared on the level of the token rather than on the character-level. This familiar step in text (pre)processing, called 'tokenization', is performed by a tokenizer and can happen on any level of granularity, for instance, on the level of syllables, words, lines, phrases, verses, paragraphs, text nodes in a normalized XML DOM instance, or any other unit suitable to the texts at hand. Another service provided by tokenizers as defined in our model relates to marked-up texts. As most collators compare texts primarily based on their textual content, embedded markup would usually get in the way and therefore needs to be filtered out—but must be kept as stand-off annotations during tokenization—so the collator can henceforward operate on tokens of textual content. Annotations must be kept because it might be valuable to have the markup context of every token available, for example, if one wants to make use of it in the comparison of tokens during the alignment step (see below). The schematic diagram on the left in Fig. 1 depicts this process: the upper line represents a comparand, each character a, b, c, and d, an arbitrary token, and the XML tags e1 and e2 are examples of embedded markup. A tokenizer transforms this marked-up text into a sequence of individual tokens, each referring to its respective markup/tagging context. From now on, a collator can compare tokenized comparands to others based on its tokenized content and does not have to deal with its specific notational conventions anymore, which are often rather specific to a particular markup language, dialect, or project. During the tokenization step, it is also possible to normalize each token, so the subsequent comparison can abstract from certain specifics, such as case-sensitivity or even morphological variants. In most use cases, we have found that abstracting away from such specifics yields useful collation results. However, it should be noted that there is no principal methodological or technical reason to enforce such abstraction. In cases where specifics would turn out to be useful as information for alignment of comparands, the model allows us to take into account such specifics.

When the comparands have been tokenized, a collator will align the tokens of all comparands involved. Aligning comparands implies the matching of equal tokens and the insertion of 'empty' tokens (so-called 'gap tokens') in such a way that the token sequences of all comparands line up properly. As mentioned before, this specific task of a collator is computationally similar to the problem of sequence alignment, as it is also encountered, for example, in computational biology. Looking again at an example (Fig. 1, center diagram), we assume that three texts (each depicted in its own column) are being compared: the first consists of the token sequence 'abcd', the second reads 'acdb', and the third 'bcd'. A collator might align these three comparands as depicted in a tabular manner. Each comparand occupies a column, matching tokens are aligned in a row, and necessary gaps as inserted during the alignment process are marked by means of a hyphen. Depending on the perspective from which one translates this alignment into a set of editing operations, one can conclude, for example, that the token 'b' in the second row was omitted in the second comparand or that it was added in the first and the third. A similar statement can be made about ‘b’ in the last row by just inverting the relationship of being added/omitted.

In addition to atomic editing operations computed in the alignment step, a further analysis of the alignment, conducted by the user and supported by the machine, can introduce additional interpretative preconditions into the process. Repeating the previous example in Fig. 1 (right diagram), one might interpret the token ‘b’ in columns 2 and 5 as being transposed instead of as being simply added and omitted. Whether these two edit operations actually can be interpreted as a transposition ultimately depends on the judgment of the editor and can at best be suggested, though not conclusively determined, via unambiguous heuristics. That is why an additional analytical step, in which the alignment results are augmented (and optionally fed back as preconditions into the collator), appears as essential to us in order to bridge the methodological ‘impedance’ between a plain computational approach to the theoretical problem and the established hermeneutical approach taken in practice. In some cases, even human interpretation may of course not determine decisively whether an actual transposition took place. We may have to conclude that some cases of potential transposition cannot be determined with absolute certainty.

The obvious remaining step is the output of the collation results, which is again a complex task. The requirements here range from the encoding of the results according to various conventions, markup dialects, and formats required by other tools to the visualization of results in multiple facets, be it in a synoptic form, either as a rendering focusing on one particular text and its variants, or as a graph-oriented networked view, offering an overview of the collation result as a whole.

After establishing this separation of concerns, implementers of collation-related software can henceforth focus on specific problems. For instance, the collation tool Juxta[11] has a feature-rich tokenizer for XML-encoded texts since version 1.4, which has been extended constantly in consecutive versions. Juxta also has support for larger comparands as well as stand-off annotations and is available as a self-contained software library[12] for reuse in other tools. Comparable work is ongoing to generalize Juxta’s visualization components.[13]

1.2 Comparing alignment algorithms

The main emphasis of CollateX’s development is on improving the alignment step. As mentioned in the introduction, aligning sequences of symbols is a well-known problem in computer science having many applications, notably in the field of computational biology. It has also been noted that the adoption of existing sequence alignment algorithms for use in the context of philology poses several problems, some of a conceptual methodological nature and some of a practical technical nature. Rather than providing a complete account of the pros and cons of particular algorithms, a task better undertaken elsewhere, we would like to draw attention to three recurring criteria, on which the quality of recent alignment algorithms is evaluated:

1. Transposition detection
   Detecting arbitrarily transposed passages in versions of a text is a much harder problem when done in the context of sequence alignment than computing insertions, deletions, and substitutions. Schmidt concludes his analysis of this problem (Schmidt 2009) with a pragmatic solution by stating that, given an NP-complete computational problem and no guaranteed correspondence between an optimal computational result and the outcome desired by the user, a heuristic algorithm might be the best solution. Accordingly, algorithms that try to detect transpositions do so heuristically and refer to benchmarks measuring computationally detected transpositions against manually predetermined ones.
2. Support for flexible token matching
   The well-known distinction between substantial versus accidental variants as well as other factors, like orthographic variation, require alignment algorithms to match tokens more flexibly than just via exact character matching. Some algorithms use edit distance–based thresholds for this purpose (e.g. Spencer/Howe’s or Juxta’s), whereas others rely on lookup tables predefined by the user, which

list possible mappings of tokens to match them despite their differing character content.

3. Base-Text-/Order-Independence
   Alignment algorithms like Juxta's compare versions one-on-one, so that as soon as more than two versions are to be compared, the task has to be reduced to pairwise comparison of two versions at a time and consecutive merging of the pairwise results. Spencer and Howe have shown the potential functional dependence of such a unified result on the order in which pairwise comparisons are merged. This poses a problem for genetic research based on such results, since a suitable order in which the pairwise comparisons should be merged depends on a hypothesis about which texts are closer related to each other and whose comparison results should consequently be merged first (Spencer and Howe, 2004).

CollateX's aligner tries to tackle all of these problems by following the modularization outlined in the section above and by finding ways to align tokens that do not inherit the trade-offs of existing sequence alignment algorithms. As such it has to be characterized as experimental, but at the same time it already yields promising results.

1.3 Comparing texts with CollateX

This section gives an overview of the major concepts by which CollateX aligns tokens of comparands. We begin by explaining the basic challenge of aligning two comparands including the detection of transpositions and extend the challenge stepwise up to the alignment of multiple versions.

Most alignment algorithms work on the basis of the following editing operations: insertion, deletion, and substitution. These operations are well defined, e.g. via Levenshtein's concept of the edit distance. A frequently recurring problem when comparing two versions of a text is the phenomenon where a passage of a text has been moved between them (i.e. transposed). Moreover, transposed passages of a text usually are not transposed literally, but contain small changes on their own, which makes the challenge to detect these even harder. Alignment algorithms that are constrained to the editing operations just mentioned will regard a transposition either as a deletion and an insertion or, in case two passages have been swapped, as two substitutions. CollateX releases this constraint by handling transpositions as an additional kind of editing operation and trying to detect those operations.

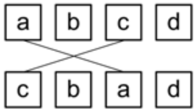


Fig. 2 Trivial case of transposition

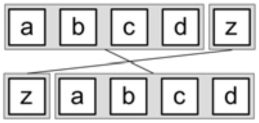


Fig. 3 Less trivial case of transposition

To start with a trivial case, detection of transpositions is easy when all tokens in the compared versions are unique (Fig. 2).

When we look at the different tokens from the two versions in each position, then it is easy to see that ‘a’ is transposed with ‘c’ and ‘c’ with ‘a’. Apart from all tokens being unique, the previous example also assumes that moved passages of text are exactly one token long. In the next example (cf. Fig. 3), we drop this constraint as well.

The desired result would be that the sequence ‘a b c d’ is transposed with ‘z’. The trivial approach described above for the detection of transpositions would not work in this case. Real-world cases of transposition involve arbitrary length sequences of tokens moving over seemingly arbitrary distances in text, in the process, more often than not, also changing the internal order of the sequence to various extents. To solve this problem, we need a more elaborate form of token matching. To this purpose, we use a match table, which is a document-to-document matrix, allowing us to compare two variant witnesses, each word to each word. Let us first consider a case where there is no variation (cf. Fig. 4). We put the tokens of witness 1 as the column headers of the matrix and the tokens of the identical

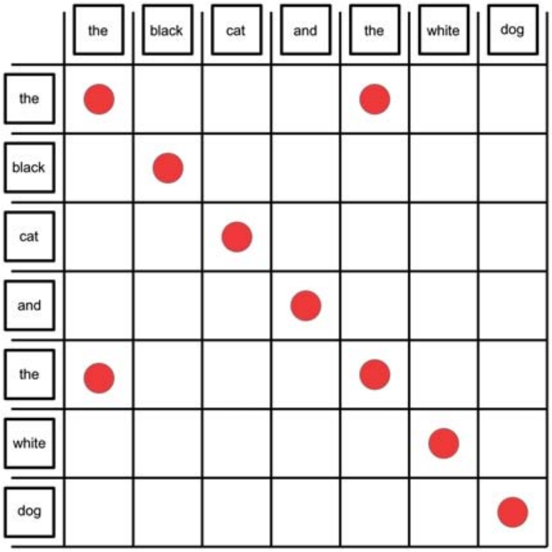


Fig. 4 Document-to-document matrix applied as a match table, cells representing tokens coinciding between two witnesses are marked ('scored') in this case with dots

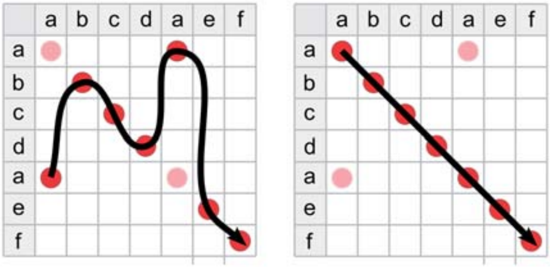


Fig. 5 a(left), and b (right): Unrealistic alignment conclusion (a) versus natural, elegant, or reader's common sense solution (b)

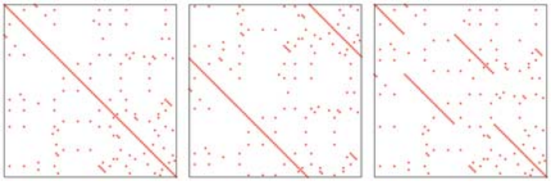


Fig. 6 a(left), b (center), and c (right)

witness 2 as the row headers of the matrix. We then simply mark cells that have identical row and column headers.

This simple case reveals an essential aspect of document-to-document matrices for variant detection: in general, the 'path' from the upper left corner to the bottom right corner that deviates the least from the exact diagonal corresponds to the similarity a reader would assume between two texts. A reader would not assume, for example, that the first 'the' in the horizontally depicted witness is actually to be identified with the second 'the' in the vertically represented witness, or vice versa. Thus, we assume the 'conclusion' depicted for the generalized case in Fig. 5a to be invalid, and the solution in Fig. 5b is preferred. This case also gives us a hard constraint for any algorithm design: we cannot select more than one token for each row and/or column, i.e. we cannot have two tokens simultaneously in one position.

Now consider a case of transposition. Text A is a six-sentence (or ninety-seven–word) quote from Samuel Beckett's *Stirrings Still*. When we compare two identical copies of this text in a matrix as explained above, we arrive on the result as depicted in Fig. 6a. Now we copy text A, but we deliberately move the last sentence to the start of the text. In this way, we effectively create an artificial transposition. We now compare the original text to the copy containing the artificial transposition. The result is depicted in Fig. 6b. The displaced sentence is clearly indicated by a diagonal in the top right corner, attesting that the last sentence of the original (horizontal direction) coincides with the start of the altered copy. If we created multiple transpositions, we would get a result as depicted in Fig. 6c. Instead of a clear diagonal, all the way through we find a rather broken-up path of smaller diagonals:

> One night as he sat at his table head on hands he saw himself rise and go. One night or day. For when his own light went out he was not left in the dark. Light of a kind came from the one high window. Under it still the stool on which till he could or would no more he used to mount to see the sky. Why he did not crane out to see what lay beneath was perhaps because the window was not made to open or because he could or would not open it.

Text A: excerpt from Samuel Beckett's *Stirrings Still*.

It will be clear from this example that in fact any changes (or 'edits' as they are called in computer science, be they intentional or not) in an initially identical copy of a text will result in a deviation from a perfect diagonal in the document-to-document matrix—even the substitution of a single character. Many such edits cause the visible diagonal of a perfect alignment to be broken up in a large collection of longer and shorter diagonals, similar to what is shown in Fig. 6c (but many times larger for real-world texts). We call these dispersed diagonals match phrases. Like the much simpler case represented in Fig. 5, in a real-world case it remains CollateX's task to determine what sequence of match phrases corresponds to the 'natural' alignment of two texts or witnesses. Of course, being a computer program, CollateX has no conception of the kind of alignment a human reader would infer, and to further complicate matters, human readers can actually have different opinions on what the 'best' alignment is. Therefore, CollateX has to rely on a mathematical approximation of the inferences human readers might make. Congruent to the argument of Bourdaillet and Ganascia (2007), the approach of CollateX for this is to determine the set of match phrases that corresponds to the smallest number of edits between two witnesses. In other words, the algorithm determines the smallest set of longest match phrases that accounts for all variants between two texts, as conceptualized in Fig. 7.

Theoretically, this process can be applied to an *n*-dimensional matrix. This would facilitate comparing an arbitrary amount of witnesses, or in other words, support for multiple witness alignment. However, the time needed to compute the *n*-dimensional case is not linear, but probably in the order of an *n*-exponential function of the text length, making it computationally unattainable in a reasonable amount of time. Therefore, multiple witness alignment must be supported in another way, as explained in the remainder of this section.

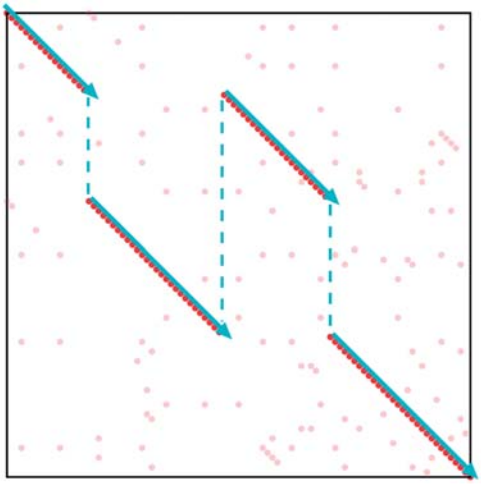


Fig. 7 Vectors describing alignment of two witnesses in a document-by-document matrix

To register the alignment and variation between witnesses traced by the algorithm, an efficient way to store the algorithm's results is needed. To this end, CollateX adopts Schmidt and Colomb's concept of a variant graph (Schmidt and Colomb, 2009). We will demonstrate this process using the case of a variant token. In Fig. 8a, we see the algorithm determining the first alignment. The algorithm traverses all cells that represent aligned tokens and adds a vertex for each, the edges of which are indexed for both witnesses. However, in the case of the token 'i', we traverse an empty column. This means that we hit a token that is represented in one witness but that does not have a corresponding token at that location in the other witness. Instead, the other witness has 'j'. In this case, we add two vertices with indexed edges, one for each witness (cf. Fig. 8b). This process ultimately results in the situation depicted in Fig. 8c.

CollateX's variant graph is a directed acyclic graph in which each vertex represents a token in one or more versions. Each edge in a variant graph is annotated with one or more identifiers for each version. Additionally, a variant graph has a start and an end vertex (the #-vertices in Fig. 8) that do not represent any tokens. By proceeding in this way, variation at the start or the end of a version can be recorded. When one reads the variant graph from left to right, following only the edges annotated with the identifier of a certain version, the complete token sequence of this version can be reconstructed. When transpositions are detected, a

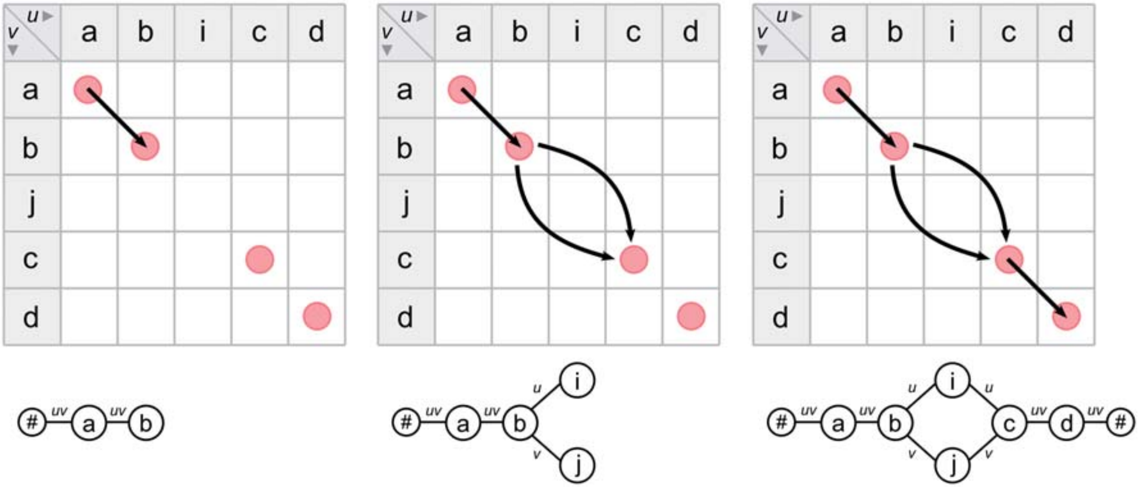


Fig. 8 a (left), b (center), and c (right): Storing alignment results using a variant graph

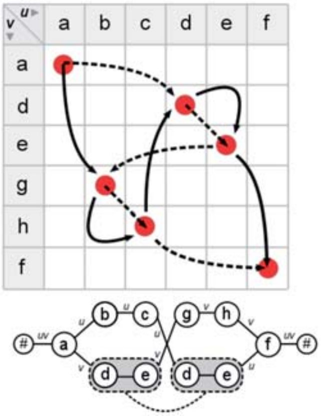


Fig. 9 Capturing a transposition in a variant graph

duplicate of the transposed vertex is created and the two copies are linked together (cf. Fig. 9).

By applying a variant graph, we can also extend the applicability of the described algorithm from pairwise comparisons to the alignment of more than two versions. To this end, we apply the same approach of matrix-wise comparison, but instead of a 2D matrix a 3D matrix is used. This allows us to compare a new witness with the entire variant graph constructed so far. This process is conceptualized in Fig. 10 where a new witness 't' (with a reading identical to witness 'u') is added to the comparison. Darker 'cubes' visualize alignment between the existing graph and the new witness. (Lighter 'cubes' are for the readers' orientation within the 3D matrix only.) Eventually, in this case, the process leads to the addition of an index 't' to all edges in the graph that had an index 'u'. Of course, when a new reading is found that has not been recorded in the graph, a new vertex would be inserted.

In this way, the graph represents a serialization of the variation between the documents of a steadily growing set. The serialization contains all the variation that is recorded during the alignment of previous comparisons and is derived from the variant graph by arranging the tokens of all vertices in topological order. Note that a variant graph ultimately describes variation between witnesses; it does not—nor does CollateX—interpret or infer the type or cause of variation. Thus, in a graph such as depicted in Fig. 11, there is no given interpretation whether the 'j' results from an addition in one witness or has been deleted in another. Of course, whenever there is additional knowledge on the provenance and the date of the witnesses, this inference becomes a trivial task in most cases.

As we noted above, CollateX's development as a whole is still in an experimental stage, but the current version (as of the time of writing version 1.3 is available on http://www.collatex.net) yields

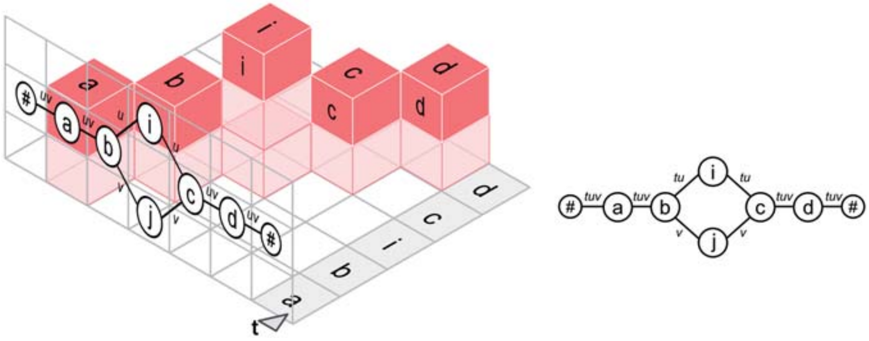


Fig. 10 Visualization of multiple witness comparison using a 3D matching matrix

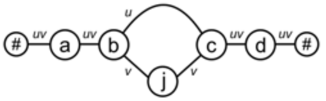


Fig. 11 Variant graph recording a variant that is either a deletion in witness ‘u’ or an addition in witness ‘v’

convincing results. Inspection of a 10% sample of a real-world test collating the first chapter of Darwin’s *Origin of Species* (Bordalejo 2009) yielded a 100% correct detection rate for textual variation, and 87.5% correct rate of transposition identification (i.e. one false positive and seven correct identifications of transpositions, judged by human control) within the sample. CollateX’s performance as to speed may leave something to wish for. Currently, sentence- to paragraph-sized collations are executed in less than a second, virtually independent of the amount of witnesses. Collation of chapter-sized texts is feasible (seconds), but at larger sizes (‘book length’), the speed degrades rapidly to unfeasible. To this end, chunking or breaking larger bodies of texts into smaller parts is an effective solution. A number of identified problems remain to be addressed in future work. Most importantly, the independence of the alignment results from the order in which the versions are aligned needs more testing. Although no dependence on the order could be witnessed in test cases found in other publications addressing the issue (Spencer and Howe, 2004), it is possible that, for example, a combination of repeated tokens in versions and a change in the order of their comparison might cause different results. Another issue is testing and benchmarking. CollateX’s algorithm is tested against an ever growing set of real-world use cases varying from simple and constructed cases such as ‘the black cat and the white dog’ versus ‘the white cat and the black dog’ to elaborate fragments of Armenian, medieval Dutch, and Hebrew prose and verse. This yields good use-case–based evidence that CollateX is indeed capable of tracing complicated examples of real-world textual variation. The development methodology used (‘agile’[14]) implies an ever growing set of such real-world cases, as new users request test runs of previously unseen material. However, there is also a need for a mathematically/computationally constructed larger test corpus of variant texts of which the variation is exactly known and modeled on real-world textual variation, so that future releases of CollateX can be benchmarked for accuracy and performance to a certain standard. Creating such a test corpus is an important step in future research and development.

2 Integrating CollateX within the infrastructure of a digital edition

Now that the conceptual framework and algorithm have been mapped out, we would like to address the issue of possible integrations of CollateX into electronic editions. The Beckett Digital Manuscript Project proved to be a suitable test case since its software infrastructure is rather typical for the way in which many digital editions are currently composed.

It uses Apache Cocoon,[15] a publishing framework, which is based entirely on XML-oriented technologies. As XML and its adjacent standards are almost ubiquitous in today's Digital Humanities landscape, ranging in their application from the encoding of source material to the publication in multiple XML-based formats like XHTML or PDF via XSL-FO, Apache Cocoon is widely deployed among the projects in this field. The framework is built around the idea of configurable transformation scenarios. Such scenarios are mapped flexibly to the URI namespace of a project's Web site and are triggered as soon as a web client sends a request to any of the mapped URIs. Cocoon then

(1) Takes input parameters from the request;
(2) Pulls additional relevant data from a variety of data sources (e.g. XML databases, relational databases, web services, or a server's file system);
(3) Converts all data into an XML document;
(4) Pushes the data through a chosen transformation scenario, configurable by the site's developer as well as the user, and
(5) Returns the transformation result in the response to the web client (Fig. 12).

The Beckett Digital Manuscript Project follows this pattern in as much as it

(1) Receives requests for specific textual resources in the edition, selected via an appropriate URI;
(2) Pulls those resources, encoded in TEI-P5 compliant XML, from the server's file system;
(3) Applies an XSLT-based transformation to the XML-encoded text resource that is fitted to the desired output format (often (X)HTML) and the current site context in which the user requests the resource, and
(4) Delivers the transformation result along with any static resources (images, stylesheets, and client-side script code) to the requesting client (often a web browser).

To seamlessly integrate CollateX's functionality in this site architecture, it was deemed to be the most elegant approach to think of the collator module as another transformation scenario, which transforms data from a selected set of text versions into an intermediary XML-based format encoding the collation result. As described above in the section on CollateX's design, the modularity of CollateX allows us to uncouple the preprocessing of input data and the postprocessing of collation results from the core of its functionality, the alignment. Because of this flexibility, it was comparatively easy to embed CollateX into Apache Cocoon as another transformer module. All that was needed was

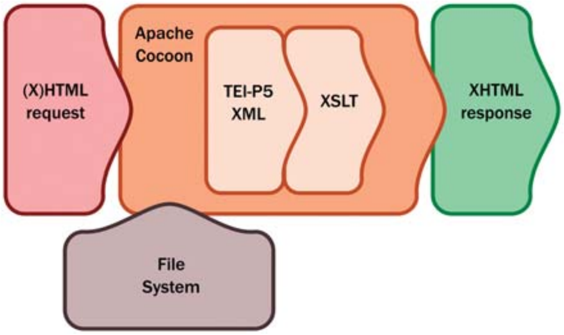


Fig. 12 Usual blueprint for Apache Cocoon architecture-based web services

(1) A preprocessing step, which transformed an XML-encoded set of versions into tokenized input to the alignment module of CollateX, and
(2) A postprocessing step, which renders the results of the alignment step in an XML-based format, so it can be further processed by Apache Cocoon's components and ultimately delivered to the user in the form desired (cf. Fig. 13).

More specifically, the transformer module looks for so-called 'data islands' in provided XML input documents, which resemble the following snippet:

```
<cx:collation xmlns:cx="http://interedition.eu/
collatex/ns/1.0" cx:outputType="tei">
        <cx:witness>...</cx:witness>
        <cx:witness>...</cx:witness>
        <cx:witness>...</cx:witness>
        ...
</cx:collation>
```

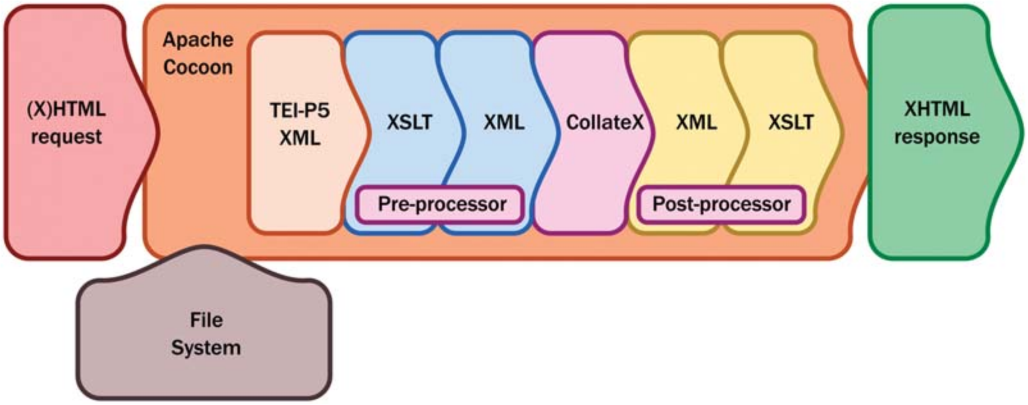


Fig. 13 Blueprint of the integrated Apache Cocoon and CollateX architecture-based web service for the Beckett Digital Manuscript Project

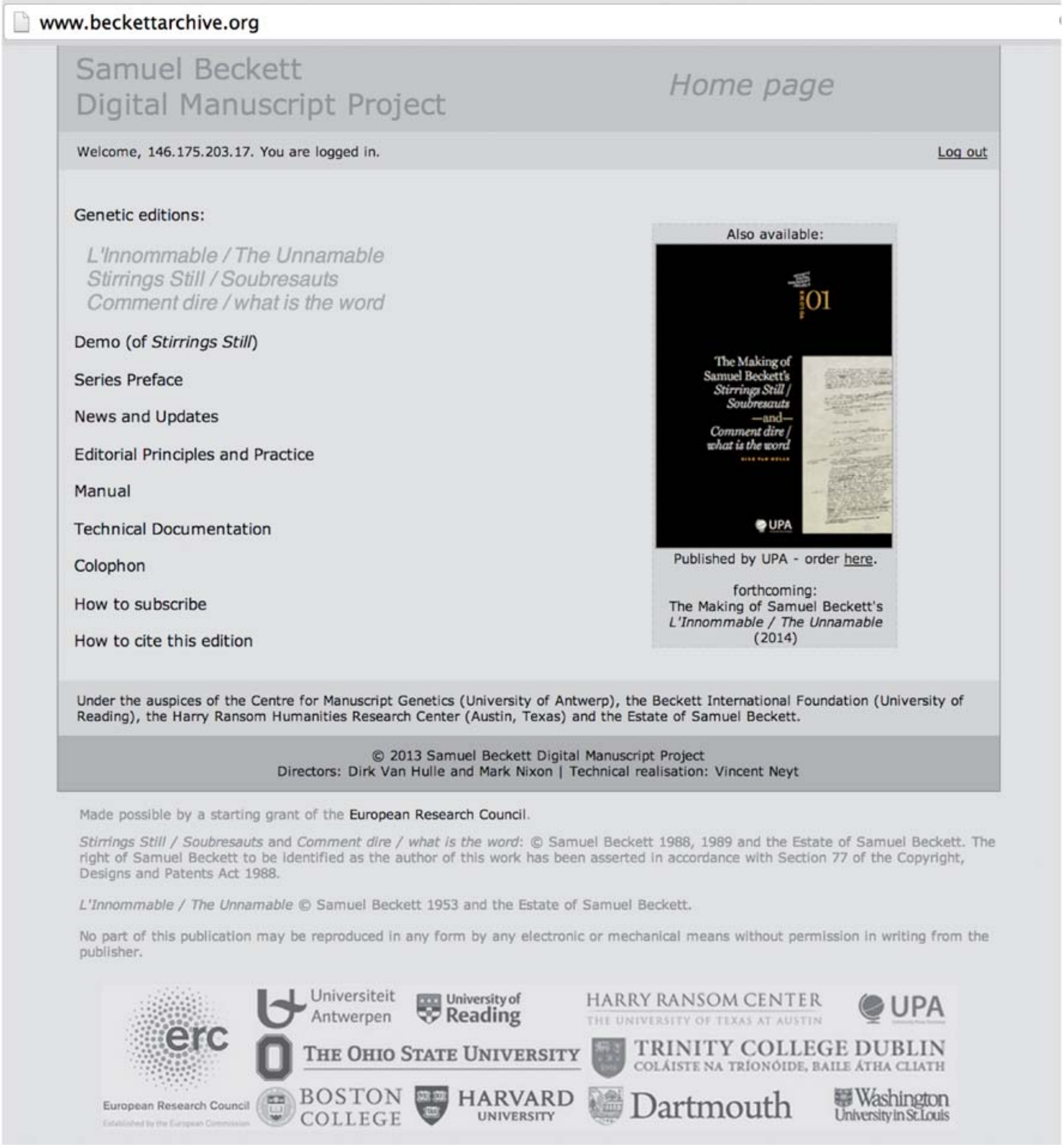
Whenever the transformer encounters such an island in an input document, it substitutes it with the resulting alignment of the given versions/witnesses in the output while just copying the surrounding data. The encoding of the output can be controlled via the attribute 'cx:outputType'. Currently a proprietary, tabular data format and TEI P5 parallel segmentation markup are supported. By assembling these data islands, dynamically based on the user's request (parameterized for instance via the input of an HTML form; see e.g. Fig. 14), and by adding consecutive transformation modules after the collation has been performed, the Beckett Digital Manuscript Project can provide for the described personalization of its critical apparatus.

Because of the modular design of CollateX, a seamless integration of its functionality with the site infrastructure of the Beckett Digital Manuscript Project has been achieved. The amount of work was comparatively limited because the CollateX team was able to reuse most of the already developed components and the Beckett project's team was able to build the integration entirely on their existing code base and platform.

However, the integration did not come without trade-offs. Particularly, the synchronous online execution of the collation considerably limits the amount of textual data that can be collated. This constraint and the requirement of more complex request/response choreographies than the ones Cocoon provides out of the box as soon as larger texts are collated, makes this a solution which cannot simply be adopted without prior adjustments by any edition with arbitrarily large text traditions. But projects such as NINES, with its collation software Juxta,[16] and the Interedition project itself develop web-service–based solutions, which aim to overcome scalability issues related to the synchronous on-the-fly collation in use today. Therefore, the adoption of these solutions and again their integration with existing infrastructures like Apache Cocoon offers a promising perspective.

3 CollateX, modern manuscripts, and the digital scholarly editorial process

Among all the interoperable tools developed within the Interedition framework CollateX takes a special place, as it realizes interoperability not just in a technical sense but also in a scholarly sense. Following textual scholars such as Bryant (2002), Buzzetti (2002), and McGann (2001) and genetic critics such as Lebrave (1993), Grésillon (1994), Hay (2002), de Biasi (2004), Ferrer (2011), and , we have come to appreciate text in its essential fluidity and its forms as a process rather than a static object.



www.beckettarchive.org

Samuel Beckett
Digital Manuscript Project

Home page

Welcome, 146.175.203.17. You are logged in.

Log out

Genetic editions:

L'Innommable / The Unnamable
Stirrings Still / Soubresauts
Comment dire / what is the word

Demo (of *Stirrings Still*)

Series Preface

News and Updates

Editorial Principles and Practice

Manual

Technical Documentation

Colophon

How to subscribe

How to cite this edition

Also available:

01
The Making of Samuel Beckett's *Stirrings Still / Soubresauts* —and— *Comment dire / what is the word*
UPA

Published by UPA - order here.

forthcoming:
The Making of Samuel Beckett's
L'Innommable / The Unnamable
(2014)

Under the auspices of the Centre for Manuscript Genetics (University of Antwerp), the Beckett International Foundation (University of Reading), the Harry Ransom Humanities Research Center (Austin, Texas) and the Estate of Samuel Beckett.

© 2013 Samuel Beckett Digital Manuscript Project
Directors: Dirk Van Hulle and Mark Nixon | Technical realisation: Vincent Neyt

Made possible by a starting grant of the European Research Council.

Stirrings Still / Soubresauts and *Comment dire / what is the word*: © Samuel Beckett 1988, 1989 and the Estate of Samuel Beckett. The right of Samuel Beckett to be identified as the author of this work has been asserted in accordance with Section 77 of the Copyright, Designs and Patents Act 1988.

L'Innommable / The Unnamable © Samuel Beckett 1953 and the Estate of Samuel Beckett.

No part of this publication may be reproduced in any form by any electronic or mechanical means without permission in writing from the publisher.

erc
European Research Council
Universiteit Antwerpen
University of Reading
HARRY RANSOM CENTER
THE UNIVERSITY OF TEXAS AT AUSTIN
UPA
THE OHIO STATE UNIVERSITY
TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH
BOSTON COLLEGE
HARVARD UNIVERSITY
Dartmouth
Washington University in St. Louis

Fig. 14 The Beckett Digital Manuscript Project

The dynamic aspect of text has caused especially McGann to argue that any electronic edition—or for that matter any attempt in that direction—should be based on dynamic processes, ideally implementing ‘flexible and multi-dimensional views of the materials’ (McGann 2001). Originally, McGann envisaged such editions in the form of hypermedia editions backed by relational databases, on top of which adaptable transformational logic would cause archived digital texts to be represented according to specific editorial practices and views of different editorial or literary communities. Of course, CollateX does not fully answer to such an ambitious perspective. However, it is interesting to note that CollateX represents at least one aspect of such a transformational logic as McGann pointed to. CollateX effectively dynamically 'reverse engineers' the variation present in textual tradition or genesis. The process is dynamic because the basic process is independent of tedious scholarly manual labor, causing CollateX's transformations to be repeatable. The process is also dynamic, as it is adaptable. By adding witnesses present in the database-backed electronic archive to the analysis (or by removing some of them from it), the effect and perspective on the collation may change. In this sense, the application of CollateX in editorial processes takes us one step further in our abilities to express the dynamics of text production, editing, and reception. At this point, the software can be serviceable in the preparation of a scholarly edition, since it can also output TEI parallel segmentation XML, which an editor can then transform and visualize the way he or she wants within the edition. Projects such as 'manuscript archives' that do not envisage the development of a full 'historical-critical' edition, could still offer their users an alternative to a traditional 'critical apparatus'. Comparable with aspects of what is currently often labeled social editing (Siemens 2012), embedding CollateX in the Beckett Digital Manuscript Project enables the user to make his or her own selection of textual versions that need to be collated and leave out the ones he or she is not immediately interested in.

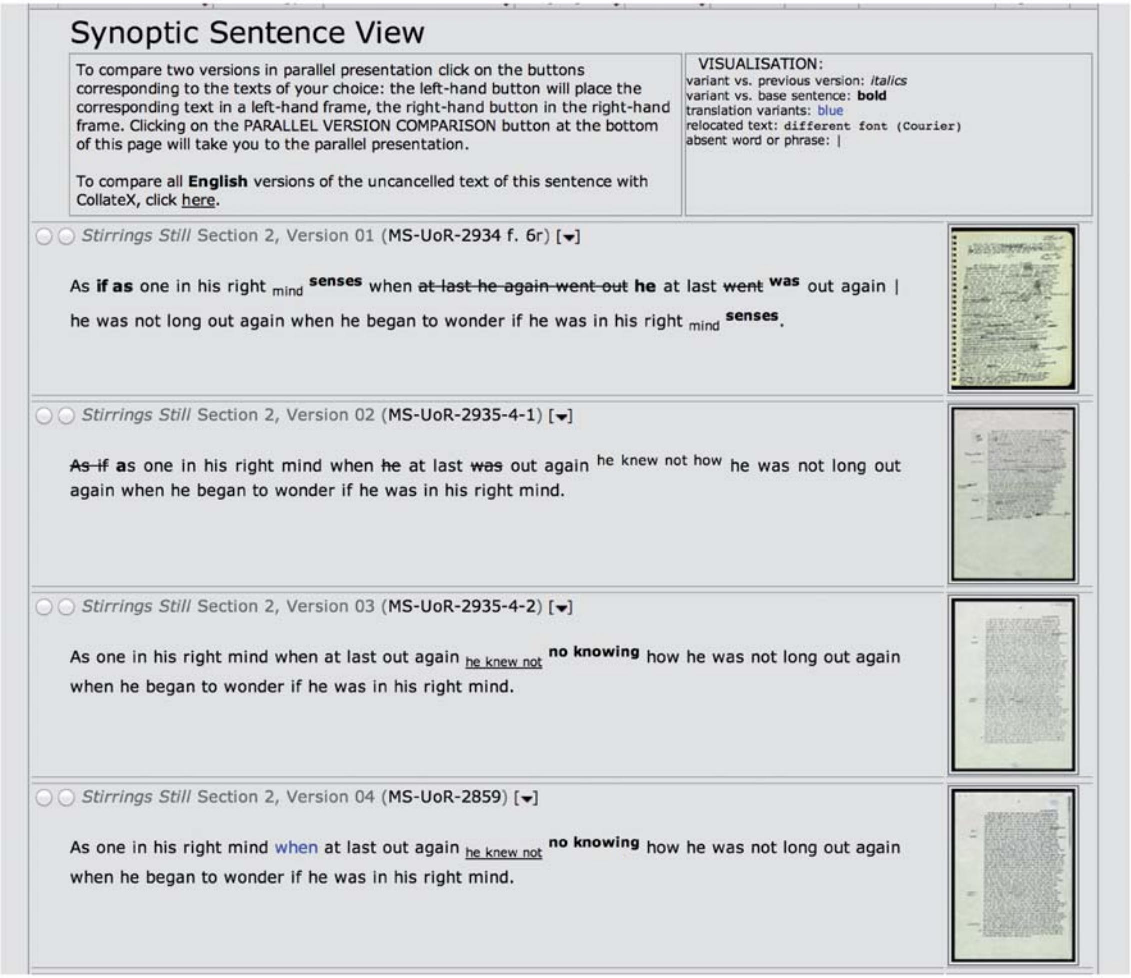
From the vantage point of editorial theory, this development has interesting consequences regarding the scholarly editor's role, whose focus may shift from the collation to a more interpretive function. In this way, the integration of a collation tool may be consequential in terms of bridging the gap between scholarly editing and genetic criticism. From the perspective of editorial practice, the application of CollateX is still at an experimental stage, but it already shows that the modular and service-oriented approach used by Interedition has the potential to be useful, both to the specialized field of digital scholarly editing and to a more general audience.

3.1 Pushing the collation envelope: modeling genetic stages

Apart from facsimiles, topographic, and linear transcriptions (encoded in XML), the Beckett Digital Manuscript Project provides the option to compare the different preparatory versions of the text—from the earliest draft stages to the page proofs. To avoid getting lost, the user is able to compare a particular segment in one version with the same segment in another version, or in all the other versions. The size of such a segment is determined by the user, the smallest unit being the sentence.

The user is offered a synoptic survey of all the extant versions of the segment of his or her choice, showing each version in its entirety with the variants highlighted (cf. Fig. 15). The syntactical context of each segment remains intact, but in order for the variants to be highlighted, they had to be encoded first. In view of the large amount of manuscript materials still to be transcribed, the project would not have been able to include the option of encoding an apparatus in all of the transcriptions. As an alternative to that manual encoding task, we tested the possibilities of digitally supported collation by means of the CollateX algorithm.[17] One of the complicating elements of this test case is the rather large number of versions in combination with the presence of deletions and additions in almost all of them.

To find solutions for the latter complicating element, there are several ways of looking at the challenge of collating modern manuscripts. One way would be to regard it as a form of collation that does not only collate versions of a text, but also stages within versions. For one manuscript, version can often be subdivided into several writing stages. A writing stage is defined, according to the suggestions of the TEI Special Interest Group (SIG) on 'Genetic Editions', as 'The a reconstructable stage in the evolution of a text, represented by a document or by a revision campaign within one or more documents, possibly assigned to a specific point in time'.[18] Ideally, this would require that the editor can identify not only different stages in the writing process, but also the writing sequence within each writing stage. If these sequences and stages can be discerned unequivocally, it would be theoretically



Synoptic Sentence View

To compare two versions in parallel presentation click on the buttons corresponding to the texts of your choice: the left-hand button will place the corresponding text in a left-hand frame, the right-hand button in the right-hand frame. Clicking on the PARALLEL VERSION COMPARISON button at the bottom of this page will take you to the parallel presentation.

To compare all **English** versions of the uncancelled text of this sentence with CollateX, click here.

VISUALISATION:
variant vs. previous version: *italics*
variant vs. base sentence: **bold**
translation variants: blue
relocated text: `different font (Courier)`
absent word or phrase: |

○○ *Stirrings Still* Section 2, Version 01 (**MS-UoR-2934 f. 6r**) [▾]

As **if as** one in his right mind **senses** when ~~at last he again went out~~ **he** at last ~~went~~ **was** out again |
he was not long out again when he began to wonder if he was in his right mind **senses**.

○○ *Stirrings Still* Section 2, Version 02 (**MS-UoR-2935-4-1**) [▾]

~~As if~~ **as** one in his right mind when ~~he~~ at last ~~was~~ out again he knew not how he was not long out again when he began to wonder if he was in his right mind.

○○ *Stirrings Still* Section 2, Version 03 (**MS-UoR-2935-4-2**) [▾]

As one in his right mind when at last out again he knew not **no knowing** how he was not long out again when he began to wonder if he was in his right mind.

○○ *Stirrings Still* Section 2, Version 04 (**MS-UoR-2859**) [▾]

As one in his right mind when at last out again he knew not **no knowing** how he was not long out again when he began to wonder if he was in his right mind.

Fig. 15 Synoptic survey of various version of one textual segment in the Beckett Digital Manuscript Project

possible to treat each stage as a version (or 'witness') to be collated.

The TEI SIG on Genetic Editions suggested working with 'stageNotes' to describe the composition stages that have been identified in the genesis of a text. These 'stages' relate to the relatively large unit of the textual version as a whole ('Textfassung'). Within a stage (say, an author writing a block of text in black ink, deleting, and adding words in the same writing tool) it is often difficult, if not impossible, to further discern different 'substages'. Still, a genetic critic might be interested in a collation tool that brings to the fore precisely this kind of moment in the writing process, when the writer did not immediately find the right words. In the case of a simple example, 'The ^black^ ~~cat~~ ^dog^ is ~~alive~~ ^dead^'—assuming all deletions (-) and additions (^) are made in the same handwriting and writing tool—all of the following combinations are theoretically possible:

W1a: the cat is alive
W1b: the black cat is alive
W1c: the cat is dead
W1d: the black cat is dead
W1e: the dog is alive
W1f: the black dog is alive
W1g: the dog is dead
W1e: the black dog is dead

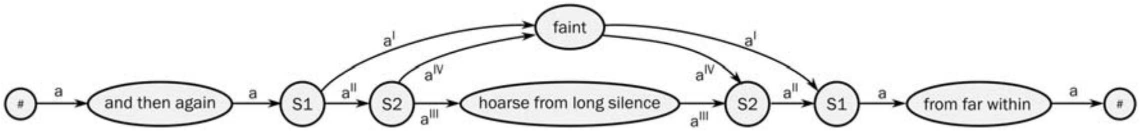


Fig. 16 Conceptual CollateX variant graph capturing genetic stages of authoring

The TEI SIG ‘Genetic Editions’ developed the ‘stageNote’ element for the documentary level. According to the TEI SIG’s suggestions, ‘A genetic editor needs to be able to assign a set of alterations (deletions, additions, substitutions, transpositions, etc.) and/or an act of writing to a particular stage’.[19] However, some authors always use the same writing tool, not only for the ‘first stage’ of their draft but for all subsequent revision campaigns. Moreover, in the TEI suggestions for genetic editions, the ‘stageNote’ element was developed for the ‘documentary’ level (related to what Hans Zeller called ‘Befund’, the record, as opposed to ‘Deutung’, its interpretation), not for the ‘textual’ level. However, collation is a text-related operation. To collate modern manuscripts, it may therefore be beneficial—for the purpose of designing digitally supported collation tools for modern manuscripts—to conceive of the manuscript as ‘a protocol for making a text’, according to Daniel Ferrer’s definition (Ferrer 1998).

A relatively straightforward application of this protocol model is to work with the ‘uncancelled text’ of each manuscript (i.e. a ‘clean’ transcription or reading text of a draft, without the deleted passages, i.e. by ignoring the passages marked by <del> . . . </del> tags). This ‘uncancelled text’ is usually an author’s last ‘protocol’ or instruction to himself when he is on the verge of fair-copying or typing out the text on another document. We tried to apply this ‘uncancelled text’ system to test the first research results of CollateX. All versions of a segment are computed and their data are handed over to CollateX for comparison. The segmentation of the textual material (see above) can now serve an extra purpose: apart from reducing the danger for the user to get lost in the jungle of manuscripts, it also determines the speed of the collation. Since the most frequently chosen textual unit in the project is the smallest segment (usually the unit of a sentence), the number of versions can be relatively high (in the test case: about twenty versions) without slowing down the instant collation.

As an intermediary step, the ‘uncancelled text’ system is useful, but it does reduce the complexity of the manuscript to a textual format. In a way, this pragmatic solution ‘de-manuscripts’ the manuscript. In order to try and refine the computer-assisted collation of modern manuscripts, it would be helpful if the collation software were XML-aware in order for the input to be derived directly from the XML-encoded transcription, and to record changes not only between the stages but also between substages within one stage. The test case provided us with the following example: a passage in one of Beckett’s manuscripts (UoR MS 2934, 9v-10r, written in Beckett’s hand in black ink), with two consecutive substitutions within the same writing stage:

> and then again ~~faint~~ ^~~hoarse from long silence~~
> ^faint^^ from far within

In XML, this could be transcribed as follows:

> and then again <subst xml:id=“subst1”><del
>   xml:id=“del1”>faint</del>
> <add xml:id=“add1”><subst xml:id=“subst2”>
>   <del xml:id=“del2”>hoarse from long
>   silence</del>
> <add xml:id=“add2”>faint</add></subst>
>   </add></subst> from far within

The subst, del, and add tags suffice to cover all stage information, which could be expressed in the ‘augmented’ variant graph of Fig. 16.

Each path in the graph represents a witness. For the purposes of the collation of modern manuscripts, a new type of node has been introduced (in the example, S1 and S2, corresponding to subst1 and subst2, respectively. This writing stage (a) then needs to be compared with other stages or other versions, i.e. with multiple witnesses, say, (b) and (c):

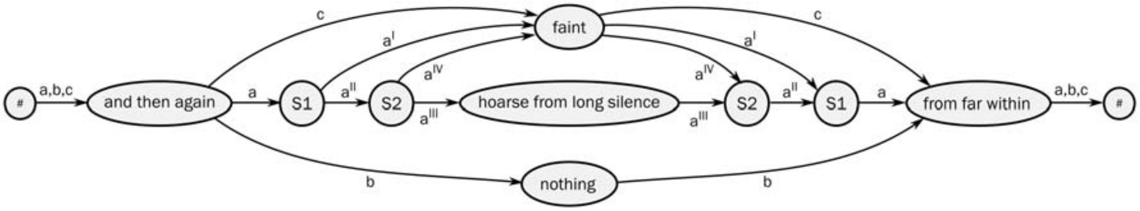


Fig. 17 Conceptual CollateX variant graph capturing genetic stages of authoring as well as witness variation

(a) and then again ~~faint~~ ^~~hoarse from long silence~~ ^faint^^ from far within
(b) and then again nothing from far within
(c) and then again faint from far within

In CollateX's internal model, this would be presented as depicted in Fig. 17. An advantage of this model is that it can support several collation options. By default, CollateX would use only the 'uncancelled text' of each witness, but whereas the 'uncancelled text' model (described above) only took the final protocol into account, this model takes in all the extra information about the cancelled words, saves it, and enables us to 'port out' these data again at the visualization stage. For instance, if for whatever reason, one would prefer to compare (b) and (c) with substage2 of (a), rather than to its 'uncancelled text', the algorithm can optionally be instructed to collate [(b) nothing] and [(c) faint] against [(a') hoarse from long silence], rather than against [(a'') faint]. All the information stored in the XML transcription passes through the collation process untouched, so that it can be retrieved for visualization purposes.

In terms of visualization, an option 'hide cancellations' (as one of the 'Tools' in the menu) could simplify the alignment table, reducing it to a visualization of the different versions' 'uncancelled text' only:

w1	and	then	Again	Faint	from	far	within
w2	and	then	Again	nothing	from	far	within
w3	and	then	Again	faint	from	far	within

But we can imagine that genetic critics and other researchers interested in modern manuscripts might want to have an overview of all the cancellations and substitutions in the manuscripts. Undoing the same 'hide cancellations' option in the menu could offer these users a more complete picture:

				faint			
				~~hoarse from long silence~~			
w1	and	then	again	faint	from	far	within
w2	and	then	again	nothing	from	far	within
w3	and	then	again	faint	from	far	within

The advantage of having introduced the new type of node (S1 and S2 in the variant graph above) is that the 'hoarse from long silence' variant can be treated as one unit during the computer-supported collation and also be presented as such at the visualization stage.

4 In Closing

The collation of modern manuscripts involves the treatment of cancelled text. A classical problem in this area of study is the division of a modern manuscript into 'stages' or even 'substages', for especially if an author uses the same writing tool for all the text on the document (including cancellations and additions), it is often almost impossible to discern separate stages. It is possible to work with the 'uncancelled text' of the documents in order to compare the different versions, but researchers working in modern manuscripts are usually especially interested in the cancellations and substitutions. Therefore, we tried to find a solution for computer-supported collation of modern manuscripts, including cancellations. We have explored how this complex research problem in the application of computers in the humanities could be approached by breaking it down into a community supported and well-defined set of subproblems, which each on its own can be solved in a more flexible and efficient way. Looking beyond the specific problem of computer-supported collation, such an approach does not only appear suitable to us because it is a well-established practice in the construction of complex software systems in general, but also because it allows for effective collaboration among researchers and developers from many different backgrounds and projects. From this perspective, it is not by accident that the development of a modularized collation solution took shape within the context of the research project 'Interedition', whose aim it is to foster such collaboration and to address the organizational and architectural issues associated with such an approach as well, issues which point beyond the development of singular software tools for singular use cases.

References

Bordalejo, B. (2009). Introduction to the online variorum of Darwin's origin of species. http://darwin-online.org.uk/Variorum/Introduction.html (accessed 7 February 2013). In van Wyhe, J. (ed.), (2002). *The Complete Work of Charles Darwin Online*. http://darwin-online.org.uk/ (accessed 7 February 2013).

Bourdaillet, J. and Ganascia, J.-G. (2007). Practical block sequence alignment with moves. *LATA 2007–International Conference on Language and Automata Theory and Applications, 3/2007*. http://www-poleia.lip6.fr/~ganascia/Medite_Project?action=AttachFile&do=view&target=LATA+2007 (accessed 13 May 2013).

Bryant, J. (2002). *The Fluid Text: A Theory of Revision and Editing for Book and Screen.* University of Michigan Press. http://books.google.nl/books?id=1w4wpOdPbu4C (accessed 7 February 2013).

Buzzetti, D. (2002). Digital representation and the text model. *New Literary History*, **33**: 61–88.

de Biasi, P.-M. (2004). Toward a science of literature: manuscript analysis and the genesis of the work. In Jed Deppman, J., Ferrer, D., and Groden, M. (eds), *Genetic Criticism: Texts and Avant-textes*. Philadelphia: University of Pennsylvania Press, pp. 36–68.

Ferrer, D. (1998). The open space of the draft page: James Joyce and modern manuscripts. In Bornstein, G. and Tinkle, T. (eds), *The Iconic Page in Manuscripts, Print, and Digital Culture*. Ann Arbor: University of Michigan Press, pp. 249–67.

Ferrer, D. (2011). *Logiques Du Brouillon: Modèles Pour Une Critique Génétique*. Paris: Éditions du Seuil.

Grafton, A., Most, G.W., and Settis, S. (2010). *The Classical Tradition.* Harvard University Press, p. 506. http://books.google.nl/books?id=LbqF8z2bq3sC (accessed 7 November 2012).

Grésillon, A. (1994). *Éléments De Critique Génétique: Lire les Manuscrits Modernes*. Paris: Presses universitaires de France.

Hay, L. (2002). *La Littérature Des écrivains*. Paris: José Corti.

Lebrave, J. L. (1993). L'édition Génétique. In Cadiot, A. and Haffner, C. (eds), *Les Manuscrits Des écrivains*. Paris: CNRS/Hachette, pp. 206–23.

Levenshtein, V. (1966). Binary codes capable of correcting insertions and reversals. *Soviet Physics: 'Doklady'*, **10**: 707–10.

McGann, J. (2001). *Radiant Textuality. Literature Since the World Wide Web*. New York: Palgrave/St Martins.

Oakman, R. L. (1984). *Computer Methods for Literary Research*. Athens: University of Georgia Press, pp. 118–37.

Schmidt, D. and Colomb, R. (2009). A data structure for representing multi-version texts online. *International Journal of Human-Computer Studies*, **67.6**: 497–514.

Schmidt, D. (2009). *Merging Multi-Version Texts: A Generic Solution to the Overlap Problem. Proceedings of Balisage: The Markup Conference 2009. Montreal: Balisage Series on Markup Technologies, vol. 3*.

Shillingsburg, P. L. (2006). *From Gutenberg to Google: Electronic Representations of Literary Texts.* Cambridge: Cambridge University Press, p. 110.

Siemens, R. (2012). Toward modeling the social edition: an approach to understanding the electronic scholarly edition in the context of new and emerging social media. *Literary and Linguistic Computing*, **27**: 445–61. http://llc.oxfordjournals.org/content/27/4/445.full (accessed 7 November 2012).

Smith, S. E. (2000). The eternal verities verified. Charlton Hinman and the roots of mechanical collation. *Studies in Bibliography*, **53**: 130–62.

Spencer, M. and Howe, C. J. (2004). Collating texts using progressive multiple alignment. *Computers and the Humanities*, **38**: 253–70.

Unsworth, J. (2000). Scholarly Primitives: what methods do humanities researchers have in common, and how might our tools reflect this? *Symposium on 'Humanities*

Computing: Formal Methods, Experimental Practice'. London: King's College. http://people.lis.illinois.edu/∼unsworth/Kings.5-00/primitives.html (accessed 7 February 2013).

Notes

[1] http://www.interedition.eu/ (accessed 7 February 2013).

[2] CollateX is an open source Java collation engine, developed, maintained, and supported by an international open source development community of developers active in the digital humanities community. Having originated and finding a number of applications in several projects of the Huygens Institute for the History of the Netherlands, the final coordination of development currently rests with this institute. However, the code is in the public domain: https://github.com/interedition/collatex (accessed 7 February 2013).

[3] Cf. Shillingsburg, 2006, 110. Scholarly editors consistently overestimate their ability to produce error-free apparatuses, as Dr T.L. Andrews also showed by reverse-engineering a multiple witness apparatus of a classical text. Owing to ambiguity in the conventional apparatus system and errors in the optical collation process, an original witness cannot be reliably reconstructed from textual apparatus. (Presentation during Interedition Schooling programme—Part II: Stemmatic, stylistic, linguistic analysis of collated text—critical edition, representation, publication. Leuven, 7 September 2012; Comparative Oriental Manuscript Studies—Oriental Textual Traditions and 21st-Century Philology: New challenges. Katholieke Universiteit Leuven, 5–7 September 2012.)

[4] For example, the currently ongoing International Greek New Testament Project edits text passages witnessed by up to 100 manuscripts, at which point the comparison of the versions written on them, if indeed it would be performed manually, could only be very selective, which would mean a major constraint for a project concerned about the relationships between all the manuscripts.

[5] Examples of such early software solutions, which despite their comparable lack of algorithmic complexity have their undisputed merits as practical tools, are Robinson's *Collate* or *TUStep*'s collation module, cf. Oakman, 1984.

[6] Mapping this field and assessing its influence on computer-supported collation in a way that would do justice to its relevance is beyond the scope of this article. For a very good overview and one such attempt see Schmidt (2009).

[7] Again Robinson's Collate and TUStep's modular approach are examples for this kind of collation software.

[8] Schmidt's *Nmerge* and Ganascia's *MEDITE* are examples for such automated solutions.

[9] http://www.cost-a32.eu/ (accessed 22 May 2013).

[10] Separation of concerns is an important principle for subdividing and decoupling complex problems in IT development to reduce the effect of dependencies between solutions. The division between XML (data), XSLT (transformative logic), and XHTML (presentation) is an excellent example of the same principle at work.

[11] http://www.juxtasoftware.org/ (accessed 14 May 2013).

[12] https://github.com/interedition/text (accessed 7 February 2013).

[13] Development is organized within the open source community and lead by the University of Virginia's NINES project, see http://code.google.com/p/juxta/ (accessed 14 May 2013).

[14] See http://en.wikipedia.org/wiki/Agile_software_development (accessed 7 February 2013).

[15] http://cocoon.apache.org/ (accessed 27 October 2011).

[16] http://www.juxtasoftware.org/ (accessed 27 October 2011).

[17] The Beckett Digital Manuscript Project publishes Beckett's manuscripts, organized in a series of modules, on a yearly basis. The first module, published in 2011, does not yet contain the full Cocoon integration of CollateX as discussed above, but works with a more basic integration by means of a REST service. The full Cocoon integration is in a demo-stage and will possibly be included from the publication of the next module. In the case of both integrations, the collation happens on the fly and the output may take several forms, one of them being a table in which columns with invariant and variant passages are aligned.

[18] http://www.tei-c.org/Activities/Council/Working/tcw19.html

[19] http://www.tei-c.org/Activities/Council/Working/tcw19.html (accessed 27 October 2012).

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 31360911.